Mathematical Programs with Equilibrium Constraints: Automatic Reformulation and Solution via Constrained Optimization *

Michael C. Ferris[†] Steven P. Dirkse[‡]

Alexander Meeraus[‡]

March 2002, revised July 2002

Abstract

Constrained optimization has been extensively used to solve many large scale deterministic problems arising in economics, including, for example, square systems of equations and nonlinear programs. A separate set of models have been generated more recently, using complementarity to model various phenomenon, particularly in general equilibria. The unifying framework of mathematical programs with equilibrium constraints (MPEC) has been postulated for problems that combine facets of optimization and complementarity. This paper briefly reviews some methods available to solve these problems and describes a new suite of tools for working with MPEC models. Computational results demonstrating the potential of this tool are given that automatically construct and solve a variety of different nonlinear programming reformulations of MPEC problems.

---

*This material is based on research partially supported by the National Science Foundation Grant CCR-9972372, the Air Force Office of Scientific Research Grant F49620-01-1-0040, Microsoft Corporation and the Guggenheim Foundation

†Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD Permanent address: Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, Wisconsin 53706, USA

‡GAMS Development Corporation, 1217 Potomac Street, N.W., Washington, D.C. 20007

1 Introduction

Nonlinear complementarity problems arise in many economic applications, most notably in the applied general equilibrium area [1, 29]. The past decade has seen an enormous increase in ability to solve large scale complementarity problems, due not only to the phenomenal increase in computer speed, but also to advances made in algorithms and software for complementarity problems. This paper attempts to review some of those advances, and revisits some older techniques for the purpose of solving optimization problems with complementarity constraints, typically termed Mathematical Programs with Equilibrium Constraints (MPEC's) in the literature [22, 26, 32].

Three advances in the past two decades have increased the capability of a modeler to solve large scale complementarity problems. The first is the implementation of large-scale complementarity solvers such as MILES [38], PATH [8] and SMOOTH [30] that exploit significant advances in techniques of linear algebra and nonlinear optimization. The second is the advent of modeling systems that are able to directly express complementarity problems as part of their syntax [3, 39, 13, 15] and to pass on the complementarity model to the solver. Included in this, are so called mini-languages, such as MPSGE[40], that allow particular important application domains to express their problems in a manner convenient to them. Furthermore, the ability of modeling systems to provide accurate first and second order derivatives vastly improves the reliability of the solver. The third advance is due to the interactions that the first two foster. The ability of a modeler to generate realistic, large scale models enables the solvers to be tested on much larger and more difficult classes of models. In many cases, new models point to deficiencies in particular facets of a solver which frequently lead to further enhancements and improved reliability [16, 21]. Furthermore, the ability to solve larger and more complex complementarity problems furthers the development of new applied economic models.

While it is clear that the state-of-the-art in solution mechanisms for MPEC's is currently far less satisfactory than that of complementarity problems, the intent of the present paper is to outline tools and approaches that may facilitate solution of MPEC's. The intent of providing these tools is to highlight the potential for new questions that can be asked of this more general model format and to foster the development of a much broader and more realistic suite of examples for algorithmic design and improvement. The aim of the paper is to initiate a dialogue between modelers and algorithm developers.

The main approach for the solution of optimization problems with complementarity constraints used in this paper is a reformulation of the problem as a standard nonlinear program, thus enabling solution using existing nonlinear programming algorithms. Attempts to do this in the past have been widespread and much of this paper builds on the lessons and examples that previous researchers have exhibited. We start the paper in Section 2 by outlining several reformulations of the MPEC as a standard nonlinear program. Inherent in such an approach are the techniques used to process the complementarity constraints, and it is natural to ask whether such approaches can be used to solve complementarity problems, essentially the underlying feasibility problem. Such techniques were somewhat discredited in the 1970's and 1980's, mainly due to the lack of robustness in finding feasible (hence complementary) solutions. The past decade has given rise to new formulations of the complementarity relationships that warrant further investigation, along with significant advances in robustness and variety of nonlinear programming solvers. Section 3 outlines the tools that we provide to perform the conversion automatically. Assuming the modeler provides a GAMS description of the MPEC, the tools generate a large variety of different but equivalent nonlinear programming formulations of the model in a variety of input formats. Some preliminary computational results using these tools then follow. Section 4.1 describes a set of experiments to outline how these approaches work on a small subset of complementarity problems known to be difficult to solve. We then proceed to describe some techniques for dealing with problems that have multiple complementary solutions. In particular, an example of how to determine all the Nash equilibria is given. The complete set of problems from MPEClib are then processed with a number of nonlinear programming solvers.

2 Formulations of the model

We consider the following optimization problem:

$$\min_{x\in\mathbf{R}^n, y\in\mathbf{R}^m} f(x,y) \tag{1}$$

subject to the constraints

$$g(x,y) \in K \tag{2}$$

and

$$y \text{ solves } \mathrm{MCP}(h(x,\cdot),\mathbf{B}). \tag{3}$$

The objective function (1) needs no further description, except to state that the solution techniques we are intending to apply require that f (g and h) are at least once differentiable, and for some solvers twice differentiable.

The constraints (2) are intended to represent standard nonlinear programming constraints. In particular, we assume that K is the Cartesian product of K_i so that equality constraints arise whenever $K_i = \{0\}$ and less-than (greater-than) inequality constraints arise when $K_i = \{\xi : \xi \leq 0\}$ ($\{\xi : \xi \geq 0\}$). Since these constraints will be unaltered in all our reformulations, we use this notation for brevity.

The constraints that are the concern of this paper are the equilibrium constraints (3). Essentially, these are parametric constraints (parameterized by x) on the variable y. (3) signifies that y is a solution to the mixed complementarity problem (MCP) that is defined by the function $h(x,\cdot)$ and the bound set $\mathbf{B}$. Due to this constraint (frequently called an equilibrium constraint), problems of this form are typically termed mathematical program with equilibrium constraints [26, 32]. We now define the precise meaning of this statement.

We partition the y variables into free $\mathcal{F}$, lower bounded $\mathcal{L}$, upper bounded $\mathcal{U}$ and doubly bounded $\mathcal{B}$ variables respectively, that is:

$$\mathbf{B} := \{y = (y_\mathcal{F}, y_\mathcal{L}, y_\mathcal{U}, y_\mathcal{B}) : a_\mathcal{L} \leq y_\mathcal{L},\ y_\mathcal{U} \leq b_\mathcal{U},\ a_\mathcal{B} \leq y_\mathcal{B} \leq b_\mathcal{B}\}$$

where it is assumed (without loss of generality) that $a_\mathcal{B} < b_\mathcal{B}$. Thus the box $\mathbf{B}$ represents simple bounds on the variables y.

The constraints (3) can now be given a precise meaning. They are entirely equivalent to the following system of equalities and inequalities:

$$\begin{array}{l} a_\mathcal{L} \leq y_\mathcal{L},\ h_\mathcal{L}(x,y) \geq 0 \text{ and } (y_\mathcal{L} - a_\mathcal{L})^T h_\mathcal{L}(x,y) = 0 \\ y_\mathcal{U} \leq b_\mathcal{U},\ h_\mathcal{U}(x,y) \leq 0 \text{ and } (y_\mathcal{U} - b_\mathcal{U})^T h_\mathcal{U}(x,y) = 0 \end{array} \tag{4}$$

and for each $i \in \mathcal{B}$ exactly one of the following must hold:

$$\begin{array}{c} a_i < y_i < b_i,\ h_i(x,y) = 0 \\ y_i = a_i,\ h_i(x,y) \geq 0 \\ y_i = b_i,\ h_i(x,y) \leq 0. \end{array} \tag{5}$$

Note in particular, that $y \in \mathbf{R}^m$ and h maps into a space of the same dimension m. Furthermore, the bounds on the variable y determine the constraints that are satisfied by h. Informally, the constraints represent orthogonality between the variables y and the function h. Some special cases are of particular interest and help illuminate the formulation. Whenever the variable is free ($a_i = -\infty$ and $b_i = +\infty$) it follows from (5) that $h_i(x, y) = 0$. Thus if all the y variables are free, then the complementarity problem is simply a system of nonlinear equations, and the MPEC is a nonlinear program. While there may be cases in which $a_i = -\infty$ and $b_i = +\infty$ is desirable, they are not of interest to the techniques developed here; we simply amalgamate such functions h_i into g.

