Paper 1820-2016

Portfolio Optimization with Discontinuous Constraint

Robert Spatz, University of Chicago; Taras Zlupko, University of Chicago

ABSTRACT

Optimization models require continuous constraints to converge. However, some real life problems are better described by models which incorporate discontinuous constraints. A common type of such discontinuous constraints becomes apparent when a regulation-mandated diversification requirement is implemented in investment portfolio model. Generally stated the requirement postulates that the aggregate of investments with individual weights exceeding certain threshold in the portfolio should not exceed some predefined total within the portfolio. This format of the diversification requirement may be defined by the rules of any specific portfolio construction methodology and is commonly imposed by the regulators. The paper discusses the impact of this type of discontinuous portfolio diversification constraint on the portfolio optimization model solution process and develops a convergent approach. The latter includes a sequence of definite series of convergent non-linear optimization problems and is presented in the framework of the PROC OPTMODEL modeling environment. The approach discussed has been used in constructing investable equity indexes.

INTRODUCTION

Investment portfolio methodology rules commonly include some diversification requirements. These requirements translate into model constraints and usually are concerned with defining some weight limits of individual securities in the portfolio. Therefore, the constraints are expressed as linear inequalities in the optimization model and are processed in a regular manner by the solvers. There are also constraints that relate to groups of securities. Mutual funds, exchange-traded funds and other investment vehicles usually communicate their concentration by disclosing ten or so largest holdings and their aggregate percentage weight in the portfolio. This approach is based purely on securities count and can be formalized in portfolio model.

Regulation-mandated approach to diversification takes the approach where aggregate weight of a group of securities identified on the basis of their individual weights rather than count should not exceed certain threshold. Securities that qualify for the group are those where weights exceed some predefined limit. This type of diversification requirement translates into a discontinuous model constraint and may prevent optimization solver to converge. Although this type of diversification constraint may be imposed by particulars of any given portfolio construction methodology, the most common versions of this constraint comes from the regulation. Being mandatory across regulated investment companies this type of constraint is discussed in the paper.

The US Internal Revenue Code requires the following two conditions to be met at the end of each quarter of a regulated investment company's tax year: 1) No more than twenty five percent of the value of the regulated investment company's assets might be invested in a single issuer. 2) The sum of the weights of all issuers representing more than five percent of the total assets should not exceed fifty percent of the fund's total assets.

Similar requirements in Europe stipulate that no more than ten per cent of assets are invested in a single issuer. The sum of the weights of all issuers representing more than five percent of the total assets should not exceed forty percent of the collective investment's total assets.

While the first requirement defined by the regulation produces a set of linear inequality constraints usual for convex optimization, the second requirement yields a discontinuous constraint leading to model non-convergence.

A mathematical programming model that includes the regulation-mandated diversification requirements was presented and solved in the PROC OPTMODEL modeling environment by the authors of this paper at the SAS Global Forum 2015 conference. This paper further exposes the issue of the discontinuous constraint in nonlinear optimization and provides discussion of the proposed solution.

THE DISCONTINUOUS CONSTRAINT

In order to illustrate the solution process of a nonlinear model with discontinuous constraint we will use the case of the US regulation mentioned above and briefly reintroduce the model presented in the previous paper. The goal of the model is to assure that the resultant portfolio is regulation-compliant while minimizing the changes to the original portfolio. The latter is expressed in the objective function as the sum of squared differences between security weights in initial and optimized portfolios. Overall, the mathematical model including constraints is as follows:

$$min \sum_{i=1}^{N} (w_i - x_i)^2 \quad (1)$$

subject to

$$\sum_{i=1}^{N} x_i = 1 \quad (2)$$

$$0 \leq x_i \leq 0.25 \quad (3)$$

$$\sum_{i:\, x_i > 0.05} x_i \leq 0.5 \quad (4)$$

where

w_i – weight of i-th security in original portfolio

x_i – weight of i-th security in regulation-compliant portfolio

N – number of securities in portfolio

Please note that since w_i is security weight in initial portfolio the sum of w_i is always unity.

The mathematical model above is formulated and solved by using the OPTMODEL procedure. In our illustrative example we use the initial portfolio which includes twenty eight securities where two of the securities exceed the 0.25 weight limit.

