# Variational Inequalities: Algorithms

Anna Nagurney
Isenberg School of Management
University of Massachusetts
Amherst, MA 01003

**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.

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.

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**

General Iterative Scheme of Dafermos which induces such algorithms as:

The Projection Method and

The Relaxation Method.

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.

## The General Iterative Scheme

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.

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.

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

**Algorithm**

**Step 0: Initialization**

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

**Step 1: Construction and Computation**

Compute $x^k$ by solving the variational inequality subproblem:

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

**Step 2: Convergence Verification**

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

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

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

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

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 $k$, this subproblem is easy to solve.

Note that if the sequence $\{x^k\}$ is convergent, i.e., $x^k \to x^*$, as $k \to \infty$, 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*

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

*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* $R^n$.

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.

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:

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

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 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 $k := 1$ and select $\rho$, 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 $\bar{x}^{k-1}$ by solving the variational inequality subproblem:

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

**Step 2: Adaptation**

Compute $x^k$ by solving the variational inequality subproblem:

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

**Step 3: Convergence Verification**

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

The modified projection method converges to the solution of $VI(F, K)$, 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. 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.

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 $i$.

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.

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*

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

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 $k := 1$; $i := 1$.

**Step 1: Linearization and Computation**

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

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

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

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

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

**Step 2: Convergence Verification**

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

## Linearized Decomposition Algorithm - Parallel Version

**Step 0: Initialization**

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

**Step 1: Linearization and Computation**

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

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

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

**Step 2: Convergence Verification**

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

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

If $A_i(x^{k-1}) = \nabla_{x_i} F_i(x^{k-1})$, then a Newton's method is obtained.

If $A_i(x^{k-1}) = D_i(x^{k-1})$, 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.

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.

A convergence theorem for the above linearized decomposition algorithms is now presented.

**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 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 $k := 1$; $i := 1$.

**Step 1: Relaxation and Computation**

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

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

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

**Step 2: Convergence Verification**

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

The parallel analogue is now given.

**Nonlinear Decomposition Algorithm - Parallel Version**

**Step 0: Initialization**

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

**Step 1: Relaxation and Computation**

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

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

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

**Step 2: Convergence Verification**

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

A convergence theorem for the above nonlinear decomposition algorithms is now given.

**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. 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.

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.

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.

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).

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."

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

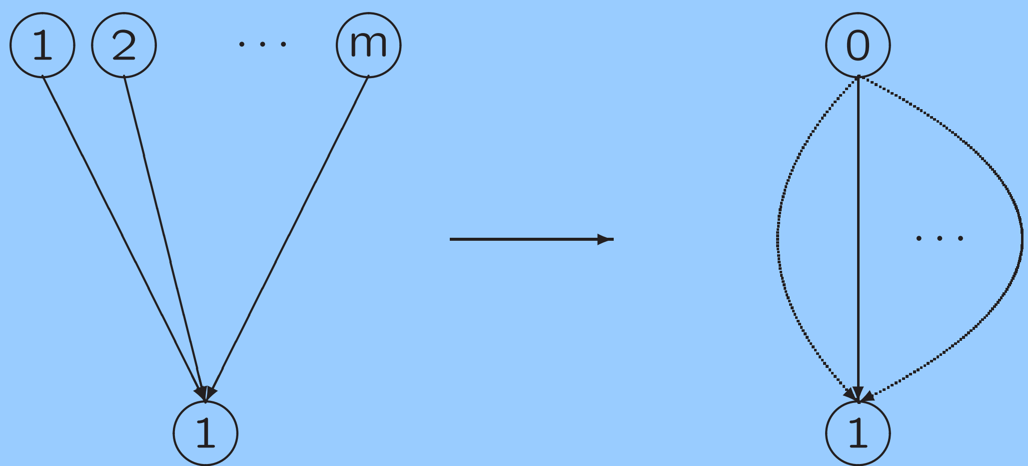
## Demand Market Equilibration Algorithm

For simplicity, we begin with an exact "demand" market equilibration algorithm which can be applied to the solution of a single O/D pair problem with elastic demand (and disjoint paths, that is, with paths having no links in common).