For a second example, suppose a lower bound a_i is zero, then by (4) it follows that h_i is constrained to be nonnegative, and furthermore that the product $y_i h_i(x, y)$ must be zero. This latter conclusion follows from the simple fact that each term in the inner product given in (4) is nonnegative, and a sum of nonnegative terms can be zero only if each of the terms themselves are zero. We use this simple fact throughout this paper without further reference; it always allows us to treat the complementarity "inner product" term either in aggregate form or split up into separate components. The variable y_i is said to be complementary to the function h_i. It is these cases and further generalizations with finite lower and/or upper bounds that are of interest here.

Of course, the relative number of complementarity constraints compared to the number of general nonlinear constraints (i.e those involving g) can have significant effects on the type of method that should be chosen to solve the problem. Implicit methods [32] work well when the complementarity constraints dominate and satisfy certain regularity conditions. They are typically limited by the ability to solve the resulting nonsmooth problem in the variable x. When the number of complementarity constraints are small, then nonlinear programming techniques should be more applicable. In this paper, we attempt to solve both types of problem using nonlinear programming reformulations.

Unfortunately, the constraints imposed by (5) depend on the solution value of y. For this reason, it is often convenient to introduce new variables

$w_{\mathcal{B}}$ and $v_{\mathcal{B}}$ and rewrite (5) in an equivalent manner as:

$$\begin{aligned} w_{\mathcal{B}} - v_{\mathcal{B}} &= h_{\mathcal{B}}(x,y) \\ a_{\mathcal{B}} \le y_{\mathcal{B}} \le b_{\mathcal{B}},\ & w_{\mathcal{B}} \ge 0,\ v_{\mathcal{B}} \ge 0 \\ (y_{\mathcal{B}} - a_{\mathcal{B}})^T w_{\mathcal{B}} = 0,\ & (b_{\mathcal{B}} - y_{\mathcal{B}})^T v_{\mathcal{B}} = 0. \end{aligned} \tag{6}$$

We often introduce auxiliary variables for the constraints (4) as well to remove the need for a nonlinear solver to evaluate the derivatives of $h_{\mathcal{L}}$ and $h_{\mathcal{U}}$ more than once. Thus, (4) can be equivalently written as:

$$\begin{aligned} & w_{\mathcal{L}} = h_{\mathcal{L}}(x,y),\ a_{\mathcal{L}} \le y_{\mathcal{L}},\ w_{\mathcal{L}} \ge 0 \text{ and } (y_{\mathcal{L}} - a_{\mathcal{L}})^T w_{\mathcal{L}} = 0 \\ & v_{\mathcal{U}} = -h_{\mathcal{U}}(x,y),\ y_{\mathcal{U}} \le b_{\mathcal{U}},\ v_{\mathcal{U}} \ge 0 \text{ and } (b_{\mathcal{U}} - y_{\mathcal{U}})^T v_{\mathcal{U}} = 0. \end{aligned} \tag{7}$$

Note that the size of the model will increase due to the additional artificial variables.

We collect all the "auxiliary definitions" together to simplify the ensuing discussion. Thus, we define a set $\mathcal{H}$ by

$$\begin{aligned} & (x,y,w,v) \in \mathcal{H} \iff \\ & \qquad g(x,y) \in K,\ w_{\mathcal{L}} = h_{\mathcal{L}}(x,y),\ v_{\mathcal{U}} = -h_{\mathcal{U}}(x,y),\ w_{\mathcal{B}} - v_{\mathcal{B}} = h_{\mathcal{B}}(x,y) \\ & \qquad \text{and } y \in \mathbf{B},\ w_{\mathcal{L}} \ge 0,\ v_{\mathcal{U}} \ge 0,\ w_{\mathcal{B}} \ge 0,\ v_{\mathcal{B}} \ge 0. \end{aligned}$$

Collecting all these observations together gives the first nonlinear programming formulation that we will consider:

$$\begin{aligned} & \min_{(x,y,w,v)\in\mathcal{H}} f(x,y) \\ & \text{subject to } (y_i - a_i) w_i = \mu,\ i \in \mathcal{L} \cup \mathcal{B} \\ & \qquad\qquad\ (b_i - y_i) v_i = \mu,\ i \in \mathcal{U} \cup \mathcal{B}. \end{aligned} \tag{8}$$

All the reformulations we give in this paper are parametrized by a scalar value μ. For $\mu = 0$, the above formulation corresponds precisely to the MPEC given as (1), (2) and (3) with the inner products treated componentwise. For positive values of μ the complementarity product terms are forced to be equal to μ; as μ is decreased to zero the corresponding solutions lie on what is typically called the "central path" in the interior point literature [50].

It is clear that each of the terms involved in the inner products of (6) and (7) are all themselves nonnegative, and hence the equality with 0 can be replaced by a less-than inequality.

$$
\begin{aligned}
&\min_{(x,y,w,v)\in\mathcal{H}} f(x,y) \\
&\text{subject to } (y_i - a_i)w_i \le \mu,\ i \in \mathcal{L}\cup\mathcal{B} \\
&\qquad\qquad\quad (b_i - y_i)v_i \le \mu,\ i \in \mathcal{U}\cup\mathcal{B}.
\end{aligned} \tag{9}
$$

Again, for $\mu = 0$, this corresponds to the MPEC given as (1), (2) and (3). For positive values of μ this corresponds to a componentwise relaxation of the original problem.

The following formulation aggregates all the complementarity constraints:

$$
\begin{aligned}
&\min_{(x,y,w,v)\in\mathcal{H}} f(x,y) \\
&\text{subject to } (y_\mathcal{L} - a_\mathcal{L})^T w_\mathcal{L} + (b_\mathcal{U} - y_\mathcal{U})^T v_\mathcal{U} + (y_\mathcal{B} - a_\mathcal{B})^T w_\mathcal{B} \\
&\qquad\qquad\qquad + (b_\mathcal{B} - y_\mathcal{B})^T v_\mathcal{B} \le \mu.
\end{aligned} \tag{10}
$$

A partial aggregation can also be carried out:

$$
\begin{aligned}
&\min_{(x,y,w,v)\in\mathcal{H}} f(x,y) \\
&\text{subject to } (y_\mathcal{L} - a_\mathcal{L})^T w_\mathcal{L} \le \mu,\ \ (b_\mathcal{U} - y_\mathcal{U})^T v_\mathcal{U} \le \mu \\
&\qquad\qquad\quad (y_\mathcal{B} - a_\mathcal{B})^T w_\mathcal{B} \le \mu,\ \ (b_\mathcal{B} - y_\mathcal{B})^T v_\mathcal{B} \le \mu.
\end{aligned} \tag{11}
$$

There is of course a similar aggregation for (8) that immediately leads to the problem

$$
\begin{aligned}
&\min_{(x,y,w,v)\in\mathcal{H}} f(x,y) \\
&\text{subject to } (y_\mathcal{L} - a_\mathcal{L})^T w_\mathcal{L} + (b_\mathcal{U} - y_\mathcal{U})^T v_\mathcal{U} + (y_\mathcal{B} - a_\mathcal{B})^T w_\mathcal{B} \\
&\qquad\qquad\qquad + (b_\mathcal{B} - y_\mathcal{B})^T v_\mathcal{B} = \mu.
\end{aligned} \tag{12}
$$

It is well known that the above formulations (for $\mu = 0$) have poor theoretical properties in terms of the classical constraint qualifications.

Instead of using the auxiliary variables $w_\mathcal{L}$ and $v_\mathcal{U}$ we can substitute the relevant functions into the formulations explicitly. To facilitate a more succinct description, we introduce a new set $\tilde{\mathcal{H}}$ that collects the definitions together.

$$
\begin{aligned}
&(x,y,w,v) \in \tilde{\mathcal{H}} \iff \\
&\qquad g(x,y) \in K,\ \ y \in \mathbf{B},\ \ h_\mathcal{L}(x,y) \ge 0,\ h_\mathcal{U}(x,y) \le 0 \\
&\qquad \text{and } w_\mathcal{B} - v_\mathcal{B} = h_\mathcal{B}(x,y),\ w_\mathcal{B} \ge 0,\ v_\mathcal{B} \ge 0.
\end{aligned}
$$