The following OPTMODEL procedure statements concisely define and solve model (1)-(4) above:

```
proc optmodel;
   number iw{1..28};
   read data input into [_n_] iw;
   /*model description*/
   var x{i in 1..28} >=0.0 <=0.25;
   min z = sum{i in 1..28}((iw[i]-x[i])**2);
   con c1:sum{i in 1..28} (if x[i]>0.05 then x[i] else 0) <=0.5;
   con c2:sum{i in 1..28} x[i]=1.0;
   solve;
   quit;
```

The OPTMODEL procedure is suited for solving different types of mathematical optimization problems. In the code above when no solver option is specifically defined, starting with SAS 9.3 the OPTMODEL

procedure will identify the Nonlinear Programming Solver (NLP) as appropriate for the type of model at hand. By default the NLP solver applies the interior point algorithm, sets the maximum number of iterations at 5000 and the feasibility tolerance at 1E-6.

Figure 1 and Figure 2 below show problem summary and solution summary of model (1)-(4) produced by the OPTMODEL procedure.

	Label1	cValue1	nValue1
1	Objective Sense	Minimization	.
2	Objective Function	z	.
3	Objective Type	Quadratic	.
4			.
5	Number of Variables	28	28.000000
6	Bounded Above	0	0
7	Bounded Below	0	0
8	Bounded Below and Above	28	28.000000
9	Free	0	0
10	Fixed	0	0
11			.
12	Number of Constraints	2	2.000000
13	Linear LE (<=)	0	0
14	Linear EQ (=)	1	1.000000
15	Linear GE (>=)	0	0
16	Linear Range	0	0
17	Nonlinear LE (<=)	1	1.000000
18	Nonlinear EQ (=)	0	0
19	Nonlinear GE (>=)	0	0
20	Nonlinear Range	0	0

Figure 1. Problem Summary. Model (1)-(4).

	Label1	cValue1	nValue1
1	Solver	NLP/INTERIORPOINT	.
2	Objective Function	z	.
3	Solution Status	Iteration Limit Reached	.
4	Objective Value	0.0571642358	0.057164
5	Iterations	5000	5000.000000
6			.
7	Optimality Error	0.004459369	0.004459
8	Infeasibility	9.8461297E-7	0.000000985

Figure 2. Solution Summary. Model (1)-(4).

Solution status in Figure 2 indicates that the solver exits on maximum number of iterations allowed by NLP default settings and not because of meeting the interior point algorithm optimality criteria. This can further be seen in Figure 3 below which includes the excerpt of the solver log within a few iterations before exit. Columns from left to right are iteration number, objective value, infeasibility and optimality error.

```
                ...
                4997        0.05716516    0.0000000006087        4.95404164
                4998        0.05716516    0.0000000006087        4.95404164
                4999        0.05716516    0.0000000006087        4.95404163
                5000        0.05701099         0.05815258        0.05815258
NOTE: Maximum number of iterations reached.
NOTE: Objective = 0.057010991.
NOTE: Objective of the best feasible solution found = 0.0571642358.
NOTE: The best feasible solution found is returned.
```

Figure 3. Solver Log at Exit. Model (1)-(4).

As evident in Figure 3, the last iteration of the solver is not the optimal one because of infeasibility. According to SAS/OR documentation exiting on maximum number of iterations means that the solver could not find a local optimum in the pre-specified number of iterations. In this case best feasible solution produced in the course of model processing is retained at solver exit. Related objective value is then presented in the solver log and in the solution summary.

To illustrate the process of model solution Figure 4 and Figure 5 show changes in value of objective function and infeasibility produced by the OPTMODEL procedure at each iteration of the NLP solver progression available in program log. Figure 4 includes all iterations. Figure 5 shows values of objective function and infeasibility for feasible iterations only.

