EMP

Contents

1 Introduction

EMP (Extended Mathematical Programming) is not a solver but an (experimental) framework for automated mathematic programming reformulations. The idea behind EMP is that new upcoming types of models which currently cannot be solved reliably are reformulated into models of established math programming classes in order to use mature solver technology. At this stage, EMP supports the modeling of Bilevel Programs, Variational Inequalities, Disjunctive Programs, Extended Nonlinear Programs and Embedded Complementarity Systems, but additional features are being added regularly.

Extended mathematical programs are collections of functions and variables joined together using specific optimization and complementarity primitives. EMP annotates the existing relationships within a model to facilitate higher level structure identification. A specific implementation of this framework is outlined that reformulates the original GAMS model automatically using directives contained in an "empinfo" file into an equivalent model that can be solved using existing GAMS solvers.

The reformulation is done by the solver JAMS which currently is the only solver that is capable of handling EMP models. Examples showing how to use the EMP framework and the solver JAMS are made available through the GAMS EMP Library which is included in the GAMS Distribution. In order to generate a copy of and EMPLIB model, one can use the library facility of the GAMS IDE, or execute the command line directive "emplib modelname" where modelname is the (stem of the) file containing the model.

EMP has been developed jointly by Michael Ferris of UW-Madison, Ignacio Grossmann of Carnegie Mellon University and GAMS Development Corporation. EMP and JAMS come free of charge with any licensed GAMS system but require a subsolver to solve the generated models.

2 JAMS: a reformulation tool

EMP models are currently processed by the JAMS solver. The solver JAMS creates a scalar version of the given GAMS model. This scalar version of the model is then solved by an appropriate subsolver. By default, there are no reformulations carried out, so the model generated is simply a GAMS scalar form of the model the actual subsolver will process. The subsolver used is by default the currently specified solver for the given model type.

2.1 The JAMS Option File

As with any GAMS solver, JAMS has an option file, typically called “jams.opt”. A JAMS option “subsolver” is available to change the subsolver used for the reformulated model, along with an option to utilize a subsolver option file (“subsolveropt”).

The actual scalar version of the model can also be seen by the modeler using the option “filename”. For example, the option file

```
subsolver path
subsolveropt 1
filename mcpmod.gms
```

when applied to an EMP model that is a complementarity problem will create a file called “mcpmod.gms” in the current directory and solve that model using the solver “path” utilizing any options for “path” that are specified in “path.opt”. The scalarized model is not particularly useful to look at since all of the original variables have been renamed into a scalar form. The mapping between original variables and the ones used in the scalar version of the model is given in a dictionary file that can also be seen by the modeler using the option

```
dict dict.txt
```

After the scalar version of the model is solved, the solution values are mapped back into the original namespace and returned to the modeler as usual in the listing file. The JAMS option “margtol” allows the modeler to suppress reporting marginals that have (absolute) values smaller than this tolerance.

Obviously, all of the above functionality is not of much value: the key part of JAMS is to interpret additional directives to take the original model and produce a *reformulated* scalar model. This is carried out using an “empinfo” file. The syntax and use of this file is the content of the remaining sections of this document.

MCF: Terminate and EMPFileName

2.2 The EMP Info File

MCF: Details on how to write this, etc. Maybe at end of document?

3 Forming Optimality Conditions: NLP2MCP

The first nontrivial use of the JAMS solver is to automatically generate the first order conditions of a linear or nonlinear program; essentially we *reformulate* the optimization problem as a mixed complementarity problem (MCP). The empinfo file to do this simply contains the following line:

```
modeltype mcp
```

Behind the scenes, JAMS forms the Lagrangian of the nonlinear program and then forms its Karush-Kuhn-Tucker optimality conditions. To be clear, given the original nonlinear program:

$$\min_x f(x) \text{ s.t. } g(x) \leq 0,\ h(x) = 0. \tag{1.1}$$

the Lagrangian is:

$$\mathcal{L}(x, \lambda, \mu) = f(x) - \lambda^T g(x) - \mu^T h(x).$$

The first order conditions are the following MCP:

$$\begin{array}{lcl} 0 = \nabla_x \mathcal{L}(x, \lambda, \mu) & \perp & x \text{ free} \\ 0 \geq -\nabla_\lambda \mathcal{L}(x, \lambda, \mu) & \perp & \lambda \leq 0 \\ 0 = -\nabla_\mu \mathcal{L}(x, \lambda, \mu) & \perp & \mu \text{ free} \end{array}$$

A specific example is:

$$\begin{array}{ll} \min\limits_{x,y,z} & -3x + y \\ \text{s.t.} & x + y \leq 1,\ x + y - z = 2,\ x, y \geq 0 \end{array}$$

which is found in the EMPLIB model nlp2mcp:

```
1  variables f,z; positive variables x,y;
2  equations g, h, defobj;
3
4  g.. x + y =l= 1;
5  h.. x + y - z =e= 2;
6  defobj.. f =e= -3*x + y;
7
8  model comp / defobj, g, h /;
9
10 file info / '%emp.info%' /;
11 putclose info / 'modeltype mcp';
12
13 solve comp using emp minimizing f;
```

Lines 10-11 write out the default "empinfo" file whose location is provided in the system string `%emp.info%`. Armed with this additional information, the EMP tool automatically creates the following MCP:

$$\begin{array}{lcl} 0 \leq -3 - \lambda - \mu & \perp & x \geq 0 \\ 0 \leq 1 - \lambda - \mu & \perp & y \geq 0 \\ 0 = \mu & \perp & z \text{ free} \\ 0 \geq x + y - 1 & \perp & \lambda \leq 0 \\ 0 = x + y - z - 2 & \perp & \mu \text{ free.} \end{array}$$

MCF: shouldn't keepobj (and objvarname) be an empinfo file directive? MCF: objvarname calls the objvar whatever you assign in the scalar model?