In particular, we are interested in computing the equilibrium "trade flows" or shipments from $m$ supply markets to the $l$-th demand market, say, satisfying the equilibrium conditions: The cost of the good from $i$ to $l$, $g_i x_{il} + h_{il}$, is equal to the demand price $-r_l \sum_{i=1}^{m} x_{il} + h_{il}$, at demand market $l$, if there is a positive shipment of the good from $i$ to $l$; if the cost exceeds the price, then there will be zero shipment between the pair of markets. Mathematically, these conditions can be stated as: For each supply market $i$; $i = 1, \ldots, m$,

$$g_i x_{il}^* + h_{il} \begin{cases} = -r_l \sum_{i=1}^{m} x_{il}^* + q_l, & \text{if} \quad x_{il}^* > 0 \\ \geq -r_l \sum_{i=1}^{m} x_{il}^* + q_l, & \text{if} \quad x_{il}^* = 0. \end{cases}$$

Here $g_i, h_{il}, r_l$, and $q_l$ are all assumed to be positive.




**Single origin/destination problem with disjoint paths**

The algorithm for the solution of this problem is now presented. It is a finite algorithm, in that the problem is solved in a finite number of steps.

**Demand Market Exact Equilibration**

**Step 0: Sort**

Sort the $h_{il}$'s in nondescending order and relabel the $h_{il}$'s accordingly. Assume, henceforth, that they are relabeled. Define $h_{m+1,l} = \infty$.

If $q_l \leq h_{1l}$, then

$$x_{il}^* = 0, \quad i = 1, \ldots, m,$$

and stop; otherwise, set $v := 1$.

**Step 1: Computation**

Compute

$$\rho_l^v = \frac{\sum_{i=1}^{v} \frac{h_{il}}{g_i} + \frac{q_l}{r_l}}{\sum_{i=1}^{v} \frac{1}{g_i} + \frac{1}{r_l}}.$$

**Step 2: Evaluation**

If $h_{vl} < \rho_l^v \leq h_{v+1,l}$, then stop, set $s' = v$, and go to Step 3. Otherwise, let $v := v + 1$, and go to Step 1.

**Step 3: Update**

Set

$$x_{il}^* = \frac{\rho_l^{s'} - h_{il}}{g_i}, \quad i = 1, \ldots, s'$$

$$x_{il}^* = 0, \quad i = s'+1, \ldots, m.$$

In the fixed case, where the demand $\sum_{i=1}^{m} x_{il}$ is known, the procedure that will equalize the costs for all positive trade flows can be obtained from the above scheme by replacing the $\frac{q_l}{r_l}$ term in the numerator by the known demand, and by deleting the second term in the denominator.

Of course, if, instead, one seeks to compute the equilibrium flows from a particular supply market $i$ to $n$ demand markets, then one can construct analogous supply market exact equilibration algorithms for the elastic supply and the fixed supply cases.

Note that equilibrium conditions are equivalent to the solution of the quadratic programming problem:

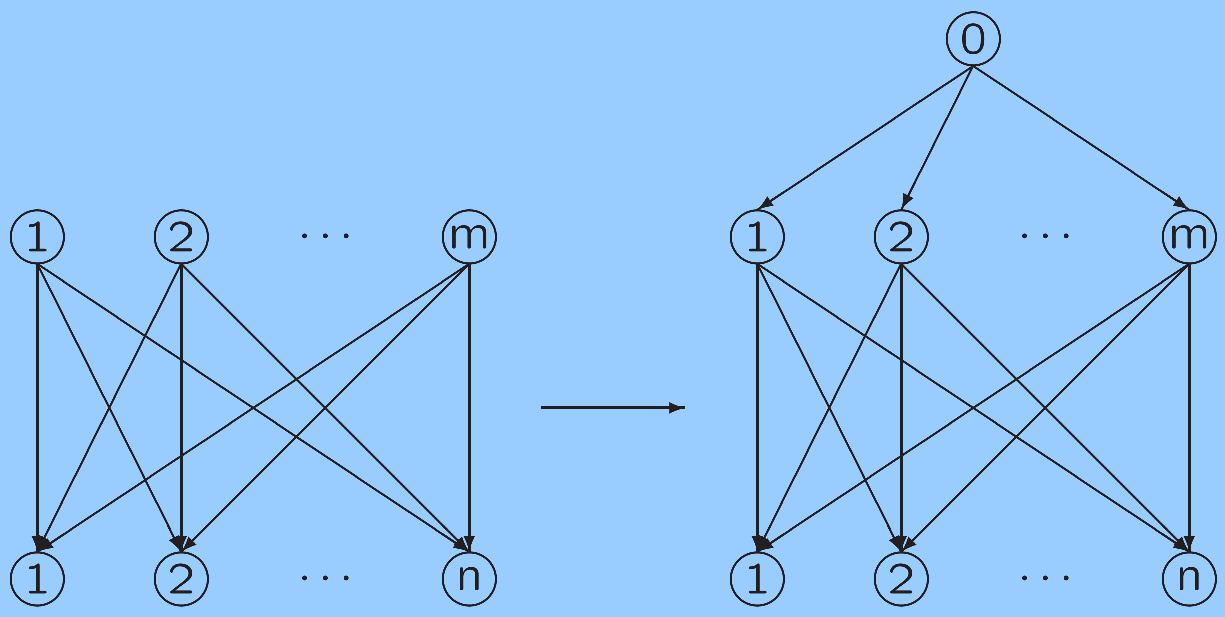
$$\min_{x_{ij} \geq 0, \forall i,j} \sum_{i=1}^{m} (\frac{1}{2} g_i x_{il}^2 + h_{il} x_{il}) + \frac{1}{2} r_l (\sum_{i=1}^{m} x_{il})^2 - q_l \sum_{i=1}^{m} x_{il}.$$

Indeed, it is easy to verify that the Kuhn-Tucker conditions of the optimization problem are equivalent to the equilibrium conditions above. Hence, although any appropriate optimization algorithm could be used to compute the equilibrium flows for this particular problem, the above procedure does possess certain advantages, specifically, finiteness, and ease of implementation.

The importance of the above procedure lies not only in its simplicity but also in its applicability to the computation of a wide range of equilibrium problems.

For example, equilibration can be used to solve an embedded quadratic programming problem when an appropriate variational inequality algorithm is used, as shall be the case in spatial price equilibrium problems and in Walrasian price equilibrium problems.

Equilibration algorithms can also solve certain classical problems that possess quadratic programming formulations of the governing equilibrium conditions, such as a classical oligopoly problem. Moreover, these exact equilibration algorithms can be implemented on massively parallel architectures.




**Network structure of market equilibrium problem**

We now generalize the above algorithm to the case of $m$ supply markets and $n$ demand markets (see Dafermos and Nagurney (1989)).

The demand market exact equilibration algorithm would be used at each iteration. The algorithm below proceeds from demand market to demand market, at each iteration solving the "relaxed" single demand market problem exactly and in closed form. The assumptions, under which the algorithm converges, is that the supply price functions, demand price functions, and the transaction cost functions are linear and separable, and that the supply price functions are monotonically increasing, the demand price functions are monotonically decreasing, and the transaction cost functions are non-decreasing.

In this case, the equilibrium conditions take on the following expanded form: For each supply market $i$; $i = 1, \ldots, m$, and each demand market $l$; $l = 1, \ldots, n$,

$$\eta_i \sum_{j=1}^{n} x_{ij}^* + \psi_i + g_{il} x_{il}^*$$

$$+h_{il} \begin{cases} = -r_l \sum_{i=1}^{m} x_{il}^* + q_l, & \text{if} \quad x_{il}^* > 0 \\ \geq -r_l \sum_{i=1}^{m} x_{il}^* + q_l, & \text{if} \quad x_{il}^* = 0. \end{cases}$$