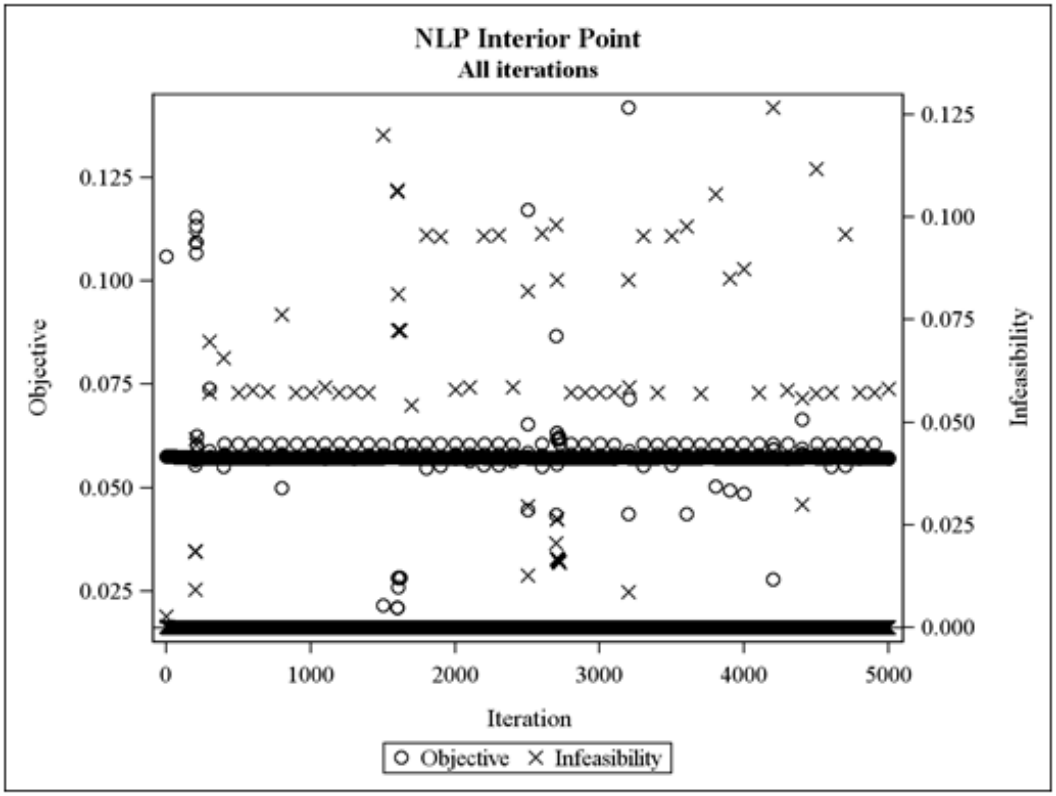

Figure 4. Objective and Infeasibility: All Iterations. Model (1)-(4).