4 Soft Constraints

In many cases, we wish to relax certain constraints in a model during solution (to help identify feasibility issues for example). As an example, consider the problem

$$\begin{array}{rl} \min\limits_{x_1,x_2,x_3} & \exp(x_1) \\ \text{s.t.} & \log(x_1) = 1, \\ & x_2^2 \leq 2, \\ & x_1/x_2 = \log(x_3), \\ & 3x_1 + x_2 \leq 5, x_1 \geq 0, x_2 \geq 0. \end{array}$$

which can be formulated in GAMS as

```
1  $title simple example of ENLP
2
3  variables obj,x1,x2,x3;
4  equations f0,f1,f2,f3,f4;
5
6  f0.. obj =e= exp(x1);
7  f1.. log(x1) =e= 1;
8  f2.. sqr(x2) =g= 2;
9  f3.. x1/x2 =e= log(x3);
10 f4.. 3*x1 + x2 =l= 5;
11
12 x1.lo = 0; x2.lo = 0;
13
14 model enlpemp /all/;
15 x1.l = 1; x2.l = 1; x3.l = 1;
16 solve enlpemp using nlp min obj;
```

4.1 Reformulation as a classical NLP

Soft constraints allow us to treat certain equations in the model as "soft" by removing the constraints and adding a penalty term to the objective function. Explicitly, we replace the above problem by:

$$\begin{aligned}\min_{x_1,x_2,x_3}\ & \exp(x_1) + 5\left\|\log(x_1) - 1\right\|^2 + 2\max(x_2^2 - 2, 0)\\ \text{s.t. }\ & x_1/x_2 = \log(x_3),\\ & 3x_1 + x_2 \le 5, x_1 \ge 0, x_2 \ge 0.\end{aligned}$$

In this problem, we still force $x_1/x_2 = \log(x_3)$, but apply a least squares penalty to $\log(x_1) - 1$ and a smaller one-sided penalization to $x_2^2 - 2$.

The above formulation is nonsmooth due to the max term in the objective function; in practice we would replace this by:

$$\begin{aligned}\min_{x_1,x_2,x_3,w}\ & \exp(x_1) + 5\left(\log(x_1) - 1\right)^2 + 2w\\ \text{s.t. }\ & x_1/x_2 = \log(x_3),\\ & 3x_1 + x_2 \le 5, x_1 \ge 0, x_2 \ge 0\\ & w \ge x_2^2 - 2, w \ge 0\end{aligned}$$

and recover a standard form NLP.

The "empinfo" file:

```
modeltype NLP
adjustequ
f1 sqr 5
f2 maxz 2
```

coupled with replacing line 15 with

```
solve enlpemp using emp min obj;
```

achieves this goal. The parameter values provide the penalty coefficients above.

4.2 Reformulation as an MCP

As an alternative, we can rewrite the problem as an MCP, also dealing explicity with the nonsmoothness. The empinfo file is given by:

```
modeltype NLP
adjustequ
f1 sqr 5
f2 maxz 2
```

and this generates the following MCP:

$$\begin{array}{lcl}
0 = \log(x_1) - 1 + y_1/10 & \perp & y_1 \text{ free,} \\
0 \le x_2^2 - 2 & \perp & y_2 \ge 0 \\
0 = x_1/x_2 - \log(x_3) & \perp & y_3 \text{ free,} \\
0 \ge 3x_1 + x_2 - 5 & \perp & y_4 \le 0 \\
0 \le exp(x_1) - y_1/x_1 - y_3/x_2 - 3y_4 & \perp & x_1 \ge 0 \\
0 \le -2y_2x_2 + x_1y_3/x_2^2 - y_4 & \perp & x_2 \ge 0 \\
0 = y_3/x_3 & \perp & x_3 \text{ free,}
\end{array}$$

where y represent the multipliers.

A complete description of the process to derive this MCP will be given later in Section 10.

5 Dual Problems

MCF: Should add ability for form the dual problem here

6 Bilevel Programs

Mathematical programs with optimization problems in their constraints have a long history in operations research including [?, ?, ?]. New codes are being developed that exploit this structure, at least for simple hierarchies, and attempt to define and implement algorithms for their solution.

The simplest case is that of bilevel programming, where an upper level problem depends on the solution of a lower level optimization. For example:

$$\begin{array}{ll}
\min\limits_{x,y} & f(x,y) \\
\text{s.t.} & g(x,y) \le 0, \\
& y \text{ solves } \min\limits_{y} v(x,y) \text{ s.t.} \quad h(x,y) \ge 0.
\end{array}$$

Often, the upper level is referred to as the “leader”, while the lower level is the “follower”.