We rewrite four of the above reformulations with such an elimination:

$$
\begin{aligned}
\min_{(x,y,w,v)\in\tilde{\mathcal{H}}} \; & f(x,y) \\
\text{subject to } & (y_i - a_i)h_i(x,y) = \mu, \; i \in \mathcal{L} \\
& (b_i - y_i)h_i(x,y) = -\mu, \; i \in \mathcal{U} \\
& (y_i - a_i)w_i + (b_i - y_i)v_i = \mu, \; i \in \mathcal{B}
\end{aligned}
\tag{13}
$$

$$
\begin{aligned}
\min_{(x,y,w,v)\in\tilde{\mathcal{H}}} \; & f(x,y) \\
\text{subject to } & (y_i - a_i)h_i(x,y) \le \mu, \; i \in \mathcal{L} \\
& (b_i - y_i)h_i(x,y) \ge -\mu, \; i \in \mathcal{U} \\
& (y_i - a_i)w_i \le \mu, \;\; (b_i - y_i)v_i \le \mu, \; i \in \mathcal{B}
\end{aligned}
\tag{14}
$$

$$
\begin{aligned}
\min_{(x,y,w,v)\in\tilde{\mathcal{H}}} \; & f(x,y) \\
\text{subject to } & (y_{\mathcal{L}} - a_{\mathcal{L}})^T h_{\mathcal{L}}(x,y) \le \mu, \;\; (b_{\mathcal{U}} - y_{\mathcal{U}})^T h_{\mathcal{U}}(x,y) \ge -\mu \\
& (y_{\mathcal{B}} - a_{\mathcal{B}})^T w_{\mathcal{B}} \le \mu, \;\; (b_{\mathcal{B}} - y_{\mathcal{B}})^T v_{\mathcal{B}} \le \mu
\end{aligned}
\tag{15}
$$

$$
\begin{aligned}
\min_{(x,y,w,v)\in\tilde{\mathcal{H}}} \; & f(x,y) \\
\text{subject to } & (y_{\mathcal{L}} - a_{\mathcal{L}})^T h_{\mathcal{L}}(x,y) - (b_{\mathcal{U}} - y_{\mathcal{U}})^T h_{\mathcal{U}}(x,y) + (y_{\mathcal{B}} - a_{\mathcal{B}})^T w_{\mathcal{B}} \\
& \qquad + (b_{\mathcal{B}} - y_{\mathcal{B}})^T v_{\mathcal{B}} = \mu.
\end{aligned}
\tag{16}
$$

A different approach involves a penalization of the complementarity conditions. We add a weighted sum of the complementarity conditions to the objective function, removing the complementarity conditions from the constraints in (10). By decreasing μ, the weight on the complementarity conditions becomes progressively larger.

$$
\begin{aligned}
\min_{(x,y,w,v)\in\mathcal{H}} f(x,y) + \frac{1}{\mu}\{ & (y_{\mathcal{L}} - a_{\mathcal{L}})^T w_{\mathcal{L}} + (b_{\mathcal{U}} - y_{\mathcal{U}})^T v_{\mathcal{U}} + \\
& (y_{\mathcal{B}} - a_{\mathcal{B}})^T w_{\mathcal{B}} + (b_{\mathcal{B}} - y_{\mathcal{B}})^T v_{\mathcal{B}}\}.
\end{aligned}
\tag{17}
$$

A similar scheme works with (16):

$$
\begin{aligned}
\min_{(x,y,w,v)\in\tilde{\mathcal{H}}} f(x,y) + \frac{1}{\mu}\{ & (y_{\mathcal{L}} - a_{\mathcal{L}})^T h_{\mathcal{L}}(x,y) - (b_{\mathcal{U}} - y_{\mathcal{U}})^T h_{\mathcal{U}}(x,y) + \\
& (y_{\mathcal{B}} - a_{\mathcal{B}})^T w_{\mathcal{B}} + (b_{\mathcal{B}} - y_{\mathcal{B}})^T v_{\mathcal{B}}\}.
\end{aligned}
\tag{18}
$$

A simple calculation (suggested in [19]) allows one to see that for two scalars r and s, the following holds:

$$\phi(r,s) = 0 \iff r \geq 0,\ s \geq 0 \text{ and } rs = 0,$$

where

$$\phi(r,s) := \sqrt{r^2 + s^2} - (r + s).$$

Note that ϕ is not differentiable at the origin which may lead to solution difficulties. To overcome the nondifferentiability problems a variety of smoothing approaches have been suggested. Essentially, they replace the solution of the MPEC by a parameterized NLP(μ), and solve a sequence of problems for decreasing values of $\mu > 0$. The perturbation μ guarantees differentiability of all constraint functions by replacing ϕ by

$$\phi_\mu(r,s) := \sqrt{r^2 + s^2 + \mu} - (r + s).$$

Note that $\phi_\mu(r,s) = 0$ if and only if $r > 0$, $s > 0$ and $rs = \mu/2$. Thus, the complementarity condition is satisfied in the limit as μ goes to zero. The formulation given below was proposed in [12]:

$$\begin{aligned} &\min_{(x,y,w,v)\in\mathcal{H}} f(x,y) \\ &\text{subject to } \phi_\mu(y_i - a_i, w_i) = 0,\ i \in \mathcal{L} \cup \mathcal{B} \\ &\qquad\qquad\ \ \phi_\mu(b_i - y_i, v_i) = 0,\ i \in \mathcal{U} \cup \mathcal{B}. \end{aligned} \tag{19}$$

It is also possible to rewrite the complementarity constraints as a system of nonlinear equations, namely

$$\begin{aligned} \min(y_\mathcal{L} - a_\mathcal{L}, h_\mathcal{L}(x,y)) &= 0 \\ \min(b_\mathcal{U} - y_\mathcal{U}, -h_\mathcal{U}(x,y)) &= 0 \\ \min(y_\mathcal{B} - a_\mathcal{B}, h_\mathcal{B}(x,y)) &= 0 \\ \min(b_\mathcal{B} - y_\mathcal{B}, -h_\mathcal{B}(x,y)) &= 0. \end{aligned} \tag{20}$$

While we provide mechanisms to form the nonlinear program using this construction, a modeler should note that the following formulation involves nonsmooth functions and thus appropriate solvers need to be invoked.

$$\begin{aligned} &\min_{(x,y,w,v)\in\mathcal{H}} f(x,y) \\ &\text{subject to } \min(y_i - a_i, w_i) \leq \mu,\ \ i \in \mathcal{L} \cup \mathcal{B} \\ &\qquad\qquad\ \ \min(b_i - y_i, v_i) \leq \mu,\ \ i \in \mathcal{U} \cup \mathcal{B} \end{aligned} \tag{21}$$

A smoothed version of (20) was proposed in [5]. In this case, $\min(r, s)$ is replaced by

$$\psi_\mu(r, s) = r - \mu \log(1 + \exp((r - s)/\mu)).$$

Updating the four equations in (20) using this replacement is an alternative way to enforce complementarity as μ is driven to 0:

$$\begin{aligned} &\min_{(x,y,w,v)\in\mathcal{H}} f(x, y) \\ &\text{subject to } \psi_\mu(y_i - a_i, w_i) = 0, \;\; i \in \mathcal{L} \cup \mathcal{B} \\ &\qquad\qquad\quad \psi_\mu(b_i - y_i, v_i) = 0, \;\; i \in \mathcal{U} \cup \mathcal{B}. \end{aligned} \tag{22}$$

It is easy to see that the functions ϕ_μ, min and ψ_μ enforce the nonnegativity of their arguments in the limit without needed the additional bounding constraints. In the following we simply remove the bound constraints in the definition of $\tilde{\mathcal{H}}$ leaving the following:

$$\begin{aligned} &(x, y, w, v) \in \mathcal{H}^* \iff \\ &\qquad g(x, y) \in K, \; w_{\mathcal{L}} = h_{\mathcal{L}}(x, y), \; v_{\mathcal{U}} = -h_{\mathcal{U}}(x, y), \; w_{\mathcal{B}} - v_{\mathcal{B}} = h_{\mathcal{B}}(x, y). \end{aligned}$$

It is unknown at this time whether the bound statements help or hinder the solution process, but the tool we describe in the next section allows the modeler to make such choices, as shown by the examples below:

$$\begin{aligned} &\min_{(x,y,w,v)\in\mathcal{H}^*} f(x, y) \\ &\text{subject to } \phi_\mu(y_i - a_i, w_i) = 0, \; i \in \mathcal{L} \cup \mathcal{B} \\ &\qquad\qquad\quad \phi_\mu(b_i - y_i, v_i) = 0, \; i \in \mathcal{U} \cup \mathcal{B}, \end{aligned} \tag{23}$$

$$\begin{aligned} &\min_{(x,y,w,v)\in\mathcal{H}^*} f(x, y) \\ &\text{subject to } \min(y_i - a_i, w_i) = \mu, \;\; i \in \mathcal{L} \cup \mathcal{B} \\ &\qquad\qquad\quad \min(b_i - y_i, v_i) = \mu, \;\; i \in \mathcal{U} \cup \mathcal{B}, \end{aligned} \tag{24}$$

$$\begin{aligned} &\min_{(x,y,w,v)\in\mathcal{H}^*} f(x, y) \\ &\text{subject to } \psi_\mu(y_i - a_i, w_i) = 0, \;\; i \in \mathcal{L} \cup \mathcal{B} \\ &\qquad\qquad\quad \psi_\mu(b_i - y_i, v_i) = 0, \;\; i \in \mathcal{U} \cup \mathcal{B}. \end{aligned} \tag{25}$$

