Traffic Modeling and Variational Inequalities using GAMS *

Steven P. Dirkse† Michael C. Ferris‡

April 1997

Abstract

We describe how several traffic assignment and design problems can be formulated within the GAMS modeling language using newly developed modeling and interface tools. The fundamental problem is user equilibrium, where multiple drivers compete noncooperatively for the resources of the traffic network. A description of how these models can be written as complementarity problems, variational inequalities, mathematical programs with equilibrium constraints, or stochastic linear programs is given. At least one general purpose solution technique for each model format is briefly outlined. Some observations relating to particular model solutions are drawn.

1 Introduction

Models that postulate ways to assign traffic within a transportation network for a given demand have been used in planning and analysis for many years, see [41] and references therein. A popular technique for such assignment is to use the shortest path between the origin and destination points of a given journey. Of course, such a path depends not only on the physical distance between these two points, but also on the mode of transport and the congestion experienced during the trip.

*This material is based on research supported by National Science Foundation Grant CCR-9157632.

†GAMS Development Corporation, 1217 Potomac Street NW, Washington, D.C. 20007 (steve@gams.com).

‡Computer Sciences Department, University of Wisconsin – Madison, 1210 West Dayton Street, Madison, Wisconsin 53706 (ferris@cs.wisc.edu).

To account for congestion, traffic assignment models use the notion of user equilibrium or Wardropian equilibrium [42]. In this context, the travel time used to define the "distance" between the origin and destination is a function of the length and capacity of each arc of the path, and the total flow on that path. Thus, the fact that many users can travel along an arc will affect the time it takes any particular user to traverse the arc. User equilibrium occurs when all users travel along their shortest path, where distance is measured using the above definition for time. Underlying the model is the notion of noncooperative behavior - everyone is out for themselves. This should be contrasted with the notion of system equilibrium when a traffic controller assigns every vehicle to a particular path to minimize the total distance traveled. Note that these kinds of odels are typically used to predict the steady-state volume of traffic on a network, not to look at the dynamic behavior.

Due to the absence of an overall objective in the user equilibrium, this problem is more easily cast in the context of a variational inequality. In this paper, we describe how to formulate, solve, and extend models for traffic assignment using the notion of a mixed complementarity problem, a specialization of the variational inequality. Many papers have discussed the formulation and solution of user equilibrium problems using complementarity and variational inequality models, as well as the applications of these problems to urban planning; see [18, 19, 21, 30, 40].

The first two sections of the paper show how the user equilibrium problem is cast as a mixed complementarity problem (MCP), formulated within the GAMS modeling language, and solved using the PATH algorithm. Some examples of problems formulated using these tools are described in [6].

There are many cases when a modeler wishes to optimize an objective function subject to the system being in equilibrium. These problems are commonly called Mathematical Programs with Equilibrium Constraints (MPEC's). The recent monograph [29] describes the current state of optimality theory and algorithms related to such problems. In Section 4, we describe the basic structure of an MPEC and give some examples of traffic design problems that can be formulated as such.

Section 5 describes extensions to the GAMS modeling language that allow MPEC's to be formulated within the language. Furthermore, an outline of new tools for large scale implementation is given. These tools allow algorithm developers direct access to relevant function and derivative values via subroutine library calls. It is intended that this suite of routines, MPECLIB, will foster the development of new applications and test problems in the MPEC format. In the ensuing section, we show how these tools are used in an analysis of a tolling problem over a network representing Sioux-Falls. The problem is cast as an MPEC and two algorithms based on an implicit programming approach, namely DFO [5] and the bundle-trust region algorithm [39], are used to demonstrate the ability of these tools and the power of the modeling format.

The final section of the paper treats some modeling issues related to traffic assignment and path choice in networks subject to failure. Here again, we show how recently developed modeling tools are effective for investigating complex issues in transportation design and analysis.

2 MCP models: user equilibrium

The mixed complementarity problem (MCP) is defined in terms of some lower and upper bounds $\ell \in \mathbf{R}^n$ and $u \in \mathbf{R}^n$ satisfying $-\infty \leq \ell < u \leq +\infty$ and a nonlinear function $F: \mathbf{B} \to \mathbf{R}^n$. Here $\mathbf{B}$ represents the box $\mathbf{B} := [\ell, u] = \{z \in \mathbf{R}^n : \ell_i \leq z_i \leq u_i\}$. The variables $z \in \mathbf{R}^n$ solve $MCP(F, \ell, u)$ if for some variables $w \in \mathbf{R}^n$ and $v \in \mathbf{R}^n$

$$w_i \geq 0,\ v_i \geq 0,\ \ell_i \leq z_i \leq u_i, \quad i = 1, \ldots, n \tag{1}$$

$$F_i(z) = w_i - v_i, \quad i = 1, \ldots, n \tag{2}$$

and

$$w_i(z_i - \ell_i) = 0 \text{ and } v_i(u_i - z_i) = 0, \quad i = 1, \ldots, n. \tag{3}$$

Note that any solution z of MCP trivially satisfies the following implications

$$z_i = \ell_i \Rightarrow z_i \neq u_i \Rightarrow v_i = 0 \Rightarrow F_i(z) \geq 0 \tag{4}$$

$$z_i = u_i \Rightarrow z_i \neq \ell_i \Rightarrow w_i = 0 \Rightarrow F_i(z) \leq 0 \tag{5}$$

and

$$\ell_i < z_i < u_i \Rightarrow w_i = v_i = 0 \Rightarrow F_i(z) = 0. \tag{6}$$

Several interesting special cases of MCP exist for particular choices of ℓ and u. When $\ell \equiv -\infty$ and $u \equiv +\infty$, it follows from (6) that $w = v = 0$ and (1) is satisfied for any $z \in \mathbf{R}^n$. Hence, the problem becomes the classical square system of nonlinear equations

$$F(z) = 0. \tag{7}$$

Many techniques used to solve MCP are inspired by techniques used for such systems. Another special case is when $\ell \equiv 0$ and $u \equiv +\infty$, whereupon it can be easily seen that the problem (1)-(3) becomes

$$0 \leq z \perp F(z) \geq 0. \tag{8}$$

Here we have introduced the notation "$\perp$" that signifies the two adjacent quantities are orthogonal, that is, in addition to the explicit inequalities $0 \leq z$ and $F(z) \geq 0$ we have

$$z^T F(z) = 0.$$

In effect this enables us to rewrite (1) and (3) more succinctly as

$$0 \leq z - \ell \perp w \geq 0 \tag{9}$$

$$0 \leq u - z \perp v \geq 0. \tag{10}$$

Problem (8) is commonly called the nonlinear complementarity problem (NCP) and has been the subject of much research over the past three decades. A plethora of applications can be found in [14, 15]; this paper is concerned with applications arising from traffic and transportation management.

The general variational inequality VI(F, C) is defined using an arbitrary convex set $C \subseteq \mathbf{R}^n$ as the following system of inequalities

$$z \in C, \ \langle F(z), y - z \rangle \geq 0 \quad \forall y \in C. \tag{11}$$

It can be reformulated as an MCP using a transformation involving multipliers. We consider two cases separately. If the feasible set C is a box, then it is elementary to show that (11) and the MCP (1)-(3) defined by F and C are completely equivalent, as their solution sets are identical. When C is polyhedral rather than rectangular, (11) can be reduced to an MCP by explicitly including the dual variables to the constraints defining C. Thus, given a box $\mathbf{B}$ and a set $X := \{z \colon Az \leq b\}$, where $A \in \mathbf{R}^{m \times n}$, it can be shown that (11) with $C = \mathbf{B} \bigcap X$ is equivalent to VI$(H, \mathbf{B} \times \mathbf{R}^m_+)$, where

$$H(z, u) = \left[\begin{array}{c} F(z) + A^T u \\ -Az + b \end{array} \right].$$

When equality constraints are used to define X, the associated dual variables u are free. Two advantages to using the MCP formulation as opposed to the NCP are the explicit treatment of simple bounds on the variables z and the availability of free variables, which enable the explicit representation of equality constraints. This is more efficient than introducing extra variables and equations to deal with bounds and equality constraints.