This problem can be reformulated as a Mathematical Program with Complementarity Constraints (MPCC) by replacing the lower level optimization problem by its first order optimality conditions:

$$\begin{array}{ll}
\min\limits_{x,y} & f(x,y) \\
\text{s.t.} & g(x,y) \le 0, \\
& 0 = \nabla_y v(x,y) - \lambda^T \nabla_y h(x,y) \perp x \text{ free} \\
& 0 \le h(x,y) \perp \lambda \ge 0.
\end{array}$$

We find a solution of the MPCC, not of the bilevel program. This approach allows the MPCC to be solved using the NLPEC code, for example. Note that this reformulation is potentially problematic. First order conditions require theoretical assumptions to be necessary and sufficient for *local optimality*. There may be cases where the lower level problem has multiple local solutions, but the modeler really was interested in the *global* solution. The approach here may not produce this solution, even if a global solver is used within NLPEC.

The following example is example 5.1.1, page 197 from [**?**]. Mathematically, the problem is

$$\begin{aligned}
&\min_{x,y} x - 4y \\
&\text{s.t. } y \text{ solves } \min_{y} y \\
&\qquad\qquad\quad \text{s.t. } x + y \geq 3 \\
&\qquad\qquad\qquad 2x - y \geq 0 \\
&\qquad\qquad\qquad -2x - y \geq -12 \\
&\qquad\qquad\qquad -3x + 2y \geq -4
\end{aligned}$$

and the EMPLIB model bard511 contains the following code:

```
1  positive variables x,y; variables objout,objin;
2  equations defout,defin,e1,e2,e3,e4;
3
4  defout.. objout =e= x - 4*y;
5  defin..  objin  =e= y;
6
7  e1..      x +   y =g=   3;
8  e2..   2*x -   y =g=   0;
9  e3..  -2*x -   y =g= -12;
10 e4..  -3*x + 2*y =g=  -4;
11
12 model bard / all /;
13
14 $echo bilevel x min objin y defin e1 e2 e3 e4 > "%emp.info%"
15
16 solve bard using emp minimizing objout;
```

Note that lines 1-12 define the functions that form the objectives and constraints of the model and assemble them into the model. Line 14 writes the "empinfo" file and states that the lower level problem involves the objective objin which is to be minimized by choice of variables y subject to the constraints specified in (defin), e1, e2, e3 and e4.

Note that the variables x are declared to be variables of the upper level problem and this example has no upper level constraints g. Having written the problem in this way, the MPCC is generated automatically, and passed on to a solver. In the case where that solver is NLPEC, a further reformulation of the model is carried out to convert the MPCC into an equivalent NLP or a parametric sequence of NLP's.

Further examples of bilevel models in EMPLIB are named: bard*, ccmg74, ccmg153, flds*, jointc1, jointc2, mirrlees, transbp.

The EMP model type allows multiple lower level problems to be specified within the bilevel format. An example of this is given in EMPLIB as ccmg71. The equations and objectives are specified in the normal manner; the only change is the definition of the empinfo file, shown below as lines 8-12:

```
1  ...
2
3  defh1.. h1 =e= sqr(u1-x1) + sqr(u2-x2) + sqr(u3-x3) + sqr(u4-x4);
4  e1.. 3*u1 + u2 + 2*u3 + u4 =e= 6;
5
6  ...
```

```
7
8  $onecho > "%emp.info%"
9  bilevel x1 x2 x3 x4
10 min h1 u1 u2 u3 u4 defh1 e1
11 min h2 v1 v2 v3 v4 defh2 e2
12 $offecho
```

This corresponds to a bilevel program with two followers, both solving minimization problems. The first follower minimizes the objective function h1 (defined in defh1 on line 3) over the variables u1, u2, u3 and u4 subject to the constraint given in e1. The second followers problem is defined analogously on line 11. Note that h1 involves the variables x1, x2, x3 and x4 that are optimization variables of the leader. The constraint in e1 could also include these variables, and also the variables v1, v2, v3 or v4 of the second follower, but all of these would be treated as parameters by the first follower.

The actual model (ccmg71) in EMPLIB uses a shortcut notation to replace lines 8-12 above by:

```
8  $onecho > "%emp.info%"
9  bilevel x1 x2 x3 x4
10 min h1 * defh1 e1
11 min h2 * defh2 e2
12 $offecho
```

In the followers problem defined on line 10, the '*' notation indicates that this agent will optimize over all the variables used in defh1 and e1 that are not under the control of any other follower or the leader. In this case, this means u1, u2, u3 and u4. To avoid confusion, it is recommended that the modeler explicity names all the variables in each followers problem as shown before.

7 Variational Inequalities

A variational inequality VI(F, X) is to find $x \in X$:

$$F(x)^T(z - x) \geq 0, \text{ for all } z \in X.$$

Here X is a closed (frequently assumed convex) set, defined for example as

$$X = \{x \mid x \geq 0, h(x) \geq 0\}. \tag{1.2}$$

Note that the first-order (minimum principle) conditions of a nonlinear program

$$\min_{z \in X} f(z)$$

are precisely of this form with $F(x) = \nabla f(x)$.

It is well known that such problems can be reformulated as complementarity problems when the set X has the representation (1.2) by introducing multipliers λ on the constraints h:

$$\begin{array}{lcl} 0 \leq F(x) - \lambda^T \nabla h(x) & \perp & x \geq 0 \\ 0 \leq h(x) & \perp & \lambda \geq 0. \end{array}$$

If X has a different representation, this construction would be modified appropriately.