Elimination of the artificial variables $w_{\mathcal{L}}$ and $v_{\mathcal{U}}$ within ϕ_μ gives the following formulation:

$$\begin{aligned} \min_{x\in\mathbf{R}^n, y\in\mathbf{R}^m, w_{\mathcal{B}}, v_{\mathcal{B}}} & f(x,y) \\ \text{subject to } & g(x,y)\in K,\ w_{\mathcal{B}} - v_{\mathcal{B}} = h_{\mathcal{B}}(x,y) \\ & \phi_\mu(y_i - a_i, h_i(x,y)) = 0,\ i\in\mathcal{L} \\ & \phi_\mu(b_i - y_i, -h_i(x,y)) = 0,\ i\in\mathcal{U} \\ & \phi_\mu(y_i - a_i, w_i) = 0,\ \ \phi_\mu(b_i - y_i, v_i) = 0,\ i\in\mathcal{B}. \end{aligned} \tag{26}$$

We can further eliminate $w_{\mathcal{B}}$ and $v_{\mathcal{B}}$ and treat finite upper and lower bounds using an approach suggested in [2]:

$$\begin{aligned} \min_{x\in\mathbf{R}^n, y\in\mathbf{R}^m} & f(x,y) \\ \text{subject to } & g(x,y)\in K \\ & \phi_\mu(y_i - a_i, h_i(x,y)) = 0,\ i\in\mathcal{L} \\ & \phi_\mu(b_i - y_i, -h_i(x,y)) = 0,\ i\in\mathcal{U} \\ & \phi_\mu(y_i - a_i, \phi_\mu(-h_i(x,y), b_i - y_i)) = 0,\ i\in\mathcal{B}. \end{aligned} \tag{27}$$

Finally, the doubly bounded variables are sometimes treated using an alternative approach due to Scholtes:

$$\begin{aligned} \min_{x\in\mathbf{R}^n, y\in\mathbf{B}, w, v} & f(x,y) \\ \text{subject to } & g(x,y)\in K,\ w_{\mathcal{B}} = h_{\mathcal{B}}(x,y) \\ & w_{\mathcal{L}} = h_{\mathcal{L}}(x,y),\ v_{\mathcal{U}} = -h_{\mathcal{U}}(x,y),\ w_{\mathcal{L}} \ge 0, v_{\mathcal{U}} \ge 0 \\ & (y_i - a_i) w_i = \mu,\ i\in\mathcal{L} \\ & (b_i - y_i) v_i = \mu,\ i\in\mathcal{U} \\ & (y_i - a_i) w_i \le \mu,\ (b_i - y_i) w_i \ge -\mu,\ i\in\mathcal{B}. \end{aligned} \tag{28}$$

Note this is only exact when $\mu = 0$. Elimination of $w_{\mathcal{L}}$ and $v_{\mathcal{U}}$ then provides the following formulation.:

$$\begin{aligned} \min_{x\in\mathbf{R}^n, y\in\mathbf{B}, w_{\mathcal{B}}} & f(x,y) \\ \text{subject to } & g(x,y)\in K,\ w_{\mathcal{B}} = h_{\mathcal{B}}(x,y),\ h_{\mathcal{L}}(x,y)\ge 0,\ h_{\mathcal{U}}(x,y)\le 0 \\ & (y_{\mathcal{L}} - a_{\mathcal{L}})^T h_{\mathcal{L}}(x,y) - (b_{\mathcal{U}} - y_{\mathcal{U}})^T h_{\mathcal{U}}(x,y) = \mu \\ & (y_i - a_i) w_i \le \mu,\ (b_i - y_i) w_i \ge -\mu,\ i\in\mathcal{B}. \end{aligned} \tag{29}$$

It is further possible to eliminate $w_{\mathcal{B}}$ with or without aggregation on the remaining complementarity constraints:

$$\begin{aligned} &\min_{x\in\mathbf{R}^n, y\in\mathbf{B}} f(x,y) \\ &\text{subject to } g(x,y)\in K,\ h_{\mathcal{L}}(x,y)\geq 0, h_{\mathcal{U}}(x,y)\leq 0 \\ &\qquad\qquad (y_{\mathcal{L}}-a_{\mathcal{L}})^T h_{\mathcal{L}}(x,y) - (b_{\mathcal{U}}-y_{\mathcal{U}})^T h_{\mathcal{U}}(x,y) \leq \mu \\ &\qquad\qquad (y_i-a_i)h_i(x,y)\leq \mu,\ (b_i-y_i)h_i(x,y)\geq -\mu,\ i\in\mathcal{B}, \end{aligned} \tag{30}$$

$$\begin{aligned} &\min_{x\in\mathbf{R}^n, y\in\mathbf{B}} f(x,y) \\ &\text{subject to } g(x,y)\in K,\ h_{\mathcal{L}}(x,y)\geq 0, h_{\mathcal{U}}(x,y)\leq 0 \\ &\qquad\qquad (y_i-a_i)h_i(x,y) = \mu,\ i\in\mathcal{L} \\ &\qquad\qquad (b_i-y_i)h_i(x,y) = -\mu,\ i\in\mathcal{U} \\ &\qquad\qquad (y_i-a_i)h_i(x,y)\leq \mu,\ (b_i-y_i)h_i(x,y)\geq -\mu,\ i\in\mathcal{B}. \end{aligned} \tag{31}$$

3 Tools for MPEC Solution

3.1 Modeling Language Tools

MPEC's can be modeled in GAMS or AMPL using quite natural syntax. For example, in GAMS we would define the functions f, g and h with standard "equation" syntax, along with the bounds on the variable y. A full example is given in Appendix A. To define the actual MPEC model, the following statement is used:

```
model mpecmod / deff, defg, defh.y /;
```

Here it is assumed that the objective (1) is defined in the equation deff, the general constraints (2) are defined in defg and the function h is described in defh. The complementarity relationship is defined by the bounds on y and the orthogonality relationship shown in the model declaration using ".". More details for GAMS MPEC models can be found in [9], while similar formulations exist in AMPL [13].

In order to solve these models we propose to automatically reformulate the problems as nonlinear programs using a "convert" tool. We provide a solver "nlpec" that automatically calls the convert tool and reports the results in the original GAMS environment. The specific syntax used by a modeler follows:

```
option mpec=nlpec;
solve mpecmod using mpec minimizing obj;
```

3.2 The Convert Tool

Many solvers are developed that require a particular form of input, or have been implemented to interact with a particular modeling system. The convert tool is an evolving program whose purpose is to overcome these restrictive input formats.

Models that are formulated as a GAMS program are typically defined in terms of equations and variables that run over sets that are specified by the modeler. At compilation time, all of these equations are resolved into scalar equations and variables in order to be passed onto a particular solver. Sparse linear algebra and computational efficiency issues are considered, and a solver sees a clean model along with routines that specify derivative information.

At this scalar level it is very easy to convert the model into another input format. For example, the GAMS model can be written out as a scalar AMPL model (using the option "ampl"). Thus, the original GAMS model can be solved by any solver that accepts AMPL input. In a similar fashion, the BARON link to GAMS uses the convert tool to convert a GAMS model into BARON's required scalar input. Details of the other conversions possible can be found at `http://www.gamsworld.org/translate/`.

We modified the tool further to allow MPEC's to be reformulated as nonlinear programs at the scalar level. In fact, we currently have 23 different reformulations whereby the original MCP or MPEC is rewritten as a scalar GAMS nonlinear programming model (i.e. without any sets), but with the complementarity constraints rewritten using one of the constructs of the previous section.

The tool is somewhat more sophisticated than just a simple converter. The mapping between the original variables and the new scalar variables is maintained, so that a solution of the original problem can be recovered from the solution of the converted problem. In this way, we can easily develop new "black box" algorithms for MPEC's built simply by changing formulation, starting point and the sequence of parametric solves.

Table 1: Options for the solver NLPEC

Option	Value	Default	Description
er	integer	1	Reformulation to be generated.
initmu	real	0	Initial value of the parameter μ. A single solve of the nonlinear program is carried out for this value.
numsolves	integer	0	Number of extra solves carried out in a loop. This should be set in conjunction with the updatefac option.
updatefac	real	0.1	The factor that multiplies μ before each of the extra solves triggered by the numsolves option.
finalmu	real	-	Final value of the parameter μ. If specified, an extra solve is carried out with μ set to this value.
initslo	real	0	The lower bound for any artificials that are added.
initsup	real	inf	The upper bound for any artificials that are added.