We now proceed to show how to model the user equilibrium problem as an MCP. In all models used for the analysis of traffic congestion, there is a transportation network given by a set of nodes $\mathcal{N}$ and a set of arcs $\mathcal{A}$. In the equilibrium setting, it is usually assumed that drivers compete noncooperatively for the resources of the network in an attempt to minimize their costs, where the cost of traveling along a given arc $a \in \mathcal{A}$ is a nonlinear function $c_a(f)$ of the total flow vector f with components f_b, $b \in \mathcal{A}$. Let $c(f)$ denote the vector with components $c_a(f)$, $a \in \mathcal{A}$. There are two subsets of $\mathcal{N}$ that represent the set of origin nodes $\mathcal{O}$ and destination nodes $\mathcal{D}$ respectively. The set of origin-destination (O-D) pairs is a given subset $\mathcal{W}$ of $\mathcal{O} \times \mathcal{D}$; associated with each such pair is a travel demand that represents the required flow from the origin node to the destination node.

There are at least two equilibrium techniques used for generating models of traffic congestion on such a network. The first model is based on considering all the paths between the origin-destination pairs, and the second uses a multicommodity formulation, representing each origin or destination node as a different commodity. Both of these formulations use the Wardropian characterizations of equilibria [42], a special case of a Nash equilibrium (see [15, 23]).

2.1 A path based formulation

The given path based formulation follows [20]. For each $w \in \mathcal{W}$, let $\mathcal{P}_w$ represent the set of paths connecting the O-D pair w and $\mathcal{P}$ represent the set of all paths joining all O-D pairs of the network. Let ξ_p denote the flow on path $p \in \mathcal{P}$; let $\gamma_p(\xi)$ be the cost of flow on this path which is a function of the path flow vector ξ. Let Δ be the arc-path incidence matrix with entries

$$\delta_{ap} \equiv \begin{cases} 1 & \text{if path } p \in \mathcal{P} \text{ traverses arc } a \in \mathcal{A} \\ 0 & \text{otherwise.} \end{cases}$$

It is clear that f and ξ are related by

$$f = \Delta\, \xi.$$

When the path cost $\gamma_p(\xi)$ on each path p is assumed to be the sum of the arc costs on all the arcs traversed by p, that is, if

$$\gamma(\xi) \equiv \Delta^T c(f),$$

the model is called *additive*. Finally, we introduce variables τ_w that depict the minimum transportation cost (or time) between O-D pair $w \in \mathcal{W}$. The travel demand between O-D pair w is assumed to be a function $d_w(\tau)$ of the vector τ in the path formulation. The model is called a *fixed-demand model* if each $d_w(\tau)$ is a constant function; the general model is often called the *elastic demand model.*

The Wardrop equilibrium principle [42] states that each driver will choose the minimum cost path between every origin destination pair and through this process, the paths used will all have equal cost; paths with costs higher than the minimum will have no flow. Mathematically, this principle can be phrased succinctly as

$$0 \leq \gamma_p(\xi) - \tau_w \;\perp\; \xi_p \geq 0, \;\; \forall w \in \mathcal{W}, p \in \mathcal{P}_w. \tag{12}$$

The demand is satisfied if

$$\sum_{p \in \mathcal{P}_w} \xi_p \geq d_w(\tau), \;\; \forall w \in \mathcal{W},$$

and the equilibrium condition of zero excess demand can be stated as follows,

$$0 \leq \sum_{p \in \mathcal{P}_w} \xi_p - d_w(\tau) \;\perp\; \tau_w \geq 0, \;\; \forall w \in \mathcal{W}. \tag{13}$$

Conditions (12) and (13) clearly define a nonlinear complementarity problem with (ξ, τ) as the variables.

For networks of reasonable size with many O-D pairs, the enumeration of all paths connecting elements of $\mathcal{W}$ is prohibitive. Thus, the above path-flow formulation is not suitable for a generic complementarity code. Nevertheless, there are path-generation schemes [24, 33] that utilize this formulation and generate the paths only if they are needed. The alternative multicommodity formulation to be discussed below completely removes the necessity of enumerating the paths.

2.2 A multicommodity formulation

In this alternative formulation of the traffic equilibrium problem, a commodity is associated with each destination node. For simplicity, we assume that each node in $\mathcal{D}$ is a destination. Let K be the cardinality of $\mathcal{D}$. The variable $x = (x^1, x^2, \ldots, x^K)$ represents the flows of the commodities $1, 2, \ldots, K$ with x_{ij}^k denoting the flow of commodity k on arc $(i,j) \in \mathcal{A}$. The variable $t = (t^1, t^2, \ldots, t^K)$ is composed of components t_i^k that represent the minimum cost (or time) to deliver commodity k (i.e. to reach destination k) from node i. Associated with each pair (k, i), $k \in \mathcal{D}$ and $i \in \mathcal{N}$, is the travel demand d_i^k, which is a function of the minimum cost vector t. There are two sets of equilibrium conditions. The first represents conservation of flow of commodity k at node i and is given by

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{j:(j,i)\in\mathcal{A}} x_{ij}^k = d_i^k, \qquad \forall i \in \mathcal{N}, k \in \mathcal{D}.$$

In terms of the standard node-arc incidence matrix $\mathcal{I}$ of the network and the demand function $d^k(t)$, these constraints can be rewritten as

$$\mathcal{I}x^k = d^k(t), \qquad \forall k \in \mathcal{D}. \tag{14}$$

The second condition ensures that if there is positive flow of commodity k along arc (i,j), then the corresponding time to deliver that commodity is minimized:

$$0 \le c_{ij}(f) + t_j^k - t_i^k \ \perp\ x_{ij}^k \ge 0 \qquad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{D}, \tag{15}$$

where the arc flow vector f is given by

$$f \equiv \sum_{k\in\mathcal{D}} x^k.$$

This is typically termed "Wardrop's second principle", although it appears first in his article [42]. It is clear that given an enumeration of the paths, the solution generated from a path based formulation can easily yield a solution to the multicommodity formulation, and vice versa.

Eliminating the flow vector f, conditions (14) and (15) define an MCP in the arc flow vector x and minimum travel cost vector t. Early studies of the traffic equilibrium problem have focused on the case of a separable cost function and constant demand function; that is, for all $a \in \mathcal{A}$ and $k \in \mathcal{D}$,

$$c_a(f) = c_a(f_a), \text{ and } d^k(t) = d^k \text{ (a constant)}.$$

A specific example of a separable cost function c_a is given by

$$c_a(f_a) = A_a + B_a \left[\frac{f_a}{\gamma_a}\right]^4 \qquad \forall a \in \mathcal{A},$$

for particular data A_a, B_a and γ_a. Cost functions of this kind have been used extensively in transportation research. Nonseparable and nonintegrable cost functions $c(f)$ have also been used in the literature; see for example [21].

It is shown in [13] how to implement the multicommodity formulation as an MCP within the GAMS modeling language. For more discussion of the complementarity approach to traffic equilibrium, see [22].

3 MCP algorithm: PATH

There has been a great deal of research into algorithms for solving MCP and NCP. Some of these recent developments are surveyed in [12]; much of this work has investigated nonsmooth analysis and algorithms based on the systems of equations

$$\min\{z_i,\ F_i(z)\} = 0 \quad i = 1, \ldots, n \tag{16}$$

and

$$z_i + F_i(z) - \sqrt{z_i^2 + F_i^2(z)} = 0,\ i = 1, \ldots, n. \tag{17}$$

However, we will outline here only the basic ideas behind the PATH algorithm [7, 8] for MCP since this is currently the most widely used code for solving such problems due to the fact that it is available as a GAMS subsystem. This code is intended for large scale applications and uses sparse matrix technology in its implementation. Since our aim is to look at traffic applications which we formulate as nonlinear complementarity problems, we will describe the algorithm only in this context.