In the expression, the term $\eta_i \sum_{j=1}^{n} x_{ij}^* + \psi_i$ denotes the equilibrium supply price at supply market $i$, and $\eta_i, \psi_i$ are assumed to be positive. The term $g_{il} x_{il}^* + h_{il}$ denotes the equilibrium transaction cost, and, as previously, the term $-r_l \sum_{i=1}^{m} x_{il}^* + q_l$ denotes the equilibrium demand price at demand market $l$. The term $\sum_{j=1}^{n} x_{ij}^*$ is the equilibrium supply at supply market $i$, whereas the term $\sum_{i=1}^{m} x_{il}^*$ denotes the equilibrium demand at demand market $l$.

The equivalent optimization formulation of the equilibrium conditions is

$$\min_{x_{ij}\geq 0, \forall i,j} \sum_{i=1}^{m} (\frac{1}{2}\eta_i (\sum_{j=1}^{n} x_{ij})^2 + \psi_i \sum_{j=1}^{n} x_{ij})$$

$$+ \sum_{i=1}^{m} \sum_{j=1}^{n} (\frac{1}{2} g_{ij} x_{ij}^2 + h_{ij} x_{ij})$$

$$+ \sum_{j=1}^{n} (\frac{1}{2} r_j (\sum_{i=1}^{m} x_{ij})^2 - q_j \sum_{i=1}^{m} x_{ij}).$$

Under the above assumptions, the optimization problem is a strictly convex quadratic programming problem with a unique solution $x^*$.

**Demand Market Equilibration Algorithm**

**Step 0: Initialization**

Start with an arbitrary nonnegative shipment $x_{ij}^0$; $i = 1,\ldots,m; j = 1,\ldots,n$. Set $k := 1; l := 1$.

**Step k: Construction and Modification**

Construct a new feasible shipment $x_{il}^k; i = 1,\ldots,m$, by modifying $x_{il}^{k-1}$, in such a way so that the equilibrium conditions hold for this demand market $l$.

Set $l := l+1$.

**Convergence Verification**

If $l < n$, set $l := l+1$ and go to Step k; otherwise, verify convergence. If convergence to a prespecified tolerance has not been reached, set $k := k+1$, $l := 1$, and go to Step $k$.

Note that Step $k$, indeed, can be solved using the demand market exact equilibration algorithm presented above, with the $g_i$ and the $h_{il}$ terms updated accordingly to take into account the supply and transaction cost terms. Specifically, if we let $g_i \equiv \eta_i + g_{il}$, and

$$h_{il} \equiv \eta_i( \sum_{j \neq l, j < l} x_{ij}^k + \sum_{j \neq l, j > l} x_{ij}^{k-1}) + \psi_i + h_{il},$$

then the exact procedure can be immediately applied.

The above demand market equilibration algorithm proceeds from demand market to demand market in cyclic fashion, until the entire system is equilibrated. One may opt, instead, to select the subsequent demand market not in a cyclic manner, but in such a way that the objective function above is reduced more substantially.

To establish convergence of the demand market equilibration algorithm, note that in computing the solution to demand market $l$, one is simply finding a solution to the original objective function, but over a reduced feasible set. Hence, the value of the objective function is nonincreasing throughout this process. Moreover, the sequence generated by the algorithm contains convergent subsequences since all the generated flows remain in a bounded set of $R^{mn}$. Finally, if throughout a cycle of $n$ subsequent iterations, the value of the objective function remains constant, then the equilibrium conditions are satisfied for all supply markets and all demand markets. Consequently, it follows from standard optimization theory that every convergent subsequence of the sequence generated by the algorithm converges to a solution of the equilibrium conditions.

## 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."

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 $G = [N, A]$, where $N$ denotes the set of nodes, and $A$ 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:

$$f_a = \sum_p x_p \delta_{ap},$$

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 $a$.

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

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

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

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 $\lambda_w$ 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 Equilibration**

**Step 0: Initialization**

Construct an initial feasible flow pattern $x^0$ satisfying (2.75), 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.