3.3 Options and Parametric Solution

At the current time, there are 23 reformulations provided by the convert tool. The following table indicates the internal code that we use for each reformulation of the previous section. To specify using the reformulation (8) for example, the modeler uses the option "er = 1" in the file "nlpec.opt".

1	2	3	4	5	6	7	8	9	10	11	12
(8)	(9)	(28)	(12)	(16)	(29)	(30)	(31)	(17)	(18)	(13)	(26)
13	14	15	16	17	18	19	20	21	22	23	
(27)	(24)	(21)	(25)	(22)	(15)	(14)	(11)	(10)	(19)	(23)	

Details of all the current options of "nlpec" are given in Table 1.

It is assumed throughout all the testing that the modeler will have provided starting point values for the variables x and y. In all the formulations that add artificial variables (w and v) we initialize their values as follows.

$$
\begin{aligned}
w_{\mathcal{B}} &= \max\{0, h_{\mathcal{B}}(x,y)\} \\
v_{\mathcal{B}} &= \max\{0, -h_{\mathcal{B}}(x,y)\} \\
w_{\mathcal{L}} &= \max\{0, h_{\mathcal{L}}(x,y)\} \\
v_{\mathcal{U}} &= \max\{0, -h_{\mathcal{U}}(x,y)\}
\end{aligned}
$$

The tool provides the ability to change the constant chosen here as 0, and also allows an upper bound to be placed on the starting value for these artificial variables. Appropriate choices for these values is a topic for future research.

Another approach of interest when the complementarity constraints dominate the problem is is to solve the complementarity problem first to generate initial values for the nonlinear programming solver. This approach has been used successfully in [18] and is a technique that is easily available to a modeler using the tools outlined here.

In many cases, it is useful to generate a sequence of problems, parameterized by μ, that converge to the solution of the original problem as μ goes to zero. The convert tool generates nonlinear programs that involve the scalar μ. We have provided some extra options to the solver "nlpec" that allow updates to μ and multiple solves in a loop.

We have used a variety of option files for our computational tests and describe them now as examples of the flexibility of this scheme.

Option file 1 results in 7 nonlinear programs to be solved, the first with a value of $\mu = 0.01$, followed by 5 more solves with values of μ multiplied each time by 0.1. The final solve has $\mu = 0$. Option file 2 has six solves, the first with $\mu = 1.0$, the second with $\mu = 0.1$, each subsequent solve multiplying μ by 0.1. The resulting sequence of solves from option files 3, 4, 5 and 6 should now be clear.

3.4 Nonlinear Optimization Codes

There are a large number of NLP solvers available for the solution of the reformulated MPEC and MCP models. We have chosen a subset of these for computational testing. While all of the solvers chosen enjoy a strong reputation, they were also chosen to represent different algorithmic approaches.

For the MCP problems, we can choose to do no reformulation and solve the original model using the MCP solver PATH [8, 15, 29, 35]. PATH iminitmu = 0.01

numsolves = 5

finalmu = 0

(a) Option file 1

initmu = 1

numsolves = 5

(b) Option file 2

initmu = 1

numsolves = 3

(c) Option file 3

initmu = 1

numsolves = 4

(d) Option file 4

initmu = 1

numsolves = 5

finalmu = 0

(e) Option file 5

initmu = 0.2

finalmu = 0.1

(f) Option file 6

Figure 1: Option files used for computational results

plements a generalization of Newton's method with linesearch applied to an equivalent formulation of a complementarity problem as a nonsmooth system of equations. The subproblems are solved using a variant of Lemke's method, a pivotal method for LCP. The pathsearch is controlled by the Fisher merit function and resorts to a gradient step of that function if the subproblem solution fails to give appropriate descent. There are some safeguards included that help when singularities are encountered. Some computational enhancements include preprocessing (logical inferences to reduce the size and complexity of the problem), a crash procedure to find a good starting basis and various strategies to overcome degeneracy.

The NLP solver CONOPT [11] is a feasible path solver based on the proven GRG method, especially suitable for highly nonlinear models. It also includes extensions for phase 0, linear mode iterations, a sequential linear programming component, and more recently the use of Hessian information. MINOS [31] solves NLPs with linear constraints using a quasi-Newton, reduced-gradient algorithm. A projected Lagrangian algorithm with quadratic penalty function is used for the nonlinear constraints. SNOPT [20] applies a sparse sequential quadratic programming (SQP) method , using limited memory quasi-Newton approximations to the Hessian of the Lagrangian. The merit function for steplength control is an augmented Lagrangian. BARON [42, 45] is a computational system for solving nonconvex optimization problems to global optimality. This *Branch And Reduce Optimization Navigator* combines constraint propagation, interval analysis, and duality in its reduce arsenal with enhanced branch and bound concepts.

While the solvers mentioned above all run locally, it is also possible to solve models on a remote machine. Remote solution is made possible via the Kestrel interface [10] to NEOS [7], the Network Enabled Optimization Server. Using Kestrel and NEOS, we have access to many more NLP solvers, in particular the interior point (or barrier) methods KNITRO and LOQO. KNITRO [4] is a trust region method which uses sequential quadratic programming methodology to treat the barrier sub-problems. LOQO [49] is a line search algorithm that has much in common with interior algorithms for linear and convex quadratic programming. It is interesting to note that both KNITRO and LOQO use AMPL interfaces; the Kestrel interface takes advantage of the convert tool described above to produce an AMPL form of the model in question.

4 Computational Results

4.1 Feasibility Problems

We consider a set of 11 test problems that have historically caused difficulties to MCP solvers. All of these are fairly small models; their sizes are given in Table 2.

There are several models that have their origins in the economics literature. The general equilibrium model for Cameroon [6] has been formulated in a number of ways, here in CAMMCP as an MCP. The model DUOPOLY is a dynamic oligopoly model described in [27, 28]. An electricity flow equilibrium model ELECTRIC, a simple exchange model SIMPLE-EX, a consumption model with spillover effects SPILLMCP were all provided in [41]. A standard n-player Nash equilibrium problem [48] is called GAMES. The von Thünen land use model [44, 14] is implemented in PGVON105, while the Shubik-Quint general equilibrium model with money [43] is used as the basis for SHUBIK. [36, 37] provides a series of complementarity models used for shadow pricing in red-blue tactical decisions, one of which is called FORCEDSA.

Other examples of complementarity arise in engineering [17]. The remaining two models are examples of these, including a lubrication model

Table 2: MCP models

Name	Variables	Nonzeros	Density (%)
CAMMCP	242	1287	2.20
DUOPOLY	63	252	6.35
EHL_KOST	101	10200	99.99
ELECTRIC	158	539	2.16
FORCEDSA	186	440	1.27
GAMES	16	140	54.69
LINCONT	419	23207	13.22
PGVON105	105	588	5.33
SHUBIK	33	136	12.49
SIMPLE-EX	17	158	54.67
SPILLMCP	110	455	3.76

EHL_KOST detailed in [25] and a friction-contact problem called LINCONT described in [33].

Table 3 gives an indication of which solver/reformulation combinations are most effective in solving the set of MCP models chosen. Effectiveness is measured here only in terms of robustness. In all these feasibility cases, we set up a dummy objective function of 0. Each solver/reformulation combination was tried without options and with one of the option files in Figure 3.3. The results reported are for the more successful of these runs.

Table 3: MCP: Successful solves; column headings refer to the reformulation equation number

Solver	MCP	ER1 (8)	ER2 (9)	ER9 (17)	ER21 (10)	ER23 (23)
PATH	9					
BARON		5	4	5	7	2
CONOPT		2	3	3	3	1
MINOS		5	6	6	5	3
SNOPT		8	5	9	6	3
FILTER		4	5	6	3	1
KNITRO		1	6	6	0	0
LOQO		5	3	4	5	1

Several points are clear from these results. Firstly, as should be expected, a specialized complementarity solver is more robust for solving these feasibility problems, but even on these difficult problems, several nonlinear programming algorithms perform well on certain reformulations. Secondly, somewhat unexpectedly, the reformulations using the Fischer function (ER23) seem to cause the nonlinear programming solvers distinct difficulty for these models. Finally, while Table 3 does not exhibit this fact, for the cases where PATH fails, we can solve the problem by one or more of these reformulations. From a modeler's perspective this is very useful, since during the development cycle many of the deficiencies of the model are best identified from a solution. Unfortunately, the models that are typically hardest to solve are those with errors in their formulation.