Many of the ideas relating to complementarity theory can be thought of as simple generalizations of equation theory. To understand this, we consider the notion of a "normal cone" to a given convex set C at some point $z \in C$, a generalization of the notion of a normal to a smooth surface. This set consists of all vectors c which make an obtuse angle with every feasible direction in C emanating from z, that is

$$N_C(x) := \{c\colon \langle c, y - z\rangle \leq 0, \text{for all } y \in C\}.$$

When $C = \mathbf{R}^n_+$, the nonnegative orthant in $\mathbf{R}^n$, we label this set as $N_+(z)$. A little thought enables one to see that the nonlinear complementarity problem is just the set-valued inclusion,

$$0 \in F(z) + N_+(z),\ z \in \mathbf{R}^n_+.$$

It is possible to look at NCP from this geometric standpoint. Firstly, it is well-known that $N_+(x_+)$ is characterized by the rays $x - x_+$, where $(x_i)_+ := \max\{x_i, 0\}$ is the projection of x onto the nonnegative orthant. An equivalent formulation of (8) is therefore to find a zero of the nonsmooth equation

$$0 = F(x_+) + x - x_+$$

The map $F_+ := F(x_+) + x - x_+$ is sometimes referred to as the "normal map", see for example [11, 37, 38]. The equivalence is established by noting that if z solves NCP, then $x = z - F(z)$ is a zero of the normal map, and if x is a zero of the normal map, then $z = x_+$ is a solution of the NCP.

Under the assumption that F is smooth, it is easy to see that the normal map is a smooth map on each of the orthants of $\mathbf{R}^n$ and is continuous on $\mathbf{R}^n$. However, the normal map is not in general differentiable everywhere. It is an example of a piecewise smooth map and is intimately related to a manifold defined by the collection of faces of the set $\mathbf{R}^n_+$, called the normal manifold. It is well known that the collection of these faces precisely determine the pieces of smoothness of the normal map. For example, it was shown in [38] that these pieces are the (full dimensional) polyhedral sets $\mathcal{F} + N_{\mathcal{F}}$ that are indexed by the faces $\mathcal{F}$ of C; here $N_{\mathcal{F}}$ represents the normal cone to the face $\mathcal{F}$ at any point in its relative interior. When the set is $\mathbf{R}^n_+$, the faces are given by $\{x : x \geq 0, x_I = 0\}$, where I runs over the subsets of $\{1, 2, \ldots, n\}$; the pieces of the manifold in this case are precisely the orthants of $\mathbf{R}^n$.

In the context of nonlinear equations, Newton's method proceeds by linearizing the smooth function F. Since F_+ is nondifferentiable, the standard version of Newton's method for NCP approximates F_+ at $x^k \in \mathbf{R}^n$ with the piecewise affine map

$$L_k(x) := F(x^k{}_+) + \nabla F(x^k{}_+)(x_+ - x^k{}_+) + x - x_+.$$

Thus, the piecewise smooth map F_+ has been approximated by a piecewise affine map. The Newton point x^k_N (a zero of the approximation L_k) is found by generating a path p^k, parametrized by a variable t which starts at 0 and increases to 1 with the properties that $p^k(0) = x^k$ and $p^k(1) = x^k_N$. The values of $p^k(t)$ at intermediate points in the path satisfy

$$L_k(p^k(t)) = (1 - t)F_+(x^k).$$

The path is known to exist locally under fairly standard assumptions and can be generated using standard pivotal techniques to move from one piece of the piecewise affine map to another. Further details can be found in [36].

A Newton method for NCP would accept x^k_N as a new approximation to the solution and re-linearize at this point. However, as is well known even in the literature on smooth systems, this process is unlikely to converge for starting points that are not close to a solution. In a damped Newton method for smooth systems of nonlinear equations (7), the merit function $F(x)^T F(x)$ is typically used to restrict the step size and enlarge the domain of convergence. In the PATH algorithm[7], the piecewise linear path p^k is computed and searched using a non-monotone watchdog path-search of the merit function $\|F_+(x)\|^2$. The watchdog technique allows path-searches to be performed infrequently, while the non-monotone technique allows the Newton point to be accepted more frequently. The combination of these techniques helps avoid convergence to minimizers of the merit function that are not zeros of F_+, without detracting from the local convergence rates [7].

The extension of the above approach to MCP is straightforward; furthermore, computational enhancements are described in [9].

4 MPEC models: tolling and inverse problems

An MPEC consists of two types of variables, namely design variables $x \in \mathbf{R}^n$ and state variables $y \in \mathbf{R}^m$. A function $\theta: \mathbf{R}^{n+m} \to \mathbf{R}$ is the overall objective function to be minimized, subject to two sets of constraints. The first set $(x, y) \in Z$ represents joint feasibility constraints for x and y, with $Z \subseteq \mathbf{R}^{n+m}$ representing a nonempty closed set. The second set of constraints are the equilibrium constraints, defined by the equilibrium function $F: \mathbf{R}^{n+m} \to \mathbf{R}^m$ and the closed convex set $C \subseteq \mathbf{R}^m$. These constraints force the state variables y to solve a VI parametrically defined by $F(x, \cdot)$ and C. The MPEC can be writtens succinctly in the following manner.

$$\begin{array}{ll} \text{minimize} & \theta(x,y) \\ \text{subject to} & (x,y) \in Z \\ \text{and} & y \text{ solves } \mathrm{VI}(F(x,\cdot),C). \end{array} \tag{18}$$

A special case of the MPEC is the bilevel program in which the mapping $F(x, \cdot)$ is the partial gradient map (with respect to the second argument) of a real-valued C^1 function. In the next two sections, we describe some traffic models that can be formulated as MPEC's and show how to carry out this modeling within GAMS.

One example of an MPEC occurring in traffic network design is described in [31]. The idea is to determine values for some design variables, e.g. arc capacities, that minimize a weighted some of the investment cost and the system operating costs. Another example is described in [4]. This is an example of an inverse problem, where estimates of O-D demands are given and the network planner wishes to adjust these demands minimally in order to satisfy the equilibrium conditions.

A third example is present in tolling. The tolling model is based on an underlying assumption of user equilibrium. As we outlined in Section 2, user equilibrium assumes that each driver uses his shortest path, where distance is measured using a function for time of journey based on the total flows on the arcs.

When a particular arc is tolled, this increases the monetary cost of traversing that arc. We use a simple weighting to add this cost to the distance each user tries to minimize. Of course, complex human behavioral models could be used to generate realistic weightings and potentially add nonlinear cost effects. Our model currently ignores these facets of the problem and simply adds the toll p_a to the cost function c_a for each arc. If we view these tolls as parameters, we have not changed the structure of the user equilibrium problem at all. If we view the tolls as design variables, however, we now have quite a bit of flexibility in designing the tolling structure, and can do so with certain objectives in mind.

What is the objective of adding tolls to some of the arcs of the network? In some cases, it is to maximize system revenue, in which case the upper level objective function for the equilibrium problem defined by (14) and (15) is as follows:

$$\theta(p, f, t) = \sum_{a \in \mathcal{A}} p_a f_a \tag{19}$$

Note that the design variables in this problem are p_a (which are typically nonzero for a small subset of $\mathcal{A}$, and the state variables are f and t. Note that if tolls p_a are set too high, then drivers will use other arcs creating a decrease in f_a and possibly forcing total revenue to decrease.

In other cases, a traffic controller is attempting to impose tolls with the aim of reducing congestion. In these cases, a typical objective function is system cost

$$\theta(p, f, t) = \sum_{a \in \mathcal{A}} c_a(f_a)$$

Other objectives arise in different applications.

The GAMS model that we developed for testing MPECLIB and the solvers that we implemented arose from such a tolling problem. The model used in our example was based on the objective shown in (19). The GAMS source of the model is available via anonymous ftp from

`ftp://www.cs.wisc.edu/math-prog/mcplib/traffic/`.

Apart from the particular data and model equations used, the formulation in GAMS follows the structure we now outline for a simple model using the newly developed MPECLIB.

5 Interfacing models and algorithms: MPECLIB

We have developed a software interface between the GAMS modeling language and MPEC algorithms that allows users to model practical, large-scale

MPEC's and algorithm developers to link in their solvers for such problems. We believe such an interface serves two purposes. Firstly, the data from realistic applications is most easily made accessible to researchers developing codes for these problems via interfaces similar to ours. This is an essential ingredient in algorithmic development, testing, and comparison. Secondly, it is only when efficient codes can be applied to real applications, such as tolling problems, and can show an improvement over the existing heuristic schemes that the modeling format of an MPEC becomes a serious alternative to such heuristics.