A simple two dimensional example may be useful to improve understanding. Let

$$F(x) = \begin{bmatrix} x_1 + 2 \\ x_1 + x_2 - 3 \end{bmatrix}, \; X = \{x \geq 0 \mid x_1 + x_2 \leq 1\},$$

so that F is an affine function, but F is not the gradient of any function $f : \mathbf{R}^2 \to \mathbf{R}$. For this particular data, VI(F, X) has a unique solution $x = (0, 1)$.

```
sets J  / 1, 2 /;
positive variable x(J)  'vars, perp to f(J)';

equations F(J), h;

F(J)..  (x('1') + 2)$sameas(J,'1') + (x('1') + x('2') - 3)$sameas(J,'2') =n= 0 ;
h..     x('1') + x('2') =l= 1;

model simpleVI / F, h/;

file fx /"%emp.info%"/;
putclose fx 'vifunc F x h';

solve simpleVI using emp;
```

Note that lines 1-9 of this file define the F and h using standard GAMS syntax and include the defining equations in the model simpleVI. The extension is the annotation "empinfo" file that indicates certain equations are to be treated differently by the EMP tool. The annotation simply says that the model is a VI (vifunc) that pairs F with x and that the remaining (unpaired) equations form the constraint set X. Thus model equations F define a function F that is to be part of a variational inequality, while the equations h define constraints of X. It is also acceptable in this setting to use the empinfo file defined by:

```
putclose fx 'vifunc F x';
```

In this case, by default any equations that are given in the model statement but not included as a pair in the vifunc statement are automatically used to form X. An alternative way to write this model without using "sameas" is given in EMPLIB as affinevi.

Further example models in EMPLIB are named: simplevi, simplevi2, simplevi3, target, traffic, traffic2, transvi and zerofunc.

Note also that the lower level problems of a bilevel program could be VI's instead of optimization problems - these problems are called Mathematical Programs with Equilibrium Constraints (MPEC) in the literature. Note that since MPCC is a special case of MPEC, the GAMS model type MPEC covers both. An example demonstrating this setup is given in EMPLIB as multmpec.

8 Embedded Complementarity Systems

MCF: Need to get our names straight: embedded comp system vs equilibrium model

A different type of embedded optimization model that arises frequently in applications is:

$$\begin{array}{lll} \max\limits_x & f(x,y) & \\ \text{s.t.} & g(x,y) \leq 0 & (\perp p \geq 0) \end{array}$$

$$H(x,y,p) = 0 \qquad (\perp y \text{ free})$$

Note the difference here: the optimization problem is over the variable x, and is parameterized by the variable y. The choice of y is fixed by the (auxiliary) complementarity relationships depicted here by H. Note that the "H" equations are not part of the optimization problem, but are essentially auxiliary constraints to tie down remaining variables in the model.

MCF: Maybe use ferris43 model instead since it does everything. But simpequil2 should go into emplib.

A specific example is:

$$\begin{array}{ll} \max\limits_x & x \\ \text{s.t.} & x + y \leq 1 \end{array}$$

$$-3x + y = 0.5 \qquad (\perp y \text{ free})$$

which is found in the EMPLIB model simpequil2:

```
1  variables y; positive variables x;
2  equations optcons, vicons;
3
4  optcons.. x + y =l= 1;
5  vicons.. -3*x + y =e= 0.5;
6
7  model comp / optcons, vicons /;
8
9  file info / '%emp.info%' /;
10 put info / 'equilibrium';
11 put      / 'max x optcons';
12 putclose / 'vifunc vicons y';
13
14 solve comp using emp;
```

In order that this model can be processed correctly as an EMP, the modeler provides additional annotations to the model defining equations (lines 1-7 above) in an "empinfo" file (lines 9-12). Specifically, line 10 indicates the problem is an equilibrium problem involving one or more agent problems. Line 11 defines the first agent as an optimizer (over x), and line 12 defines the second agent as solving a VI in y. Armed with this additional information, the EMP tool automatically creates the following MCP:

$$\begin{array}{lll} 0 \le -1 + p & \perp & x \ge 0 \\ 0 \le 1 - x - y & \perp & p \ge 0 \\ 0 = -3x + y - 0.5 & \perp & y \text{ free,} \end{array}$$

(which is formed by the steps we outline below).

The above example is slightly simpler than the general form described above in which H is a function of x, y and p, the multiplier on the constraint of the optimization problem. The problem is that we do not have that variable around in the model code if we only specify the optimization problem there. This occurs for example in the clasical PIES Model due to Hogan. In this setting, the problem is described by a linear program

$$\begin{array}{ll} \min_x & c^T x \\ \text{s.t.} & Ax = q(p) \\ & Bx = b \\ & x \ge 0 \end{array}$$

in which the quantity q is a function of p, which is a multiplier on one of the LP constraints. To do this in EMP, we simply add the annotation:

```
1  model piesemp / defobj, dembal, cmbal, ombal, lmbal, hmbal, ruse /;
2
3  file myinfo /'%emp.info%'/;
4  put myinfo 'equilibrium ';
5  put 'min obj c o ct ot lt ht defobj dembal cmbal ombal lmbal hmbal
6  ruse ';
7  putclose 'dualvar p dembal';
8
9  solve piesemp using emp;
```

where dembal is the name of the constraint for which p needs to be the multiplier. The full model is found in the EMPLIB model pies. Two final points: the dualvar directive identifies the variable p with the multiplier on the dembal constraint, and all variables and constraints must be owned by a single agent. In this case, since there is only one agent (the minimizer), all constraints of the model are explicity claimed in line 5, along with all variables except p. However, line 5 identifies p with the dembal constraint which is owned by the min agent, and hence p is also owned by that agent. EMP explicitly enforces the rule that every variable and constrain