Comparison of solution times is quite important, but can easily be misleading. In the case of the solvers tested via the remote Kestrel interface, it is difficult to say for certain what machines the solvers ran on. This and other factors make it difficult to use solution times for any Kestrel solvers in a meaningful way. For these reasons we have not included the results in the above table. However, timing comparisons can be found at `http://www.gamsworld.org/mpec/nlpectests`. These show that in general the nonlinear programming reformulations are slower than the specialized complementarity solvers. In order not to repeat results that are given elsewhere, we note that for large scale problems, PATH is typically very effective and fast. Detailed results can be found in [30] for example.

It is also clear that by adjusting certain options (for example feasibility or optimality tolerances) to each of the solvers, a different set of models could have been solved. We limited our computational testing to the default settings of each solver.

4.2 Small Optimization Problems

There is a considerable literature on multiplicity of solutions to complementarity problems, arising both from applications of Nash equilibria to crack propagation in structural mechanics. Determining which of these multiple solutions satisfies some "optimality criteria" is a problem of much practical interest.

Since in many cases the complementary solutions are isolated, nonlinear programming techniques that find local minimizers are extremely prone to failure, in that while they may find feasible points, the value of the objective could be arbitrarily poor. In order to solve these problems reliably, one of two approaches is needed. As usual, the first (and most generally applicable) approach requires the modeler to provide a starting point that is close to the solution required. The second approach is to use a nonlinear programming code that is designed to find global solutions. Due to the enormous difficulties of these problem classes, the second approach is currently severely limited in problem size, but we will outline its use on two small examples to exhibit the potential of further research in this area.

The first problem comes from the mathematical programming literature

[24] and is a four variable nonlinear complementarity problem with exactly two isolated solutions, namely $(1.2247, 0, 0, 0.5)$ and $(1, 0, 3, 0)$. We set up two MPEC's, the first KOJSHIN3 minimizes x_3 while the second KOJSHIN4 minimizes x_4. Both of these problems have a feasible set consisting of two points, and each has an optimal value of 0. As is to be expected, the nonlinear programming algorithms applied to the formulations outlined above either fail to find a feasible point, or have the tendency to terminate at the non-optimal solution.

However, applying the BARON solver (a global method) to reformulation 1 with $\mu = 0$ solves both problems to optimality in under 0.2 secs. In fact, all the feasible points that lie in some compact set can be enumerated for this example if desired using the "numsol -1" option of BARON. There are some potential difficulties in discriminating among solutions that are subject to rounding error, but in general all solutions will be found.

The second example of this nature is a Nash equilibrium example given in [23]. In this example, three distinct equilibria are known; the models KEHOE1, KEHOE2 and KEHOE3 have objectives set up that respectively minimize, maximize the price variables, or find a solution closest to the starting point. In order to enumerate the distinct equilibria, we found it easiest to use BARON on a modification of KEHOE1; we first found the equilibrium that minimized the sum of the prices, then added an extra constraint on the price sum to exclude that solution. Thus, with three solves under BARON, we were able to enumerate all the equilibria, without special knowledge of starting points. A fourth solve confirmed that no more equilibria existed within the (large) compact set used for the problem variables. The example file given in Appendix A was used for this purpose. Note that the complementarity problem is defined using the "." notation and that the income definitions can be treated as general nonlinear constraints. The restriction equation removes any solutions for which the sum of the prices is less than 3.64.

These techniques are unlikely to work for large scale problems. In these cases, it is likely that multistart or sampling methods will be needed to improve the likelihood of generating a global solution. Some promising approaches that can be used from within GAMS are given in [47, 34].

4.3 Feasibility tests

Computational tests of the sort discussed in this paper underscore the need for a separate utility to verify the correctness of the solutions obtained and create uniform reports of their accuracy. The GAMS "solver" Examiner is such a utility. GAMS/Examiner is currently under development and was extended to allow checks for feasibility of the MPEC solutions. It performs three separate checks on MPEC models.

The first check is for feasibility in the primal variables x and y with respect to the variable bounds. The error reported is the maximum violation found. GAMS solvers typically maintain primal variable feasibility with zero tolerance, so there is usually nothing to report here.

The second check is for feasibility with respect to the NLP constraints (2) and the equilibrium constraints (3). For the NLP constraints (2), the residual error in the ith row is computed in the obvious way. For the equilibrium constraints (3), however, we only assign a nonzero residual to row i if:

1. the matching variable is in $\mathcal{L}$ and h_i is negative, or
2. the matching variable is in $\mathcal{U}$ and h_i is positive, or
3. the matching variable is in $\mathcal{F}$ and h_i is nonzero.

Note that if the matching variable is in $\mathcal{B}$ the residual is set to zero. The error reported is the maximum residual taken over both sets of constraints.

The third check is for complementarity; this check involves only the equilibrium constraints (3) and the variables y. Again, the error reported is the maximum violation found, taken now over all the equilibrium constraints. For each such constraint, we compute errors with respect to the lower and upper variable bounds; the maximum of these two is the residual error r_i. We describe this computation below.

1. $c = \max(0, a_i - y_i)$, $d = \min(1, \max(0, y_i - a_i))$.
2. $r_i = \max(c, d \max(h_i, 0))$
3. $c = \max(0, y_i - b_i)$, $d = \min(1, \max(0, b_i - y_i))$.
4. $r_i = \max(r_i, \max(c, d \max(-h_i, 0)))$

Unless the variable y is outside of its bounds (a *very* unusual case for any of the NLP solvers tested), the deviation c will always be zero, and the effect is to assign zero error for the lower bound if h_i is negative, and otherwise to scale the error h_i by $\min(y_i - a_i, 1)$. Similarly, we assign zero error for the upper bound if h_i is positive, and otherwise scale the error by $\min(b_i - y_i, 1)$. This definition of the residual error is taken from the GAMS MCP solvers, where it has proven to be very useful in identifying the constraints of interest in unsolvable, poorly formulated, and partially completed models. For the purposes of this paper we declare a solution to be feasible if the maximum residual is less than 10^{-5}.

4.4 Larger Optimization Problems

Techniques for solving larger problems cannot rely on the sampling techniques or enumerative/branch and reduce techniques that work well on small problems. Instead, currently, much more emphasis is placed on the modeler to provide problems for which the complementarity problems have nice properties (ie stability under perturbations, local uniqueness, etc), and for which good starting points are known or can be effectively generated.

We have taken as our test bed for MPEC's the MPEClib problems. Details on problem size and characteristics can be found in Appendix B. MPEClib currently contains 92 problems. For each of these problems we attempted solution with each of 40 different reformulation / option file combinations and each of the 4 NLP solvers BARON, CONOPT, MINOS, and SNOPT, for a total of 14,720 solves.

The solution results for the different formulations we outlined in Section 3 are given in Table 4. For brevity, we only report the percentage of times that the solvers terminated in less than 10 seconds of CPU time with a feasible solution of the MPEC. If a particular option file significantly outperforms an alternate, we have not reported the poorer results. We have not reported any results for er14 and er15 since the reformulations are nonsmooth. The reformulations er16 and er17 perform poorly due to evaluation errors that occur in the exponential. The row er*.any reports the percentage of successes of each solver on any reformulation with any option file. The column anysolv indicates the percentage of models solved with each reformulation/option combination and at least one of the 4 solvers.

Table 5 show how well the objectives were minimized compared to the best solution that any solver found over all reformulations. We believe this

Table 4: Percentage of successful solves resulting in feasible solutions of MPEC using the NLP reformulations of Section 3 with GAMS solver links

	CONOPT	MINOS	SNOPT	BARON	anysolv
er1 .0	73	39	76	46	85
er1 .1	82	58	78	50	90
er2 .0	72	40	80	46	88
er2 .1	75	73	71	70	90
er3 .1	82	58	79	51	90
er4 .0	71	64	71	66	87
er4 .1	72	84	65	75	89
er5 .0	61	60	70	64	86
er5 .1	68	52	55	71	87
er6 .0	58	59	68	63	85
er7 .0	53	64	75	67	86
er8 .0	62	37	60	53	84
er9 .3	63	54	63	47	79
er10 .4	51	48	54	34	72
er11 .1	79	49	65	59	88
er12 .0	41	60	73	63	85
er12 .5	72	58	64	66	89
er13 .0	37	60	68	64	85
er13 .5	71	61	68	66	89
er16 .0	4	8	8	13	15
er17 .0	9	9	8	16	17
er18 .0	59	67	77	72	89
er19 .0	70	32	76	57	85
er20 .0	73	66	76	72	89
er20 .1	78	86	71	79	91
er21 .0	71	64	74	70	87
er21 .1	76	83	67	77	89
er21 .5	76	84	64	77	89
er22 .5	71	47	63	43	90
er23 .5	75	67	63	64	90
er*.any	96	91	91	85	96

table shows that the approaches postulated here are extremely promising and allow both small and medium scale MPEC's to be solved with a variety of algorithms. More details on our testing strategy, coupled with more detailed results of all the tests we performed are available at
`http://www.gamsworld.org/mpec/nlpectests`.