Unfortunately, developing an interface that easily allows both algorithm developers and application experts to perform their respective research is difficult. We have chosen to allow modelers to use the GAMS [3] modeling language and force algorithmic development to occur in Fortran or C, although a possible extension of this work to support algorithm development in Matlab [17] is possible. Comparable tools that enable algorithmic development for MCP have previously been described in [10], with the result that many more complementarity applications are now being developed.

From a modeling standpoint, the MPEC interface tool is very similar to the MCP interface in GAMS. This allows the application expert to move from an MCP to an MPEC formulation with very little difficulty. The modeling interface to MPEC is similar to the MCP one, consisting of the usual GAMS language features (e.g. sets, parameters, variables, equations, control structures) and extensions to the GAMS `model` and `solve` statements. These extensions are required to allow the modeler to specify the complementarity conditions (i.e. the equilibrium constraints) along with an objective variable and its defining equation. Just as with MCP's, the MPEC `model` statement includes a collection of equation-variable pairs, where each pair defines a complementarity relationship between the function determined by the equation and the variable in the pair. In case equations and variables are indexed by sets, functions and variables with matching indices are paired. In addition, the MPEC `model` may include an unpaired "objective" equation. While MCP must have an equal number of functions and variables, this is not the case with MPEC models. Each function in an MPEC must have a matching variable, but unmatched variables are now allowed; these are the design variables x in (18), while the "matched" variables are the state variables y.

The objective variable and the direction of optimization are specified in the GAMS `solve` statement, using the `MPEC` keyword as the model type. In order to define the objective, a scalar "objective" variable and an equation defining this variable are often used. In this case, the objective variable must appear only in the objective equation, and must appear linearly in it. As an example, consider the simple MPEC below (a reformulation of [32], Example 1):

$$\begin{array}{ll} \text{minimize} & \theta(x,y,u) := {x_1}^2 - 2x_1 + {x_2}^2 - 2x_2 + {y_1}^2 + {y_2}^2 \\ \text{subject to} & x_i \in [0,2] \\ \text{and} & (y,u) \text{ solves } \mathrm{MCP}(F(x,\cdot,\cdot),\mathbf{B}), \end{array} \tag{20}$$

where

$$F(x,y,u) := \left[\begin{array}{c} -2x_1 + 2y_1 + 2(y_1-1)u_1 \\ -2x_2 + 2y_2 + 2(y_2-1)u_2 \\ -(y_1-1)^2 + .25 \\ -(y_2-1)^2 + .25 \end{array} \right]$$

and $\mathbf{B} := \{(y,u) \in \mathbf{R}^4 \colon u \geq 0\}$. The GAMS model for this example is given in Figure 1. Readers familiar with GAMS will understand the sets, variables, and equations declared in Figure 1 immediately, while those not familiar with GAMS will appreciate the concise yet descriptive style and should recognize the parallel to (20). In the `model` statement, we see the "objective" equation given first, while the remaining pairs define the equilibrium constraints. The variables `y` and `u` are paired with equations, and are state variables. The variable `x` is not paired, so it is a design variable.

While an indexed GAMS variable will often have components of only one type (i.e. design or state variables) this is not always the case. For example, it is possible to pair a variable `w(I)` declared and defined over the set `I` with an equation `g(I)` declared over the same set `I` but defined over a subset `II` of `I`. In this case, the components of `w` with indices in `II` will be state variables matched to `g`, while the components of `w` with indices in `I\ II` will be design variables.

When the GAMS `solve` statement is executed, the equation-variable pairs defining the MPEC model, together with information about the sets, variables, and equations themselves, are sent to the disk as a sequence of scratch files, and an MPEC solver is called. This solver uses the interface library MPECLIB to read and interpret the scratch files, evaluate functions and gradients, and write solution data.

```
set I / 1 * 2 /;
alias (I,J);

variables
obj,
x(J)          'design variables',
y(I)          'state variables',
u(I)          'state vars, duals in MCP reformulation of VI';
x.lo(J) = 0;
x.up(J) = 2;
u.lo(I) = 0;

equations
objeq,
Fy(I),
Fu(I);

objeq ..  obj =e= sum(J, sqr(x(J))) - 2 * sum (J, x(J))
                  + sum(I, sqr(y(I)));

Fy(I) ..  (-2)*x(I) + 2*y(I) + 2*(y(I)-1)*u(I) =e= 0;

Fu(I) ..  (.25) =g= sqr(y(I)-1);


model oz1 / objeq, Fy.y, Fu.u /;

option mpec=bundle;
solve oz1 using mpec minimizing obj;
```

Figure 1: GAMS Model for (20)

Interface Initialization

```
int mpecInit (char *controlFileName, int indexStart,
              int diagRequired, int noObj,
              mpecRec **mpec);
void sparseInit (mpecRec *mpec, int colPtrx[], int cpxdim,
                 int rowIdxx[], int rixdim,
                 int colPtry[], int cpydim,
                 int rowIdxy[], int riydim);
```

The first task of the solver is to call the mpecInit routine to read in the scratch files and construct a problem of the form

$$\begin{array}{ll} \text{minimize} & \theta(x,y) \\ \text{subject to} & x \in \mathbf{B}_x \\ \text{and} & y \text{ solves } \mathrm{MCP}(F(x,y),\mathbf{B}_y). \end{array} \tag{21}$$

Note that the form of (21) is not as general as that specified in (18). In (21) we force the modeler to use $Z = \mathbf{B}_x \times \mathbf{R}^m$, although in practice there may be models that include side constraints of the form $h(x) = 0$ or $g(x,y) = 0$. Furthermore, we assume the equilibrium problem is written as an MCP. The motivation for this restriction is simply that there are currently no large scale implementations that allow for such side constraints. Such extensions to the model format would be easy to implement whenever appropriate solvers for these problems reach maturity.

If there is any inconsistency in the MPEC specification, it is detected during this initialization phase. It is here that the variables are partitioned into design and state variables, and the required maps are set up to support efficient function and gradient evaluation by other routines. Also, parameters to mpecInit exist allowing the user to specify if index ranges must begin with 0 (C style) or 1 (Fortran style), whether or not a dense Jacobian diagonal is required, and whether the model is an MCP instead of an MPEC (the library can be used for both model types). A pointer to a record containing all information specific to this model is passed back to the calling routine. This pointer will be passed on all subsequent MPECLIB calls.

In order to fully initialize the MPEC model, some space is required for the row indices and column pointers used to store the sparse Jacobians. Rather than allocating this space inside the library, the mpecInit routine returns estimates of the amount of space required for this to the calling routine. A second routine, `sparseInit`, must then be called, which passes in these arrays, as well as the number of elements actually allocated for them. This routine completes the initialization, using the space provided it. The assumption here is that the user will not modify these arrays, as they are used by both solver and interface library. This assumption saves having to store two copies of the sparsity structure and copy it to the user's data structure at each derivative evaluation.

Variable Bounds and Level Values

```
void getxBounds (mpecRec *mpec, double lb[], double ub[]);
void getxLevels (mpecRec *mpec, double x[]);
void setxbar (mpecRec *mpec, double xbar[]);
void getyBounds (mpecRec *mpec, double lb[], double ub[]);
void getyLevels (mpecRec *mpec, double y[]);
```

The routines to obtain variable bounds and initial level values are for the most part self-explanatory. The `setxbar` routine is used to store a vector of design variables $\bar{x}$ in MPECLIB for use in subsequent calls to function and gradient routines that pass only state variables y. This is useful for solvers that implement a two-level solution scheme in which the inner solver (an MCP code) has no knowledge of the variables x in the outer optimization problem. In our current implementation, the box $\mathbf{B}_y$ does not depend on y, so the `getyBounds` routine would be called only once. A possible generalization is to allow $\mathbf{B}_y$ to depend on x, in which case a new function with the input parameter x would be required. This function would of course be called whenever x changed.