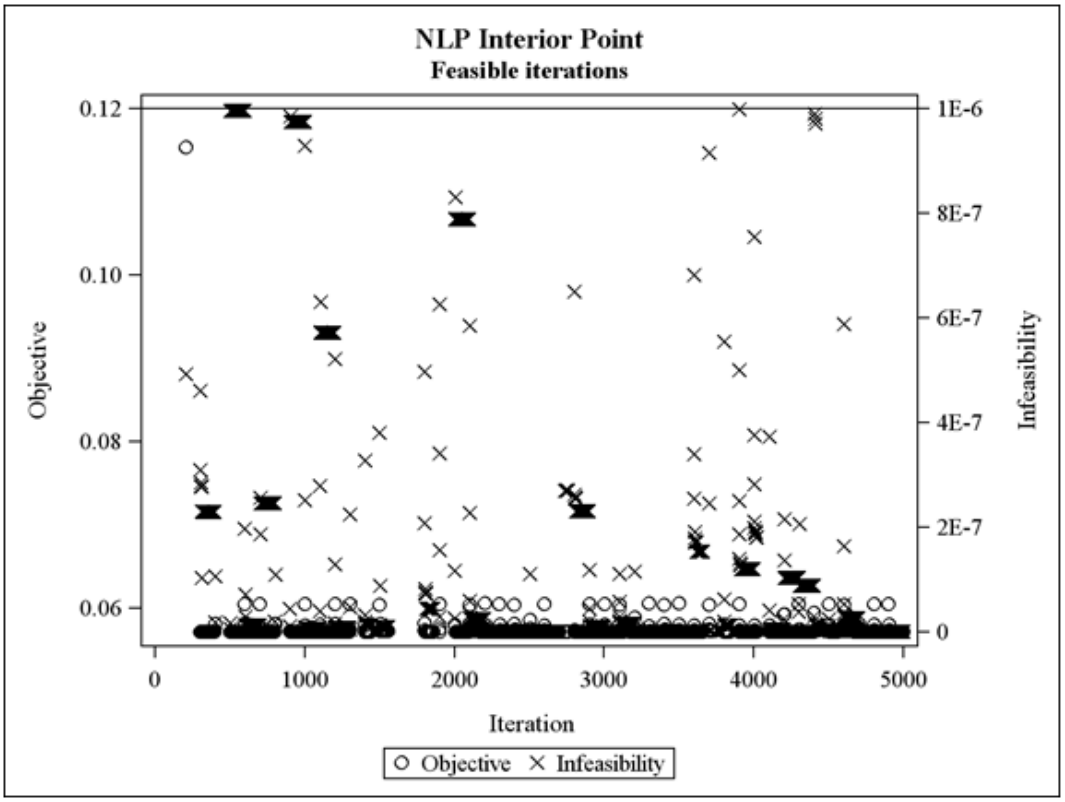

Figure 5. Objective and Infeasibility: Feasible Iterations. Model (1)-(4).

By including feasible iterations only, Figure 5 makes it evident that the objective function minimum is achieved at multiple iterations in the course of the problem solving process while at the same time satisfying feasibility. However, the solver continues to go on to next iteration and exits only when maximum number of allowed iterations is reached.

As mentioned earlier, the NLP solver in the example above used the default interior point algorithm. Another algorithm available with the NLP solver is active set. Similarly to the interior point algorithm the active set algorithm exits at the maximum number of iterations reached and not on optimal solution status.

SOLUTION TO DISCONTINUITY

Upon examination, the discontinuity of the model (1)-(4) is caused by the fourth constraint that is conditional:

$$\sum_{i:\, x_i > 0.05} x_i \;\leq 0.5 \qquad (4)$$

Examining the conditional clause in the constraint (4) above it becomes clear that while i can include different securities in a portfolio, no more than nine securities in any given portfolio can have weights greater than 0.05 (“large” securities) and have the sum of their weights be less than or equal to 0.5 for the constraint (4) to be true.

Further it can also be shown that the rank order of weights does not change in a least squares minimization (i.e. if $w_i < w_j$, then $x_i < x_j$). Therefore the candidates for the largest securities in the

regulation-compliant portfolio can be selected from a sequence of initial portfolio weights sorted in descending order.

This allows to re-state the model (1)-(4) as follows:

$$min \left\lfloor \sum_{l=1}^{L}(w_l - x_l)^2 + \sum_{s=1}^{N-L}(w_s - x_s)^2 \right\rfloor \quad (1a)$$

subject to

$$\sum_{l=1}^{L} x_l + \sum_{s=1}^{N-L} x_s = 1 \quad (2a)$$

$$0 \leq x_s \leq 0.05 \quad (3a)$$

$$0 \leq x_l \leq 0.25 \quad (3b)$$

$$\sum_{l=1}^{L} x_l \leq 0.5 \quad (4a)$$

where:

x_l – weight of l-th security that may be "large" in regulation-compliant portfolio

w_l – weight of l-th security in the original portfolio

x_s – weight of s-th security that must be "small" in regulation-compliant portfolio

w_s – weight of s-th security in the original portfolio

w_i – weights of securities in the original portfolio are sorted in descending order

N – number of securities in portfolio

L – number of securities that may be "large" in regulation-compliant portfolio. L can range from zero to nine. When L=0, the terms based on L treated as zero.

Model (1a)-(4a) above is reflective of the state when both initial and optimized portfolios are split into "large" and "small". The model doesn't make any assumptions with regard to the weight composition of original portfolio. It utilizes the knowledge of the following which is true for any portfolio given the task at hand and initial model set-up:

- weights of original portfolio may need to be redistributed in such a way that up to nine securities may be "large" in the optimized portfolio
- any "large" security in the optimized portfolio will come from the first nine largest securities in the original portfolio

Although the maximum number of "large" securities in the regulation-compliant portfolio is known and equals nine, the unknown is the actual number of securities that will need to be "large" in the optimized portfolio in order to achieve the minimum of the objective function. Therefore, model (1a)-(4a) is solved consecutively for each number of potential "large" securities in optimal portfolio which can be none to nine.

Optimal solution is the one that yields minimal objective function among the solutions of consecutive runs of the model (1a)-(4a) as described above.

IMPLEMENTATION NOTES AND EXAMPLE

Example below shows the SAS code solving the model (1a)-(4a). For clarity of illustration this is done for only one L which in this case is assumed to equal two.

The steps are as follows:

1. Sort original portfolio weights in descending order.
2. Designate “large” and “small” securities.
3. Define and solve the model using PROC OPTMODEL.
4. Combine optimized portfolio weights into single dataset.