There are several shorthands possible here. The first is that line 5 can be replaced by the '*' form:

```
5 put 'min obj * defobj dembal cmbal ombal lmbal hmbal ruse ';
```

Alternatively, an even shorter version is possible since there is only one agent present in this model, namely:

```
model piesemp / defobj, dembal, cmbal, ombal, lmbal, hmbal, ruse /;

file myinfo /'%emp.info%'/;
putclose myinfo 'dualvar p dembal';

solve piesemp using emp minimizing obj;
```

Note that in this form, all the variables and constraints of the original model are included in the (single) agents problem, and the original variable p is identified in the constructed MCP with the multiplier on the dembal constraint.

In the general case where the empinfo file contains all three lines:

```
min x optcons
vifunc vicons y
dualval p optcons
```

namely that the function H that is defined in vicons is complementary to the variable y (and hence the variable y is a parameter to the optimization problem), and furthermore that the dual variable associated with the equation optcons in the optimization problem is one and the same as the variable p used to define H, the EMP tool automatically creates the following MCP:

$$\begin{array}{lcl} 0 = \nabla_x \mathcal{L}(x,y,p) & \perp & x \text{ free} \\ 0 \geq -\nabla_p \mathcal{L}(x,y,p) & \perp & p \leq 0 \\ 0 = H(x,y,p) & \perp & y \text{ free,} \end{array}$$

where the Lagrangian is defined as

$$\mathcal{L}(x,y,p) = f(x,y) - p^T g(x,y).$$

Essentially, this MCP consists of the first order optimality conditions of the optimization problem, coupled with the VI that is the second agents problem. An example that does both of these things together is provided in EMPLIB as scarfemp-primal.

Note that since the PIES model has no y variables, this is a special case of the general form in which the second agents (VI) problem is simply not present.

Example models are named: ferris43, flipper, pies, scarfemp-dual, simpequil, transecs, transeql

9 MOPECs

MCF: Use a Bimatrix game example?

Perhaps the most popular use of this formulation is where competition is allowed between agents. A standard method to deal with such cases is via the concept of Nash Games. In this setting x^* is a Nash Equilibrium if

$$x_i^* \in \arg\min_{x_i \in X_i} \ell_i(x_i, x_{-i}^*, q), \forall i \in \mathcal{I},$$

where x_{-i} are other players decisions and the quantities q are given exogenously, or via complementarity:

$$0 \leq H(x,q) \quad \perp \quad q \geq 0.$$

This mechanism is extremely popular in economics, and Nash famously won the Nobel Prize for his contributions to this literature.

This format is again an EMP, more general than the example given above in two respects. Firstly, there is more than one optimization problem specified in the embedded complementarity system. Secondly, the parameters in each optimization problem consist of two types. Firstly, there are the variables q that are tied down by the auxiliary complementarity condition and hence are treated as parameters by the ith Nash player. Also there are the variables x_{-i} that are treated as parameters by the ith Nash player, but are treated as variables by a different player j.

While we do not specify the syntax here for these issues , EMPmanual provides examples that outline how to carry out this matching within GAMS. Finally, two points of note: first it is clear that the resulting model is a complementarity problem and can be solved using PATH, for example. Secondly, performing the conversion from an embedded complementarity system or a Nash Game automatically is a critical step in making such models practically useful.

We note that there is a large literature on discrete-time finite-state stochastic games: this has become a central tool in analysis of strategic interactions among forward-looking players in dynamic environments. The model of dynamic competition in an oligopolistic industry given in [?] is exactly in the format described above, and has been used extensively in applications such as advertising, collusion, mergers, technology adoption, international trade and finance. Ongoing work aims to use the EMP format to model these problems.

10 Extended Nonlinear Programs

Optimization models have traditionally been of the form (1.1). Specialized codes have allowed certain problem structures to be exploited algorithmically, for example simple bounds on variables. However, for the most part, assumptions of smoothness of f, g and h are required for many solvers to process these problems effectively. In a series of papers, Rockafellar and colleagues [?, ?, ?] have introduced the notion of extended nonlinear programming, where the (primal) problem has the form:

$$\min_{x\in X} f(x) + \theta(-g_1(x), \dots, -g_m(x)). \tag{1.3}$$

In this setting, X is assumed to be a nonempty polyhedral set, and the functions $f, g_1, \dots, g_m$ are smooth. The function θ can be thought of as a generalized penalty function that may well be nonsmooth. However, when θ has the following form

$$\theta(u) = \sup_{y\in Y}\{y^T u - k(y)\}, \tag{1.4}$$

a computationally exploitable and theoretically powerful framework can be developed based on conjugate duality. A key point for computation and modeling is that the function θ can be fully described by defining the set Y and the function k. Furthermore, from a modeling perspective, an extended nonlinear program can be specified simply by defining the functions $f, g_1, \dots, g_m$ in the manner already provided by the modeling system, with the additional issue of simply defining Y and k. Conceptually, this is not much harder that what is carried out already, but leads to significant enhancements to the types of models that are available. Once a modeler determines which constraints are treated via which choice of k and Y, the EMP model interface automatically forms an equivalent variational inequality or complementarity problem. As we show later, there may be alternative formulations that a re computationally more appealing; such reformulations can be generated using different options to JAMS.