Function and Jacobian Evaluation

```
int getF ( mpecRec *mpec, double x[], double y[],
           double F[], double *theta);
int getdF (mpecRec *mpec, double x[], double y[],
           double F[], double *theta,
           double Jx[], int colPtrx[], int rowIdxx[],
           double Jy[], int colPtry[], int rowIdxy[],
           double dthetadx[], double dthetady[]);
int getFbar ( mpecRec *mpec, int n, double y[], double F[]);
int getdFbar (mpecRec *mpec, int n, int nnz, double y[],
              double F[],
              double Jy[], int colPtry[], int rowIdxy[]);
```

The routine getF takes the current point (x, y) as input and outputs the value of the functions F and θ at this point. The routine getdF does this also, but it computes the derivative of F and θ as well. The derivative of F w.r.t. x and y are returned in separate matrices, both stored sparsely in row index, column pointer fashion, while the derivative of θ w.r.t. x and y is returned as two dense vectors. The routines getFbar and getdFbar are similar, but in these routines, the input x is assumed to be the constant value $\bar{x}$ fixed in the previous call to setxbar. In this case, derivatives w.r.t. x and objective function values and derivatives are also not passed back. These routines are designed for use by an algorithm solving an inner (MCP) problem.

Solver Termination

```
void putxLevels (mpecRec *mpec, double x[]);
void putyLevels (mpecRec *mpec, double y[]);
void putObjVal (mpecRec *mpec, double theta);
void putStatus (mpecRec *mpec, int modelStat, int solverStat);
int mpecClose (mpecRec *mpec);
```

Once a solution has been found, the solver must pass this solution on to the interface library. This is done via the putxLevels, putyLevels, and putObjVal routines. The putStatus routine is used to report the model status (e.g. local optimum found, infeasible, unbounded, intermediate nonoptimal) and solver status (e.g. normal, iteration limit, out of memory, panic termination) via integer codes. All of this information is stored in the mpec data structure and written to disk when the mpecClose routine is called. When the solver terminates, these files are read by GAMS so that the solution information is available for reporting purposes, as data to formulate other models, etc.

6 MPEC algorithms: implicit approaches

Algorithms for solving MPEC's are not nearly as well developed as those for MCP. Given the efficiency of the known methods for solving MCP and NLP, we describe here several techniques for solving MPEC's using an implicit approach that solves a sequence of MCP's.

We assume first that the problem has the form (21). For the implicit programming approach to work, a further assumption is needed, namely that there is a (locally) unique solution y of the equilibrium problem $\mathrm{MCP}(F(x,\cdot), C)$

for each value of x. We denote this solution by $y(x)$. Under these assumptions, the problem (21) is equivalent to the implicit program:

$$\begin{array}{ll} \text{minimize} & \Theta(x) = \theta(x, y(x)) \\ \text{subject to} & x \in \mathbf{B}_x. \end{array} \tag{22}$$

This implicit programming formulation has the advantage of simple constraints, but a rather complex, in fact nondifferentiable objective function Θ, even though the original functions f and F may be smooth. Nonsmoothness results from the underlying complementarity condition.

A promising solution strategy for the implicit program (22) is to apply a "bundle method", that is an algorithm specifically designed to solve nonsmooth optimization problems. This idea is presented in [27, 28, 32]. The implementation of the bundle method we used, `btnc` [39], was developed for bound constrained problems and requires the user to provide a Fortran implementation to evaluate the objective function and a subgradient at a user supplied point. The formulas to generate these quantities were developed in [32] for the MPEC format described in (21).

The Fortran function was easy to code using the routines developed in Section 5; essentially the only modification needed to the PATH solver was an option to allow the optimal basis of an MCP to be returned to the caller.

Derivative free optimization has a long history in mathematical programming. We used MPECLIB in conjunction with a prototype Matlab implementation of DFO [5] provided by Ph. Toint. This implementation was designed under the assumption that the dimension of the underlying optimization problem is small and the time to evaluate the objective function is large. Even though the underlying MCP is large dimensional, provided the number of tolled arcs is small, these properties are present in the implicit form of tolling model (22). Based on these assumptions, the DFO algorithm uses multivariate interpolation to develop good local models and ideas from the trust region literature to search the (small dimensional) space.

While both of these MPEC solvers succeeded in solving a large class of tolling problems of the form outlined in Section 4, their performance was not entirely satisfactory. Even though the number of variables in the underlying MCP can be large, preliminary numerical results demonstrate the fact that the current implementations of both these codes are limited to small numbers of design variables. We believe that future research and improvements in these and other algorithms will lead to substantially more robust and efficient implementations. The use of MPECLIB will remain critical in each such development.

7 Stochastic models in GAMS

In most practical instances, many of the variables in the previous formulations are not known with certainty, but are estimated from observed data, perhaps using an MPEC formulation of an inverse problem. Even when we are willing to believe these estimations as being truly representative of the actual data in the problem, the network may be subject to dynamic changes that mean its behavior over time varies considerably. Thus, when failures occur in a network (due for example to accidents or roadworks), the shortest paths that each user views in the network changes, and each driver tries to compensate for these changes by modifying their path choice.

In this stochastic setting, a typical formulation of the user equilibrium problem assumes that all users have complete information regarding the plans that every other user has under all possible scenarios or states of the network. We believe this to be an unreasonable assumption. Instead, we review below some recent work [16] on how a single user can modify their shortest path in order to develop a plan that is more robust to failures in the network. We show how to use some extensions of the GAMS modeling language to implement these models, and briefly describe the results of this work.

The model assumes that the network may be in one of finitely many states characterized by different travel times along the arcs, and allows transitions between the states according to a continuous-time Markov chain. The objective is to guide the vehicles in a manner minimizing the total expected travel time.

We describe the case when the only possible transitions are between state 0 (representing the *normal* operation mode) and states $\ell \neq 0$ (representing *failure* modes). The rate of transition from 0 to $\ell \neq 0$ will be denoted by λ^ℓ, and the transition rate back by μ^ℓ, see Figure 2. The general case is treated in [16].

The problem is as follows. At each node $n \in \mathcal{N}$ there is a constant demand flow s_n that must be moved through the network to some destination node D at the minimum expected travel time. To facilitate the analysis and to provide ground for more general cases we make the following simplifying assumptions.

(A1) If the state of the system changes from k to ℓ when a vehicle is on arc (i,j) the travel time on (i,j) remains equal to c_{ij}^k for this vehicle; it experiences new travel times only after hitting j.

(A2) The products $\lambda^\ell c_{ij}^0$ and $\mu^\ell c_{ij}^\ell$ are much smaller than one for each ℓ

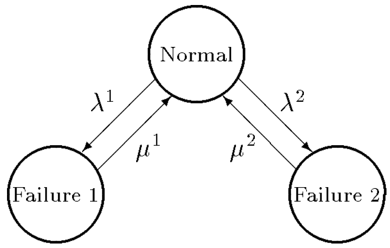

Figure 2: Transition diagram of network's states.

and each $(i,j) \in \mathcal{A}$.

Condition (A1) amounts to assuming that failures occur at the initial sections of the arcs and do not affect those who have passed them. It is equally simple to consider other cases, except the notation then becomes more involved.

Condition (A2) implies that the probability of more than one state transition during the travel time of a vehicle on an arc (i,j) is negligibly small.

Whenever the arcs are uncapacitated, the problem can be solved as a stochastic shortest path problem [1, 2, 34]. We analyze the capacitated problem, in which the main issue is the interaction between vehicles that started at different times but reach a node at the same time, thus leading to jams. In this general case there are arc capacities u_{ij}^{ℓ}, associated with states $\ell = 0, \dots, L$, so we cannot ignore the interactions between different flow subvectors, if they share the same arc at the same time.

We assume that all travel times are integer and let M be an upper bound on all of them. Suppose that a transition from state 0 to state ℓ takes place, and let $t = 0$ denote the time of this transition. Let $Y_{ij}^{\ell}(t)$ be the flow of re-routed vehicles entering arc (i,j) at time t. They satisfy the flow conservation equations

$$\sum_{(i,j)\in\mathcal{A}} Y_{ij}^{\ell}(t) - \sum_{\substack{(j,i)\in\mathcal{A} \\ c_{ji}^{\ell}\le t}} Y_{ji}^{\ell}(t - c_{ji}^{\ell}) = \sigma_i(t), \quad i \in \mathcal{N} \setminus D, \quad t = 0, 1, 2, \dots, \tag{23}$$

where $\sigma_i(t)$ is the inflow into i of the vehicles that experienced the state transition while traveling along the arcs leading to i:

$$\sigma_i(t) = \sum_{\substack{(j,i)\in\mathcal{A} \\ c^0_{ji} > t}} X_{ji}. \tag{24}$$

Since the supply (24) vanishes after a finite time (for which an upper bound M is known), we know that the flows Y^ℓ will vanish after a finite time, too, although this time may be much larger than M.