The initial portfolio used here is the same as in the example illustrating model (1)-(4). It is the dataset “input” which includes two columns: Security identifier - “idx”, and security weight - “iw”. There are twenty eight securities in the portfolio.

```
      /*sort original portfolio weights in descending order*/
      proc sort data=d.input; by descending iw;
      run;

      /*designate “large” and “small” securities when L=2*/
      data d.input_l (keep=idxl iwl) d.input_s (keep=idxs iws);
         set d.input;
            length idxl iwl idxs iws 8;
               if (_N_ <= 2) then do;
                  idxl  = idx;
                  iwl   = iw;
                  output d.input_l;
                                 end;
               else do;
                  idxs  = idx;
                  iws   = iw;
                  output d.input_s;
                    end;
      run;

      /*define and solve the model*/
      proc optmodel;
            ods output ProblemSummary=exps SolutionSummary=exss;

      /*define initial values*/
      number iws{1..26};
      number iwl{1..2};
      read data d.input_s into [_n_] iws;
      read data d.input_l into [_n_] iwl;

/*declare variables, objective and constraints and solve*/
var xs{i in 1..26} >=0.0 <=0.05;
var xl{i in 1..2}  >=0.0 <=0.25;
min z=sum{i in 1..26}((iws[i]-xs[i])**2)+sum{i in 1..2}((iwl[i]-xl[i])**2);
con c1:sum{i in 1..2}  xl[i] <=0.5;
con c2:sum{i in 1..26} xs[i] + sum{i in 1..2} xl[i] = 1.0;
solve;

/*combine regulation-compliant portfolio weights into single dataset*/
create data d.output_large from [_n_] iwl xl;
```

```
create data d.output_small from [_n_] iws xs;
quit;
```

Problem summary and solution summary of model (1a)-(4a) produced by the OPTMODEL procedure when L=2 are shown in Figure 6 and Figure 7 respectively.

	Label1	cValue1	nValue1
1	Objective Sense	Minimization	.
2	Objective Function	z	.
3	Objective Type	Quadratic	.
4			.
5	Number of Variables	28	28.000000
6	Bounded Above	0	0
7	Bounded Below	0	0
8	Bounded Below and Above	28	28.000000
9	Free	0	0
10	Fixed	0	0
11			.
12	Number of Constraints	2	2.000000
13	Linear LE (<=)	1	1.000000
14	Linear EQ (=)	1	1.000000
15	Linear GE (>=)	0	0
16	Linear Range	0	0

Figure 6. Problem Summary. Model (1a)-(4a), L=2.

	Label1	cValue1	nValue1
1	Solver	NLP/INTERIORPOINT	.
2	Objective Function	z	.
3	Solution Status	Optimal	.
4	Objective Value	0.0571645707	0.057165
5	Iterations	6	6.000000
6			.
7	Optimality Error	5E-9	5E-9
8	Infeasibility	1.5079629E-9	1.5079629E-9

Figure 7. Solution Summary. Model (1a)-(4a), L=2.

Comparing to the original model (1)-(4), the restated model (1a)-(4a) is a quadratic linear model and solves in six iterations for L=2 reaching optimal solution.

A macro can be written to efficiently find the solution of model (1a)-(4a) where a number of “large” securities is consecutively increased to nine. An idea of implementation of this macro can be gleaned from the example given in the authors’ earlier paper. Sorting original portfolio weights needs only be done once before “large” and “small” securities are designated and macro executed. The optimal weights are retained for each L until objective functions from all runs are obtained and compared, and best solution identified.

The summary of the statistics created by all ten runs of model (1a)-(4a) is presented in Table 1. The

minimum of the objective function is achieved when the number of “large” securities in the regulation-compliant portfolio is two. Note that each of the runs reached optimal solution. This was achieved in six or less iterations and combined took only 53 iterations and 0.125 seconds of solver time. This is in contrast to not concluding with optimality at 5,000 iterations when solving the initial model (1)-(4).

L	Objective Value	Iteration	Solver Time	Solution Status
0	0.2568828	6	0.016	Optimal
1	0.1154255	4	0.000	Optimal
2	0.0571646	6	0.031	Optimal
3	0.0603897	4	0.000	Optimal
4	0.0617770	6	0.000	Optimal
5	0.0630380	5	0.016	Optimal
6	0.0641847	5	0.015	Optimal
7	0.0654369	6	0.016	Optimal
8	0.0664391	6	0.015	Optimal
9	0.0674577	5	0.016	Optimal

Table 1. Solution Results and Summary Statistics. Model (1a)-(4a).