10.1 Forms of θ

The EMP model type makes the problem format (1.3) available to users in GAMS. As special cases, we can model piecewise linear penalties, least squares and L_1 approximation problems, as well as the notion of soft and hard constraints.

For ease of exposition, we now describe a subset of the types of functions θ that can be generated by particular choices of Y and k. In many cases, the function θ is separable, that is

$$\theta(u) = \sum_{i=1}^{m} \theta_i(u_i).$$

so we can either specify θ_i or θ itself.

Extended nonlinear programs include the classical nonlinear programming form (1.1) as a special case. This follows from the observation that if K is a closed convex cone, and we let ψ_K denote the "indicator function" of K defined by:

$$\psi_K(u) = \begin{cases} 0 & \text{if } u \in K \\ \infty & \text{else,} \end{cases}$$

then (1.1) can be rewritten as:

$$\min_x f(x) + \psi_K((-g(x), -h(x)),\ K = \mathbf{R}_+^m \times \{0\}^p,$$

where m and p are the dimensions of g and h respectively and $\mathbf{R}_+^m = \{u \in \mathbf{R}^m \mid u \geq 0\}$. An elementary calculation shows that

$$\psi_K(u) = \sup_{v \in K^\circ} u^T v,$$

where $K^\circ = \left\{u \mid u^T v \leq 0, \forall v \in K\right\}$ is the polar cone of the given cone K. Thus, when $\theta(u) = \psi_K(u)$ we simply take

$$k \equiv 0 \text{ and } Y = K^\circ. \tag{1.5}$$

In our example, $K^\circ = \mathbf{R}_-^m \times \mathbf{R}^p$. To some extent, this is just a formalism that allows us to claim the classical case as a specialization; however when we take the cone K to be more general than the polyhedral cone used above, we can generate conic programs for example.

The second example involves a piecewise linear function θ: Formally, for $u \in \mathbf{R}$,

$$\theta(u) = \begin{cases} \rho u & \text{if } u \geq 0 \\ \sigma u & \text{else.} \end{cases}$$

In this case, simple calculations prove that θ has the form (1.4) for the choices:

$$k \equiv 0 \text{ and } Y = [\sigma, \rho].$$

The special case where $\sigma = -\rho$ results in

$$\theta(u) = \rho\, |u|\,. \tag{1.6}$$

This allows us to model nonsmooth L_1 approximation problems. Another special case results from the choice of $\sigma = -\gamma$, $\rho = 0$, whereby

$$\theta(u) = \gamma \max\{-u, 0\}.$$

This formulation corresponds to a soft penalization on an inequality constraint, namely if $\theta(-g_1(x))$ is used then nothing is added to the objective function if $g_1(x) \leq 0$, but $\gamma g_1(x)$ is added if the constraint $g_1(x) \leq 0$ is violated. Contrast this to the classical setting above, where ∞ is added to the objective if the inequality constraint is violated. It is interesting to see that truncating the set Y, which amounts to bounding the multipliers, results in replacing the classical constraint by a linearized penalty.

The third example involves a more interesting choice of k. If we wish to replace the "absolute value" penalization given above by a quadratic penalization (as in classical least squares analysis), that is

$$\theta(u) = \gamma u^2 \tag{1.7}$$

then a simple calculation shows that we should take

$$k(y) = \frac{1}{4\gamma} y^2 \text{ and } Y = \mathbf{R}.$$

By simply specifying this different choice of k and Y we can generate such models easily and quickly within the modeling system: note however that the reformulation we would use in (1.6) and (1.7) are very different as we shall explain in the simple example below. Furthermore, in many applications it has become popular to penalize violations using a quadratic penalty only within a certain interval, afterwards switching to a linear penalty (chosen to make the penalty function θ continuously differentiable - see [**?**]. That is:

$$\text{i.e. } \theta(u) = \begin{cases} \gamma u - \frac{1}{2}\gamma^2 & \text{if } u \geq \gamma \\ \frac{1}{2}u^2 & \text{if } u \in [-\gamma, \gamma] \\ -\gamma u - \frac{1}{2}\gamma^2 & \text{else.} \end{cases}$$

Such functions arise from quadratic k and simple bound sets Y. In particular, the somewhat more general function

$$\theta(u) = \begin{cases} \gamma\beta^2 + \rho(u-\beta) & \text{if } u \geq \beta \\ \gamma u^2 & \text{if } u \in [\alpha, \beta] \\ \gamma\alpha^2 + \sigma(u-\alpha) & \text{else} \end{cases}$$

arises from the choice of

$$k(y) = \frac{1}{4\gamma} y^2 \text{ and } Y = [\sigma, \rho],$$

with $\alpha = \frac{\sigma}{2\gamma}$ and $\beta = \frac{\rho}{2\gamma}$.