Since we have many sources, and the network is not layered, we cannot ignore the interactions of the rescheduled flow Y^ℓ with the flow $X^\ell(t)$ of vehicles that started *after* the state transition to ℓ. We make a simplifying assumption that further state transitions do not occur during the time that we are calculating X^ℓ. Even with this assumption, we cannot avoid modeling the initial non-stationary phase, when the re-routed flow $Y^\ell(t)$ and the new flow $X^\ell(t)$ interact. The policy that we develop under this assumption is termed a one-step lookahead policy.

Denoting by T the optimization horizon and by $Z^\ell(t) = Y^\ell(t) + X^\ell(t)$ the effective flow after the state transition, we obtain the problem

$$\min \sum_{t=0}^{T} \sum_{(i,j)\in\mathcal{A}} c^\ell_{ij} Z^\ell_{ij}(t) \tag{25}$$

$$\sum_{(i,j)\in\mathcal{A}} Z^\ell_{ij}(t) - \sum_{\substack{(j,i)\in\mathcal{A} \\ c^\ell_{ji} \le t}} Z^\ell_{ji}(t - c^\ell_{ji}) = s_i + \sigma_i(t), \quad i \in \mathcal{N} \setminus D, \quad t = 0, 1, \ldots, T, \tag{26}$$

$$0 \le Z^\ell_{ij}(t) \le u^\ell_{ij}, \quad (i,j) \in \mathcal{A}, \quad t = 0, 1, \ldots, T, \tag{27}$$

where the additional supply $\sigma_i(t)$ is given by (24). The optimal value $Q^\ell(X)$ of the above problem is the rescheduling cost for the plan X, when transition to state ℓ occurs.

There are reasons to believe that the value T does not matter for determining the robust plan X, provided T is large enough, and that the solution to (25)–(27) becomes, for large t, equal to a solution of the 'steady-state' problem associated with state ℓ:

$$\min \sum_{(i,j)\in\mathcal{A}} c^\ell_{ij} \overline{X}^\ell_{ij}$$

$$\sum_{(i,j)\in\mathcal{A}} \overline{X}^\ell_{ij} - \sum_{(j,i)\in\mathcal{A}} \overline{X}^\ell_{ji} = s_i, \quad i \in \mathcal{N} \setminus D,$$

$$0 \le \overline{X}^{\ell}_{ij} \le u^{\ell}_{ij}, \quad (i,j) \in \mathcal{A}.$$

To avoid some terminal effects associated with the fact that the vehicles that start late cannot make it to the destination anyway, and therefore choose short arcs, we may augment (25)–(27) with terminal conditions:

$$Z^{\ell}_{ij}(t) = \overline{X}^{\ell}_{ij}, \quad t = T - \tau, T - \tau + 1, \ldots, T-1, T, \quad (i,j) \in \mathcal{A}, \tag{28}$$

where τ is some constant (for example, the maximum travel time on the arcs). In fact, by choosing T (or τ) one may change the allowed length of the transient period, before the flow settles on the new steady-state solution. This is easily done from within GAMS.

We are now ready to formulate the robust planning problem in the capacitated case:

$$\min \Big\{ \sum_{(i,j)\in\mathcal{A}} c^0_{ij} X_{ij} + \sum_{\ell=1}^{L} \lambda^{\ell} Q^{\ell}(x) \Big\} \tag{29}$$

$$\sum_{(i,j)\in\mathcal{A}} X_{ij} - \sum_{(j,i)\in\mathcal{A}} X_{ji} = s_i, \quad i \in \mathcal{N} \setminus D, \tag{30}$$

$$0 \le X_{ij} \le u^0_{ij}, \quad (i,j) \in \mathcal{A}. \tag{31}$$

The functions $Q^{\ell}(x)$ are the optimal values of the re-routing problems in scenarios $\ell = 1, \ldots, L$.

Problem (29)–(31) is similar to two-stage stochastic programming problems (see [25, 35, 43]). Much is known about these problems, and efficient solution techniques exist that exploit the structure of the model in question (see [26, 43] and the references therein). The simplest approach, however, is to include the linear programs defining $Q^{\ell}(X)$ into (29)–(31) and construct a giant linear programming problem with a dual block angular structure:

$$\min \sum_{(i,j)\in\mathcal{A}} \Big(c^0_{ij} X_{ij} + \sum_{\ell=1}^{L} \lambda^{\ell} \sum_{t=0}^{T} c^{\ell}_{ij} Z^{\ell}_{ij}(t) \Big) \tag{32}$$

subject to (30)–(31) and (26)–(27), and (28). This problem, the deterministic equivalent to a two-stage stochastic program, can be solved by standard linear programming techniques, such as the simplex method or interior point methods. In addition, a GAMS link to the SPOSL solver [26] for stochastic programs is under development and has been used on this problem. This link accepts the deterministic equivalent, but passes it to SPOSL in stochastic form for more efficient solution.

```
set     i 'nodes in traffic network'  /1*24/;
alias (j,i), (k,i), (l,i);

set     dest(j) identification of destination nodes,
        arcs(i,i)  arcs;
$include sioux-falls.dat

set time / t0*t100 /
    fixed(time) 'steady state solution reached' ;
fixed(time) = yes$(ord(time) gt 41);

* nodes in the stochastic tree, node0 the root
set nodes 'nodes' /node0*node2/;
parameter cost(nodes,i,i), capacity(i,i), tranrate(nodes);

cost(nodes,arcs) = coef_a(arcs);
cost("node1","1","2") = 100; cost("node2","21","24") = 100;
capacity(arcs) = inf;
capacity("15","14") = 0.5; capacity("22","23") = 0.5;

variables       z(nodes,i,j,time), obj;

equations       balance(nodes,i),
                robbalance(nodes,i,time), costdef;

balance(nodes,i)$(ord(nodes) eq 1 and not dest(i))..
  sum(arcs(i,j), z(nodes,arcs,"t0"))
  - sum(arcs(j,i), z(nodes,arcs,"t0"))
  =e= demand(i);

robbalance(nodes,i,time)$(ord(nodes) gt 1 and not dest(i))..
  sum(arcs(i,j), z(nodes,arcs,time))
  - sum(arcs(j,i), z(nodes,arcs,time-cost(nodes,arcs)))
  =e= demand(i) +
  sum(arcs(j,i)$(cost("node0",arcs) ge ord(time)),
      z("node0",arcs,"t0"));

costdef..
  obj =e= sum(arcs(i,j), sum(nodes, tranrate(nodes)*
              sum(time, cost(nodes,arcs)*z(nodes,arcs,time))));
```

Figure 3: Robust Path Choice: GAMS model using SP/OSL

```
z.lo(nodes,arcs,time) = 0.0;
z.up(nodes,arcs,time) = capacity(arcs);
z.fx(nodes,i,j,time)$(not arcs(i,j)) = 0;
z.fx(nodes,dest,j,time)$(arcs(dest,j)) = 0;
z.fx("node0",arcs,time)$(ord(time) gt 1) = 0;

model robust / costdef, balance, robbalance /;

tranrate(nodes) = 0; tranrate("node0") = 1;
option lp = sposl;
solve robust using lp minimizing obj;

tranrate("node1") = 0.01; tranrate("node2") = 0.05;

* determine steady state flows after each scenario has occurred
set inc_node(nodes); inc_node(nodes) = no;
equation steadyobj(nodes), steadybal(nodes,i);

steadybal(inc_node,i)$(not dest(i))..
  sum(arcs(i,j), z(inc_node,arcs,"t0"))
  - sum(arcs(j,i), z(inc_node,arcs,"t0"))
  =e=  demand(i);

steadyobj(inc_node)..
  obj =e= sum(arcs, cost(inc_node,arcs)*z(inc_node,arcs,"t0"));

model steady / steadyobj, steadybal /;

option lp = osl;
loop(nodes$(ord(nodes) gt 1),
  inc_node(nodes) = yes;
  solve steady using lp minimizing obj;
  z.fx(nodes,arcs,fixed) = z.l(nodes,arcs,"t0");
  inc_node(nodes) = no;
);

option lp = sposl;
solve robust using lp minimizing obj;
```