Figure 8 and Figure 9 show objective function and infeasibility at each iteration of the solver.

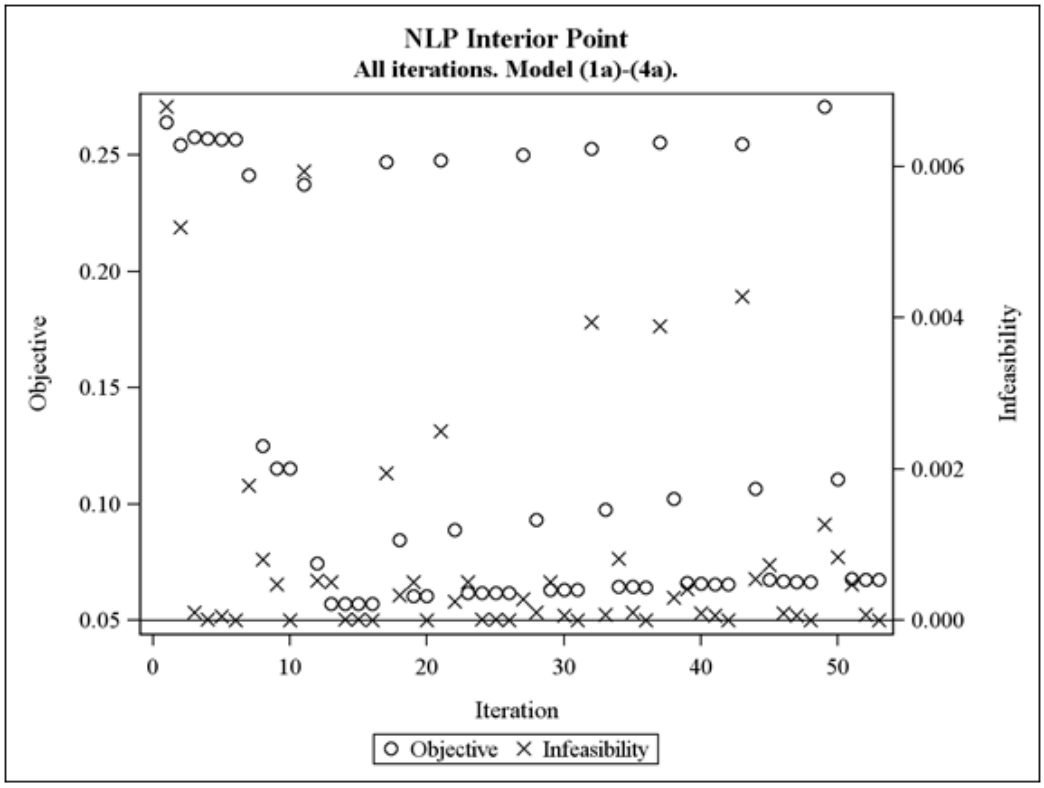

Figure 8. Objective and Infeasibility: All Iterations. Model (1a)-(4a).