The final example that we give is that of L_∞ penalization. This example is different to the examples given above in that θ is not separable. However, straightforward calculation can be used to show

$$\theta(u) = \max_{i=1,\ldots,m} u_i$$

results from the choice of

$$k \equiv 0 \text{ and } Y = \left\{ y \in \mathbf{R}^m \mid y \geq 0, \sum_{i=1}^m y_i = 1 \right\},$$

that is, Y is the unit simplex.

10.2 Underlying theory

The underlying structure of θ leads to a set of extended optimality conditions and an elegant duality theory. This is based on an extended form of the Lagrangian:

$$\mathcal{L}(x,y) = f(x) - \sum_{i=1}^m y_i g_i(x) - k(y)$$
$$x \in X, y \in Y$$

Note that the Lagrangian $\mathcal{L}$ is smooth - all the nonsmoothness is captured in the θ function. The theory is an elegant combination of calculus arguments related to g_i and its derivatives, and variational analysis for features related to θ.

It is shown in [**?**] that under a standard constraint qualification, the first-order conditions of (1.3) are precisely in the form of the following variational inequality:

$$\text{VI}\left(\begin{bmatrix} \nabla_x \mathcal{L}(x,y) \\ -\nabla_y \mathcal{L}(x,y) \end{bmatrix}, X \times Y \right). \tag{1.8}$$

When X and Y are simple bound sets, this is simply a complementarity problem.

Note that EMP exploits this result. In particular, if an extended nonlinear program of the form (1.3) is given to EMP, then the optimality conditions (1.8) are formed as a variational inequality problem and can be processed as outlined above. For a specific example, we cite the fact that if we use the (classical) choice of k and Y given in (1.5), then the optimality conditions of (1.3) are precisely the standard complementarity problem given as (**??**). While this is of interest, we believe that other choices of k and Y may be more useful and lead to models that have more practical significance.

Under appropriate convexity assumptions on this Lagrangian, it can be shown that a solution of the VI (1.8) is a saddle point for the Lagrangian on $X \times Y$. Furthermore, in this setting, the saddle point generates solution s to the primal problem (1.3) and its dual problem:

$$\max_{y \in Y} d(y), \quad \text{where } d(y) = \inf_{x \in X} \mathcal{L}(x,y),$$

with no duality gap.

10.3 A simple example

MCF: This repeats stuff from earlier section, needs work. EMPLIB model is simpenlp

As an example, consider the problem

$$\begin{aligned} \min_{x_1,x_2,x_3} \quad & \exp(x_1) + 5\left\|\log(x_1) - 1\right\|^2 + 2\max(x_2^2 - 2, 0) \\ \text{s.t.} \quad & x_1/x_2 = \log(x_3), \\ & 3x_1 + x_2 \le 5, x_1 \ge 0, x_2 \ge 0. \end{aligned}$$

In this problem, we would take

$$X = \left\{x \in \mathbf{R}^3 \mid 3x_1 + x_2 \le 5, x_1 \ge 0, x_2 \ge 0\right\}.$$

The function θ essentially treats 3 separable pieces:

$$\begin{aligned} g_1(x) &= \log(x_1) - 1, \\ g_2(x) &= x_2^2 - 2, \\ g_3(x) &= x_1/x_2 - \log(x_3). \end{aligned}$$

A classical problem would force $g_1(x) = 0$, $g_2(x) \le 0$ and $g_3(x) = 0$, while minimizing $f(x) = \exp(x_1)$. In our problem, we still force $g_3(x) = 0$, but apply a (soft) least squares penalty on $g_1(x)$ and a smaller one-sided penalization on $g_2(x)$. The above formulation is nonsmooth due to the max term in the objective function; in practice we could replace this by:

$$\begin{aligned} \min_{x_1,x_2,x_3,w} \quad & \exp(x_1) + 5\left\|\log(x_1) - 1\right\|^2 + 2w \\ \text{s.t.} \quad & x_1/x_2 = \log(x_3), \\ & 3x_1 + x_2 \le 5, x_1 \ge 0, x_2 \ge 0 \\ & w \ge x_2^2 - 2, w \ge 0 \end{aligned}$$

and recover a standard form NLP. If the penalty on $g_1(x)$ would be replaced by a one-norm penalization (instead of least squares), we would have to play a similar game, moving the function $g_1(x)$ into the constraints and adding additional variable(s). To some extent, this seems unnatural - a modeler should be able to interchange the penalization without having to reformulate the problem from scratch. The proposed extended NLP would not be reformulated at all by the modeler, but allows all these "generalized constraints" to be treated in a similar manner within the modeling system. The actual formulation would take:

$$\theta(u) = \theta_1(u_1) + \theta_2(u_2) + \theta_3(u_3)$$

where

$$\begin{aligned} \theta_1(u_1) &= 5u_1^2, \\ \theta_2(u_2) &= 2\max(u_2, 0), \\ \theta_3(u_3) &= \psi_{\{0\}}(u_3). \end{aligned}$$

The discussion above allows us to see that

$$\begin{aligned} Y &= \mathbf{R} \times [0, 2] \times \mathbf{R}, \\ k(y) &= \frac{1}{20} y_1^2 + 0 + 0. \end{aligned}$$