**Step 2: Computation**

Compute

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

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

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 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:

$$\text{Minimize} \quad \sum_a \frac{1}{2} g_a f_a^2 + h_a f_a$$

subject to conservation of flow equations and the non-negativity 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.

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 load pattern. Set $k := 1$.

**Step 1: Equilibration**

Apply $E^1$.

**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).

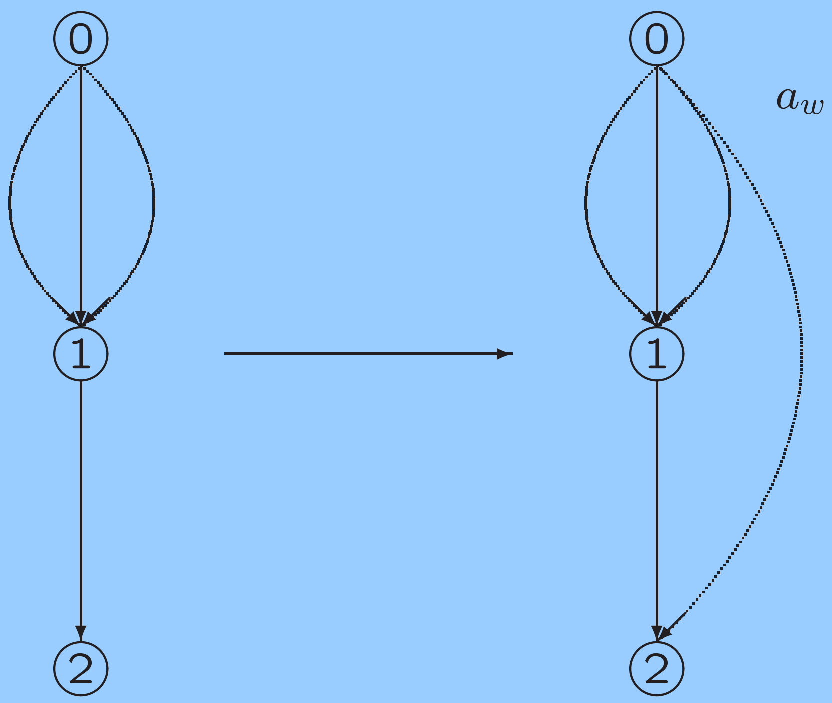
**Step 1: Equilibration**

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 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 $a_w$ connecting the O/D pair $w$. The user cost on such an arc is $c_{a_w} \equiv \lambda_w(\bar{d}_w - f_{a_w})$, where $f_{a_w}$ denotes the flow on arc $a_w$. The fixed demand for O/D pair $w$ then is set equal to $\bar{d}_w$.




**Fixed demand reformulation of elastic demand problem**

## 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

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

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.

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 $w$:

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

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

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

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

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

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

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

**Step 2: Computation**

Compute

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

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

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.

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

**References**


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

Amman, H. M., Belsley, D. A., and Pau, L. F., editors, **Computational Economics and Econometrics**, *Advanced Studies in Theoretical and Applied Econometrics* **22**, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992.

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.

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.

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.

Judd, K. L., **Numerical Methods in Economics**, MIT Press, Cambridge, Massachusetts, 1998.

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.

Murty, K. G., **Linear Complementarity Problems, Linear and Nonlinear Programming**, Heldermann, Berlin, Germany, 1988.

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.

Ortega, J. M., and Rheinboldt, W. C., **Iterative Solution of Nonlinear Equations in Several Variables**, Academic Press, New York, 1970.

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.

Perko, L., **Differential Equations and Dynamical Systems**, Springer-Verlag, New York, 1991.

Scarf, H. E. (with T. Hansen), **The Computation of Economic Equilibria**, Yale University Press, New Haven, Connecticut, 1973.

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

Thompson, G. L., and Thore, S., **Computational Economics**, The Scientific Press, San Francisco, California, 1992.

Todd, M. J., **The Computation of Fixed Points and Applications**, *Lecture Notes in Economics and Mathematical Systems* **124**, Springer - Verlag, Berlin, Germany, 1976.

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