It is clear that on this test set, a variety of the reformulations are very effective ways to find both feasible solutions and good locally optimal solutions of the MPEC. In particular, it seems that (ordered by increasing solution times) er3 (28), er21 (10), er1 (8), er22 (19) and er13 (27) (coupled with an appropriate option file) are very promising solution approaches.

In a recent paper [46], a suite of MPEC examples were described along with a variety of techniques to solve them. The results reported there seem to broadly agree with the results described herein. In particular, for large, hard examples, the formulations involving the Fischer function (especially formulation er22) were found to be most effective in terms of solution time, and objective value.

5 Conclusions

This paper has described the notion of a mathematical program with equilibrium constraints and given several reformulations of such problems as standard nonlinear programming problems. It has outlined several tools to facilitate the automatic generation of these formulations from a GAMS specification of the original problem.

A number of algorithms have been applied to solve a suite of MPEC models that have been collected from a variety of application domains. All the examples cited in this paper are available from the gamsworld website at `http://www.gamsworld.org/mpec/`.

Several conclusions can be drawn. Firstly, the ability to formulate problems with complementarity constraints as nonlinear programs enhances the ability of a modeler to use complementarity as a technique for answering important economic questions. We have demonstrated both improvements in overall robustness, and several new techniques for exploring more thoroughly the solution space. Secondly, tools for reformulation provide a variety of solution techniques for MPEC's. While this paper does not show definitively what solver or which formulation is to be preferred, it does give a modeler a suite of tools that allow him/her to generate solutions of these problems.

Table 5: Percentage of solves resulting in solutions of MPEC (within 1% of the best found) using GAMS solver links

	CONOPT	MINOS	SNOPT	BARON	anysolv
er1 .0	43	20	49	41	63
er1 .1	64	43	61	41	80
er2 .0	43	18	45	41	63
er2 .1	47	55	51	55	76
er3 .1	64	43	60	42	82
er4 .0	39	35	39	52	66
er4 .1	41	53	40	58	72
er5 .0	26	23	36	47	62
er5 .1	43	38	40	59	76
er6 .0	24	22	35	46	61
er7 .0	26	21	35	49	62
er8 .0	35	22	32	37	61
er9 .3	45	42	41	41	62
er10 .4	37	37	42	32	57
er11 .1	59	36	40	40	71
er12 .0	26	35	46	32	57
er12 .5	61	33	49	46	78
er13 .0	22	35	41	33	57
er13 .5	61	36	53	45	77
er16 .0	4	7	7	8	10
er17 .0	8	7	7	10	10
er18 .0	30	25	38	53	65
er19 .0	36	21	40	34	60
er20 .0	43	38	45	55	71
er20 .1	49	57	45	64	76
er21 .0	41	35	42	54	68
er21 .5	54	60	49	63	80
er22 .5	64	28	41	37	78
er23 .5	64	48	52	51	75
er*.any	95	75	85	83	96

In particular, a modeler is able to write down an explicit formulation of the problem as an MPEC, and use these tools to generate the required equations to treat complementarity, as opposed to having to generate different model descriptions for each specific way of processing complementarity. Thirdly, the ability to reliably solve large and complex models with complementarity constraints should enable applications (such as optimal tariff determination) to be processed by modelers more readily in the very near future.

It is hoped that the techniques outlined here will provide a basis for future application work in this area, and will generate more of the interactions between modelers and algorithmic developers that have proven so successful in the complementarity field. One area of particular interest in applying MPEC models is the choice of optimal tariffs. There is a need for large scale algorithms in this case due to the size and detail of the underlying datasets. Such problems are regarded as extremely difficult.

Acknowledgments

The authors are grateful to Todd Munson, Nick Sahinidis and Sven Leyyfer for their advice and help with regard to algorithmic aspects. Both Tom Rutherford and Francis Tin-Loi have provided invaluable test problems and insight into specific applications without which this paper would not have been possible.

References

[1] K. Arrow and G. Debreu. Existence of equilibrium for a competitive economy. *Econometrica*, 22:265–290, 1954.

[2] S. C. Billups. *Algorithms for Complementarity Problems and Generalized Equations*. PhD thesis, University of Wisconsin, Madison, Wisconsin, August 1995.

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

[4] R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[5] Chunhui Chen and O. L. Mangasarian. A class of smoothing functions for nonlinear and mixed complementarity problems. *Computational Optimization and Applications*, 5:97–138, 1996.

[6] T. Condon, H. Dahl, and S. Devarajan. Implementing a computable general equilibrium model on GAMS – the Cameroon model. DRD Discussion Paper 290, 1987. The World Bank, Washington, DC.

[7] J. Czyzyk, M. P. Mesnier, and J. J. Moré. The NEOS server. *IEEE Journal on Computational Science and Engineering*, 5:68–75, 1998.

[8] 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.

[9] S. P. Dirkse and M. C. Ferris. Modeling and solution environments for MPEC: GAMS & MATLAB. In M. Fukushima and L. Qi, editors, *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, pages 127–148. Kluwer Academic Publishers, 1999.

[10] Elizabeth D. Dolan and Todd S. Munson. The Kestrel interface to the NEOS Server. Technical Memorandum ANL/MCS-TM-248, Argonne National Laboratory, Argonne, Illinois, 2001.

[11] A. Drud. CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31:153–191, 1985.

[12] F. Facchinei, H. Jiang, and L. Qi. A smoothing method for mathematical programs with equilibrium constraints. *Mathematical Programming*, 85:107–134, 1999.

[13] M. C. Ferris, R. Fourer, and D. M. Gay. Expressing complementarity problems and communicating them to solvers. *SIAM Journal on Optimization*, 9:991–1009, 1999.

[14] M. C. Ferris and T. S. Munson. Case studies in complementarity: Improving model formulation. In M. Théra and R. Tichatschke, editors, *Ill–Posed Variational Problems and Regularization Techniques*, number 477 in Lecture Notes in Economics and Mathematical Systems, pages 79–98. Springer Verlag, Berlin, 1999.

[15] M. C. Ferris and T. S. Munson. Complementarity problems in GAMS and the PATH solver. *Journal of Economic Dynamics and Control*, 24:165–188, 2000.

[16] M. C. Ferris and T. S. Munson. Preprocessing complementarity problems. In M. C. Ferris, O.L. Mangasarian, and J. S. Pang, editors, *Complementarity: Applications, Algorithms and Extensions*, volume 50 of *Applied Optimization*, pages 143–164, Dordrecht, The Netherlands, 2001. Kluwer Academic Publishers.

[17] M. C. Ferris and J. S. Pang. Engineering and economic applications of complementarity problems. *SIAM Review*, 39:669–713, 1997.

[18] M. C. Ferris and F. Tin-Loi. Limit analysis of frictional block assemblies as a mathematical program with complementarity constraints. *International Journal of Mechanical Sciences*, 43:209–224, 2001.

[19] A. Fischer. A special Newton-type optimization method. *Optimization*, 24:269–284, 1992.

[20] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 28, 2002.

[21] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. A practical anti-cycling procedure for linearly constrained optimization. *Mathematical Programming*, 45:437–474, 1989.

[22] P. T. Harker and J. S. Pang. Existence of optimal solutions to mathematical programs with equilibrium constraints. *Operations Research Letters*, 7:61–64, 1988.

[23] T. Kehoe. A numerical investigation of the multiplicity of equilibria. *Mathematical Programming Study*, 23:240–258, 1985.

[24] M. Kojima and S. Shindo. Extensions of Newton and quasi-Newton methods to systems of PC^1 equations. *Journal of Operations Research Society of Japan*, 29:352–374, 1986.

[25] M. M. Kostreva. Elasto-hydrodynamic lubrication: A non-linear complementarity problem. *International Journal for Numerical Methods in Fluids*, 4:377–397, 1984.

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

[27] E. Maskin and J. Tirole. A theory of dynamic oligopoly, I: Overview and quantity competition with large fixed costs. *Econometrica*, 56:549–569, 1988.

[28] E. Maskin and J. Tirole. A theory of dynamic oligopoly, II: Price competition, kinked demand curves, and edgeworth cycles. *Econometrica*, 56:571–579, 1988.

[29] L. Mathiesen. Computation of economic equilibria by a sequence of linear complementarity problems. *Mathematical Programming Study*, 23:144–162, 1985.

[30] T. S. Munson, F. Facchinei, M. C. Ferris, A. Fischer, and C. Kanzow. The semismooth algorithm for large scale complementarity problems. *INFORMS Journal on Computing*, 13:294–311, 2001.