The corresponding Lagrangian is the smooth function:

$$\mathcal{L}(x, y) = f(x) - \sum_{i=1}^{3} y_i g_i(x) - k(y).$$

The corresponding VI (1.8) can almost be formulated in GAMS (except that the linear constraint in X cannot be handled currently except by introducing a $\theta_4(x)$). Thus

$$g_4(x) = 3x_1 + x_2 - 5,\ \theta_4(u) = \psi_{\mathbf{R}_+}$$

resulting in the following choices for Y and k:

$$\begin{aligned} Y &= \mathbf{R} \times [0,2] \times \mathbf{R} \times \mathbf{R}_-, \\ k(y) &= \frac{1}{20}y_1^2 + 0 + 0 + 0. \end{aligned}$$

Since X and Y are now simple bound sets, (1.8) is now a complementarity problem and can be solved for example using PATH. A simple "empinfo" file details the choices of Y and k from the implemented library:

```
Adjustequ
e1 sqr 5
e2 MaxZ 2
```

The full model and option files are available in [?].

10.4 Reformulation as a classical NLP

Suppose

$$\theta(u) = \sup_{y\in Y}\{u^T y - \frac{1}{2}y^T Q y,\}$$

for a polyhedral set $Y \in \mathbf{R}^m$ and a symmetric positive semidefinite $Q \in \mathbf{R}^{m\times m}$ (possibly $Q = 0$). Suppose further that

$$\begin{aligned} &X = \{x \mid Rx \le r\},\ \ Y = \left\{y \mid S^T y \le s\right\}, \\ &Q = DJ^{-1}D^T,\ \ F(x) = (g_1(x), \ldots, g_m(x)), \end{aligned}$$

where J is symmetric and positive definite (for instance $J = I$). Then, as outlined by [?], the optimal solutions $\bar{x}$ of (1.3) are the $\bar{x}$ components of the optimal solutions $(\bar{x}, \bar{z}, \bar{w})$ to

$$\begin{array}{ll} \min & f(x) + s^T z + \frac{1}{2}w^T J w \\ \text{s.t.} & Rx \le r, z \ge 0, F(x) - Sz - Dw = 0. \end{array}$$

The multiplier on the equality constraint in the usual sense is the multiplier associated with $\bar{x}$ in the extended Lagrangian for (1.3). (Note that a Cholesky factorization may be needed to determine D.)

It may be better to solve this reformulated NLP than to solve (1.8). However, it is important that we can convey all types of nonsmooth optimization problems to a solver as smooth optimization problems, and hence it is important to communicate the appropriate structure to the solver interface. We believe that specifying Y and k is a theoretically sound way to do this.

Example models are named:

11 Disjunctive Programs

There are many ways that the EMP model type can be used for further extensions to the modeling capabilities of a given system. In particular, the procedures outlined in [?] for disjunctive programming extensions are also implemented within the EMP model type.

One simple example to highlight this feature is the notion of an ordering of tasks, namely that either job i comes before job j or the converse. Such a disjunction can be specified using an empinfo file containing lines:

```
  disjuncton * seq(i,j) else seq(j,i)
```

In such an example, one can implement a Big-M method, employ indicator constraints, or utilize a convex hull reformulation. The convex hull reformulation is the default strategy; to utilize the Big-M formulation, the additional option

```
  default bigm 1000
```

would add binary variables and constraints to impose the disjunction using a Big-M value of 1000. Alternatively, for the CPLEX solver, the option setting (for EMP):

```
  default indic
```

writes out a model and a CPLEX option file that implements a reformulation using indicator constraints. The EMP model library that is part of the standard GAMS distribution contains a sequencing model that implements all of these options.

More complicated (nonlinear) examples make the utility of this approach clearer. The design of a multiproduct batch plan with intermediate storage described in [?] and a synthesis problem involving 8 processes from [?] are also included in the EMP model library. As a final example, the gasoline emission model outlined in [?] is precisely in the form that could exploit the features of EMP related to (nonlinear) disjunctive programming.

Example models are named: makespan, sequence.

12 Options available

The empinfo file has a vectorized format and a more powerful (but more complex) scalar version.

The JAMS solver has the following options:

Can pass user defined parameters to the subsolver ("subsolverpar").

The format of the empinfo file is given below:

```
Disjunction [chull [big eps] | bigM [big eps threshold] | indic]
            [NOT] var|* [NOT] {equ} {ELSEIF [NOT] var|* [NOT] {equ}} [ELSE [NOT] {equ}]

Default [chull [big eps] | bigM [big eps threshold] | indic]

ParallelStep1 {equ|*}

AdjustEqu equ abs|sqr|maxz|huber|... {weight {param}}

ModelType MCP|NLP|MIP|...

BiLevel {var} {MAX|MIN obj {var|*} {[-] equ}} {VI {var|*} {[-] equ var} {[-] equ}} {DualVar {var [-] eq

Equilibrium {MAX|MIN obj {var|*} {[-] equ}} {VI {var|*} {[-] equ var} {[-] equ}} {DualVar {var [-] equ}

VI {var|*} {[-] equ var} {[-] equ}

DualEqu {[-] equ var}

DualVar {var [-] equ}

-------
[ ] optional      | exclusive      { } can be repeated
```

Last modified February 12, 2011 by MCF