Figure 4: Robust Path Choice: GAMS model using SP/OSL (cont)

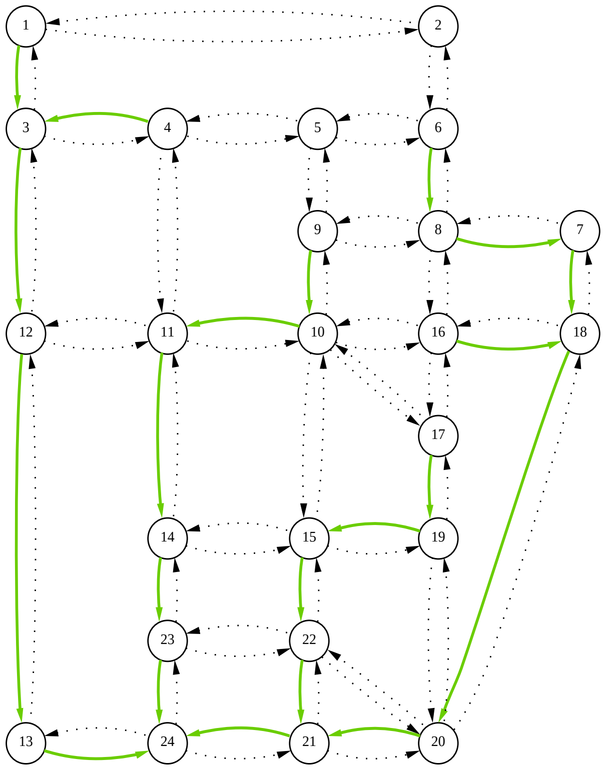

Figure 5: Shortest path solution for Sioux Falls network.

We have investigated the effects of our modeling format on a simple example using the same Sioux Falls network as in the MPEC model. Our GAMS implementation of the model is shown in Figure 3 and Figure 4.

Ignoring the possibility of failure on the arcs, we first solved the shortest path problems to find the solution shown in Figure 5. In Figure 4, this is computed as a special case of the robust plan, with 0 probability of failure. We then considered the possibility of 2 failures in the network, on arcs (1,2) and arc (21,24). We incorporated capacities of 0.5 only on arcs (15,14) and (22,23) so that when a failure of arc (21,24) occurs, all the flow could not be rerouted through these arcs. The loop statement in Figure 4 is used to compute the steady state solution for each arc failure, required to enforce (28). The resulting robust solution plan obtained from minimizing (32) subject to (30)–(31), (26)–(27) and (28) is depicted in Figure 6. An interesting paradox can be observed. Arcs (15,14) and (22,21) that are not

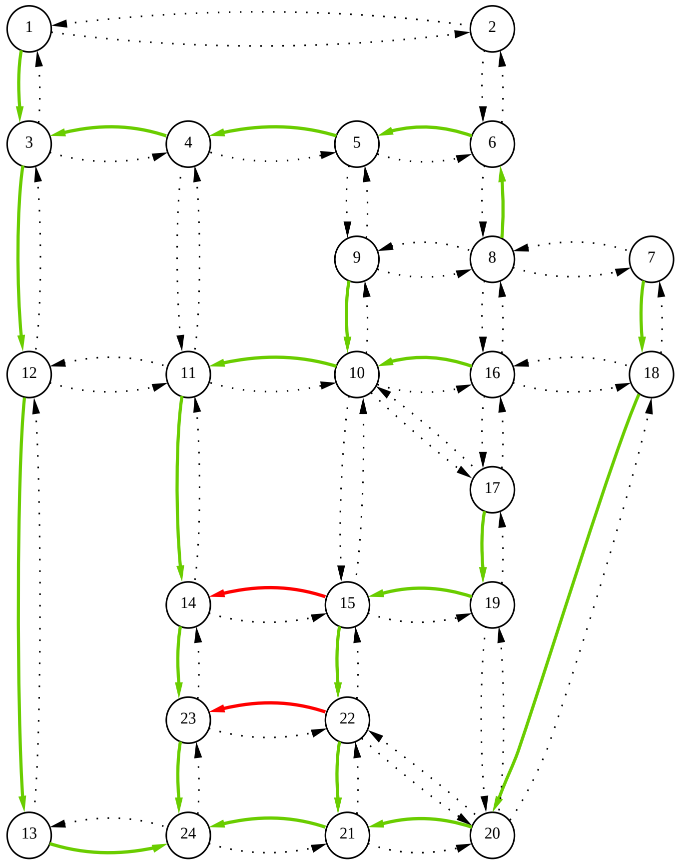

Figure 6: Robust solution in capacitated case ($T = 40$).

used at all in the shortest path plan have saturating flow sent across them in the robust plan. This paradox can be explained by the fact that the arcs are heavily used for rerouting. Thus, to avoid the major expense of rerouting large amounts of flow through arc (10,11) in the event that (21,24) fails, it is better to send as much flow as possible away from potential bottlenecks. Also, in the robust plan, flow is sent from node 8 to node 6, which is in direct contrast to the shortest path solution depicted in Figure 5 which sends flow from 6 to 8.

In order to demonstrate the effect of our robust plan on the transient behavior of the jam, we show the flows on two representative arcs after a failure occurs. These rerouting flows are calculated under two different plans. The charts on the top of Figure 7 depict the transient behavior of the flows on the arcs (15,10) and (10,11) under the plan that chooses shortest paths initially, and then reroutes the flow when a failure occurs. The charts on the bottom depict the rerouting flows that occur when we follow the robust plan; both of these rerouting procedures allow a period $T_p = 40$ to attain the steady state solution. Note that on both of these arcs, the amount of flow that has to be rerouted in the robust case is less than half that needed to be rerouted when the shortest path plan is followed.

8 Conclusions

In this paper, we have shown the connection between various forms of traffic design problems and modeling formats based on complementarity and variational inequalities. We have used the modeling language GAMS to develop all the models in this paper; extensions to other modeling languages such as AIMMS and AMPL could be similarly implemented.

Stochastic modeling and MPEC solution are active areas of current research. We believe that some emerging algorithms for MPEC and stochastic LP may prove to be suitably quick and robust to satisfactorily solve large classes of these models. The techniques we have described here allow for easy formulation of many MPEC's and interfacing with solvers. The class of problems for which the algorithms described herein are applicable is restricted by size of the problem and/or dealing with underlying nonconvexities. Future research extending both the models and tools outlined in this paper will remain critical for more reliable problem formulation and solution.

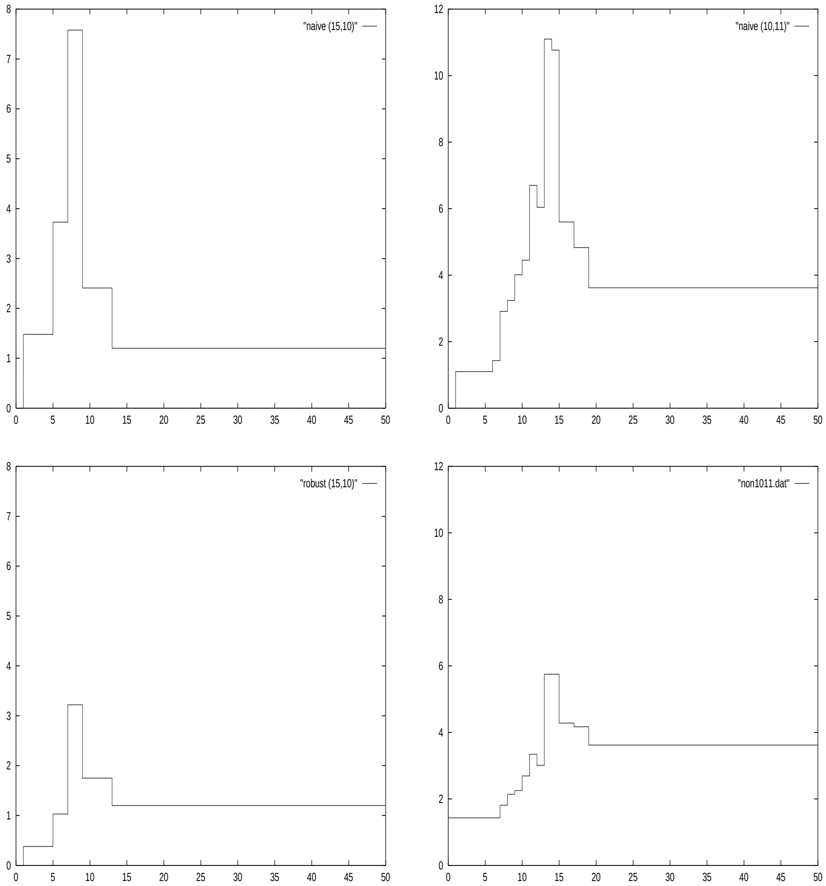

Figure 7: Flow on (15,10) and (10,11) during transition from shortest path plan (top) or robust plan (bottom). Failure occurs at $t = 1$.