[31] B. A. Murtagh and M. A. Saunders. MINOS 5.0 user's guide. Technical Report SOL 83.20, Stanford University, Stanford, California, 1983.

[32] J. Outrata, M. Kočvara, and J. Zowe. *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[33] J. S. Pang and J. C. Trinkle. Complementarity formulations and existence of solutions of multi-rigid-body contact problems with Coulomb friction. *Mathematical Programming*, 73:199–226, 1996.

[34] J. P. Pinter. *Global Optimization in Action*. Kluwer Academic Publishers, Dordrecht, 1996.

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

[36] S. M. Robinson. Shadow prices for measures of effectiveness I: Linear model. *Operations Research*, 41:518–535, 1993.

[37] S. M. Robinson. Shadow prices for measures of effectiveness II: General model. *Operations Research*, 41:536–548, 1993.

[38] T. F. Rutherford. MILES: A mixed inequality and nonlinear equation solver. Working Paper, Department of Economics, University of Colorado, Boulder, 1993.

[39] T. F. Rutherford. Extensions of GAMS for complementarity problems arising in applied economic analysis. *Journal of Economic Dynamics and Control*, 19:1299–1324, 1995.

[40] T. F. Rutherford. Applied general equilibrium modeling with MPSGE as a GAMS subsystem: An overview of the modeling framework and syntax. *Computational Economics*, 14:1–46, 1999.

[41] T. F. Rutherford. Private communication, January 2002.

[42] N. V. Sahinidis. BARON: A General Purpose Global Optimization Software Package. *Journal of Global Optimization*, 8:201–205, 1996.

[43] M. Shubik. *Game Theory, Money and the Price System: The Selected Essays of Martin Shubik*, volume 2. Edward Elgar, Cheltenham, England, 1999.

[44] B. H. Stevens. Location theory and programming models: The von thünen case. *Papers of the Regional Science Association*, 21:19–34, 1968.

[45] M. Tawarmalani and N. V. Sahinidis. Global Optimization of Mixed Integer Nonlinear Programs: A Theoretical and Computational Study. *Mathematical Programming*, (submitted 1999).

[46] F. Tin-Loi and N.S. Que. Nonlinear programming approaches for an inverse problem in quasibrittle fracture. *International Journal of Mechanical Sciences*, 2002.

[47] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, and R. Marti. A multistart scatter search heuristic for smooth NLP and MINLP problems. Technical report, University of Texas at Austin, 2002.

[48] G. van der Laan, A.J.J. Talman, and L. Van der Heyden. Simplicial variable dimension algorithms for solving the nonlinear complementarity problem on a product of unit simplicies using a general labelling. *Mathematics of Operations Research*, 12:377–397, 1987.

[49] R. J. Vanderbei and D. F. Shanno. An interior–point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

[50] S. J. Wright. *Primal–Dual Interior–Point Methods*. SIAM, Philadelphia, Pennsylvania, 1997.

A Example of GAMS MPEC syntax

```
$TITLE  Multiple equilibria in a simple GE model

SET     G       GOODS /G1*G4/
        S       SECTORS /S1,S2/
        C       CONSUMERS /C1*C4/;

TABLE E(G,C) Factor endowments
                C1      C2      C3      C4
        G1      5
        G2              5
        G3                      40
        G4                              40

TABLE ALPHA(G,C)  Budget shares
                C1      C2      C3      C4
        G1      0.52    0.86    0.50    0.06
        G2      0.40    0.10    0.20    0.25
        G3      0.04    0.02    0.2975  0.0025
        G4      0.04    0.02    0.0025  0.6875

TABLE A(G,S)  Activity analysis matrix
                S1      S2
        G1       6      -1
        G2      -1       3
        G3      -4      -1
        G4      -1      -1

POSITIVE
VARIABLES       Y(s)    Activity level
                P(g)    Relative price;
VARIABLES       OBJ
                H(c)    Income level;

EQUATIONS       PROFIT, MARKET, INCOME, OBJDEF;

OBJDEF..        OBJ =E= SUM(G,P(G));
```

```
* The following constraint removes one equilibrium
RESTRICT..      SUM(G,P(G)) =G= 3.64;

PROFIT(S)..     SUM(G, -A(G,S)*P(G)) =G= 0;

MARKET(G)..     SUM(C, E(G,C)) + SUM(S, A(G,S)*Y(S))
                =G= SUM(C, ALPHA(G,C) * H(C)/P(G));

INCOME(C)..     H(C) =E= SUM(G, P(G) * E(G,C));

P.L(G) = 1;
* Protect against domain violations
P.LO(G) = 1e-4;
* Fix a numeraire
P.FX("G1") = 1;

MODEL KEHOE /OBJDEF, PROFIT.Y, MARKET.P, INCOME/;

Solve KEHOE using MPEC min obj;
```

B Model Statistics for Test Problems

Name	m	n	nz	nlnz
AAMPEC_1	70	72	430	247
AAMPEC_2	70	72	430	247
AAMPEC_3	70	72	430	247
AAMPEC_4	70	72	430	247
AAMPEC_5	70	72	430	247
AAMPEC_6	70	72	430	247
BARD1	5	6	14	2
BARD2	10	13	33	4
BARD3	6	7	19	5
BARTRUSS3_0	29	36	96	38
BARTRUSS3_1	29	36	96	38
BARTRUSS3_2	29	36	96	38
BARTRUSS3_3	27	34	90	38
BARTRUSS3_4	27	34	90	38
BARTRUSS3_5	27	34	90	38
DEMPE	4	5	9	5
DEMPE2	3	4	7	5
DESILVA	5	7	13	10
EX9_1_1M	8	9	23	0
EX9_1_2M	6	7	14	0
EX9_1_3M	7	9	23	0
EX9_1_4M	5	6	12	0
FINDA10L	229	211	877	200
FINDA10S	229	211	877	200
FINDA10T	229	211	877	200
FINDA15L	229	211	877	200
FINDA15S	229	211	877	200
FINDA15T	229	211	877	200
FINDA30S	229	211	877	200
FINDA30T	229	211	877	200
FINDA35L	229	211	877	200
FINDA35S	229	211	877	200
FINDA35T	229	211	877	200
FINDB10L	203	198	812	200

FINDB10S	203	198	812	200
FINDB10T	203	198	812	200
FINDB15L	203	198	812	200
FINDB15S	203	198	812	200
FINDB15T	203	198	812	200
FINDB30L	203	198	812	200
FINDB30S	203	198	812	200
FINDB30T	203	198	812	200
FINDB35L	203	198	812	200
FINDB35S	203	198	812	200
FINDB35T	203	198	812	200
FINDC10L	187	190	772	200
FINDC10S	187	190	772	200
FINDC10T	187	190	772	200
FINDC15L	187	190	772	200
FINDC15S	187	190	772	200
FINDC15T	187	190	772	200
FINDC30L	187	190	772	200
FINDC30S	187	190	772	200
FINDC30T	187	190	772	200
FINDC35L	187	190	772	200
FINDC35S	187	190	772	200
FINDC35T	187	190	772	200
FJQ1	7	8	21	10
FRICTIONALBLOCK_1	682	682	2690	0
FRICTIONALBLOCK_2	1154	1154	4618	0
FRICTIONALBLOCK_3	854	854	3338	0
FRICTIONALBLOCK_4	979	979	3776	0
FRICTIONALBLOCK_5	1025	1025	3924	0
FRICTIONALBLOCK_6	2855	2855	11364	0
GAUVIN	3	4	8	2
HQ1	2	3	5	2
KEHOE1	11	11	49	20
KEHOE2	11	11	49	20
KEHOE3	11	11	49	24
KOJSHIN3	5	5	18	8
KOJSHIN4	5	5	18	8
MSS	5	6	26	25

NAPPI_A	98	116	330	88
NAPPI_B	98	116	330	88
NAPPI_C	98	116	330	88
NAPPI_D	98	116	330	88
OUTRATA31	5	6	17	10
OUTRATA32	5	6	18	11
OUTRATA33	5	6	18	11
OUTRATA34	5	6	20	13
OZ3	6	7	19	0
QVI	3	5	9	4
THREE	4	3	8	6
TINLOI	101	105	10201	100
TINQUE_DHS2	4834	4805	65315	13024
TINQUE_DNS2	4834	4805	65315	13024
TINQUE_MIS2	4066	4037	48803	10912
TINQUE_PSS2	4578	4549	59555	12320
TINQUE_SWS2	4578	4549	59555	12320
TINQUE_SWS3	5699	5671	67397	17920
TOLLMPEC	2377	2380	10488	1754