References

[1] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont MA, 1995.

[2] D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16:580–595, 1991.

[3] A. Brooke, D. Kendrick, and A. Meeraus. *GAMS: A User's Guide*. The Scientific Press, South San Francisco, CA, 1988.

[4] Y. Chen and M. Florian. O-D demand adjustment problem with congestion: Part I. model analysis and optimality conditions. Publication CRT-94-56, Centre de Recherche sur les Transports, Université de Montréal, Montréal, Canada, 1994.

[5] A. R. Conn and Ph. L. Toint. An algorithm using quadratic interpolation for unconstrained derivative free optimization. In G. Di Pillo and F. Giannessi, editors, *Nonlinear Optimization and Applications*. Plenum Press, New York, 1996.

[6] S. P. Dirkse and M. C. Ferris. MCPLIB: A collection of nonlinear mixed complementarity problems. *Optimization Methods and Software*, 5:319–345, 1995.

[7] S. P. Dirkse and M. C. Ferris. The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software*, 5:123–156, 1995.

[8] S. P. Dirkse and M. C. Ferris. A pathsearch damped Newton method for computing general equilibria. *Annals of Operations Research*, 1996.

[9] S. P. Dirkse and M. C. Ferris. Crash techniques for large-scale complementarity problems. In Ferris and Pang [14].

[10] S. P. Dirkse, M. C. Ferris, P. V. Preckel, and T. Rutherford. The GAMS callable program library for variational and complementarity solvers. Mathematical Programming Technical Report 94-07, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1994.

[11] B. C. Eaves. On the basic theorem of complementarity. *Mathematical Programming*, 1:68–87, 1971.

[12] M. C. Ferris and C. Kanzow. Recent developments in the solution of nonlinear complementarity problems. Technical report, Computer Sciences Department, University of Wisconsin, 1997. In preparation.

[13] M. C. Ferris, A. Meeraus, and T. F. Rutherford. Computing Wardropian equilibrium in a complementarity framework. Mathematical Programming Technical Report 95-03, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1995.

[14] M. C. Ferris and J. S. Pang, editors. *Complementarity and Variational Problems: State of the Art*, Philadelphia, Pennsylvania, 1997. SIAM Publications.

[15] M. C. Ferris and J. S. Pang. Engineering and economic applications of complementarity problems. *SIAM Review, forthcoming*, 1997.

[16] M. C. Ferris and A. Ruszczyński. Robust path choice and vehicle guidance in networks with failures. Mathematical Programming Technical Report 97-04, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1997.

[17] M. C. Ferris and T. F. Rutherford. Accessing realistic complementarity problems within Matlab. In G. Di Pillo and F. Giannessi, editors, *Nonlinear Optimization and Applications*. Plenum Press, New York, 1996.

[18] M. Florian, editor. *Traffic Equilibrium Methods*, Berlin, 1976. Springer-Verlag.

[19] M. Florian. Nonlinear cost network models in transportation analysis. *Mathematical Programming Study*, 26:167–196, 1986.

[20] T. L. Friesz. Network equilibrium, design and aggregation. *Transportation Research*, 19A:413–427, 1985.

[21] T. L. Friesz, R. L. Tobin, T. E. Smith, and P. T. Harker. A nonlinear complementarity formulation and solution procedure for the general derived demand network equilibrium problem. *Journal of Regional Science*, 23:337–359, 1983.

[22] P. T. Harker. *Lectures on Computation of Equilibria with Equation–Based Methods*. CORE Lecture Series. CORE Foundation, Louvain–la–Neuve, Université Catholique de Louvain, 1993.

[23] A. Haurie and P. Marcotte. On the relationship between Nash–Cournot and Wardrop equilibria. *Networks*, 15:295–308, 1985.

[24] D. W. Hearn, S. Lawphongpanich, and J. A. Ventura. Restricted simplicial decomposition: Computation and extensions. *Mathematical Programming Study*, 31:99–118, 1987.

[25] P. Kall and S. W. Wallace. *Stochastic Programming*. John Wiley & Sons, Chichester, 1994.

[26] A. J. King. SP/OSL version 1.0 stochastic programming interface library user's guide. Technical Report RC 19757, IBM T.J. Watson Research Center, Yorktown Heights, New York, 1994.

[27] M. Kočvara and J. V. Outrata. On optimization of systems governed by implicit complementarity problems. Technical Report 513, Institute of Applied Mathematics, University of Jena, Leutragraben, 1994.

[28] M. Kočvara and J. V. Outrata. On the solution of optimum design problems with variational inequalities. In *Recent Advances in Nonsmooth Optimization*, pages 172–192. World Scientific Publishers, Singapore, 1995.

[29] Z.-Q. Luo, J. S. Pang, and D. Ralph. *Mathematical Programs with Equilibrium Constraints.* Cambridge University Press, 1996.

[30] T. L. Magnanti. Models and algorithms for predicting urban traffic equilibria. In M. Florian, editor, *Transportation Planning Models*, pages 153–186. North Holland, 1984.

[31] P. Marcotte. Network design problem with congestion effects: A case of bilevel programming. *Mathematical Programming*, 34:142–162, 1986.

[32] J. V. Outrata and J. Zowe. A numerical approach to optimization problems with variational inequality constraints. *Mathematical Programming*, 68:105–130, 1995.

[33] J. S. Pang and C. S. Yu. Linearized simplicial decomposition methods for computing traffic equilibria on networks. *Networks*, 14:427–438, 1984.

[34] G. H. Polychronopoulos and J. N. Tsitsiklis. Stochastic shortest path problems with recourse. *Networks*, 27:133–143, 1996.

[35] A. Prékopa. *Stochastic Programming.* Kluwer Academic Publishers, Dordrecht, 1995.

[36] D. Ralph. Global convergence of damped Newton's method for nonsmooth equations, via the path search. *Mathematics of Operations Research*, 19:352–389, 1994.

[37] S. M. Robinson. Mathematical foundations of nonsmooth embedding methods. *Mathematical Programming*, 48:221–229, 1990.

[38] S. M. Robinson. Normal maps induced by linear transformations. *Mathematics of Operations Research*, 17:691–714, 1992.

[39] H. Schramm and J. Zowe. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM Journal on Optimization*, 2:121–152, 1992.

[40] M. J. Smith. The existence, uniqueness and stability of traffic equilibria. *Transportation Research*, 13B:295–304, 1979.

[41] Ph. L. Toint. Transportation modelling and operations research: A fruitful connection. In Ph. L. Toint, M. Labbe, K. Tanczos, and G. Laporte, editors, *Operations Research and Decision Aid Methodologies in Traffic and Transportation Management.* Springer-Verlag, 1997.

[42] J. G. Wardrop. Some theoretical aspects of road traffic research. *Proceeding of the Institute of Civil Engineers, Part II*, pages 325–378, 1952.

[43] R. J.-B. Wets. Large scale linear programming techniques in stochastic programming. In Yu. M. Ermoliev and R. J. B. Wets, editors, *Numerical Techniques for Stochastic Optimization*, pages 65–93. Springer-Verlag, Berlin, 1988.