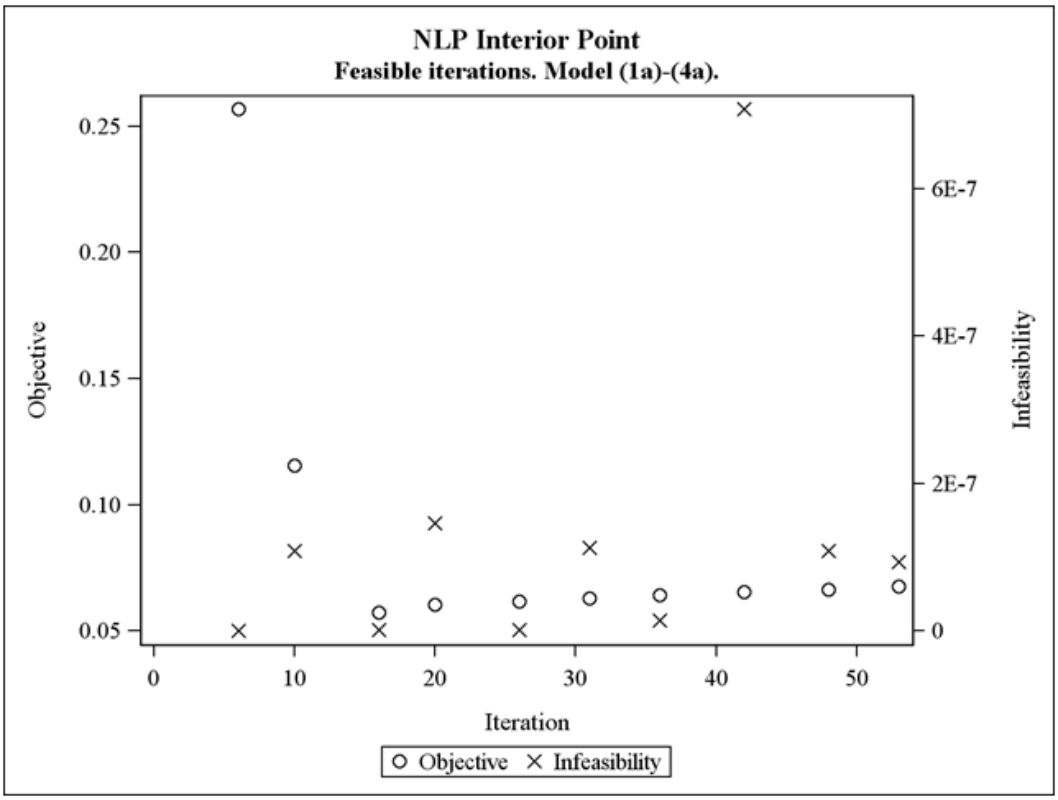

Figure 9. Objective and Infeasibility: Feasible Iterations. Model (1a)-(4a).

Solver iterations for all L runs in Figure 8 and Figure 9 are combined into one series. Figure 9 shows ten observations which are within the default OPTMODEL solver feasibility tolerance of 1E-6. These observations correspond to solutions of each of the L runs of model (1a)-(4a) when the solver exits with the optimal solution status.

In addition to SAS macro which sets-up and solves model (1a)-(4a) for different values of L, the code can be written to automate the post processing, including selecting the data associated with the lowest objective value of different L runs, checking for any error conditions or non-optimal solution status, cleaning up the temporary files, and other maintenance tasks.

In a general case, it is possible that the original portfolio includes some securities with exactly the same weights and these securities are ranked over the nine-to-ten count span when sorted descending by weight as required by model (1a)-(4a). A way to make the algorithm deterministic is to use the primary key as a secondary sort column. For example, a security with the same weight and smaller key would arbitrarily, but consistently, be placed in the same order with respect to other securities with the same weight.

It is also possible that different runs of L will solve with the same objective value. In this case, in order to assure that the algorithm is deterministic, a selection criterion is established which can be based, for example, on preferences with regard to the overall concentration of the optimized portfolio.

CONCLUSION

This paper discusses a solution of the optimization model that includes discontinuous constraint. This type of model arises when constructing investment portfolios that comply with certain regulatory requirements. The proposed solution is based on the insight about the weight composition of the regulation-compliant optimized portfolio and properties of objective function employed by the model. This allows to re-state the initial model and replace discontinuous constraint with a continuous linear inequality constraint. The solution is found by solving the re-stated model while varying number of securities that may exceed weight threshold as defined by the regulation.

The example given in the paper shows that the proposed solution efficiently converges to optimality and significantly improves solver performance. While solving the initial model, the solver exits on reaching maximum number of iterations. The re-stated model is solved by reaching optimal solution with much fewer iterations of the solver.

Discussion of the model, discontinuous constraint, solution and examples in the paper is conducted in the context of specific regulatory requirements. The solution approach presented in the paper is applicable in other contexts and can be generalized to be useful in other cases which are described by relevant type of model and constraints.

REFERENCES

Directive 2009/65/EC of the European Parliament and of the Council of 13 July 2009 on the coordination of laws, regulations and administrative provisions relating to undertakings for collective investment in transferable securities (UCITS), Chapter VII, Obligations concerning the investment policies of UCITS, Article 52. Accessed March 10, 2016. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:302:0032:0096:en:PDF

IRS Code Title 26 section 851(b)(3). Accessed March 10, 2016. http://www.gpo.gov/fdsys/pkg/USCODE-2010-title26/pdf/USCODE-2010-title26-subtitleA-chap1-subchapM.pdf

Lobo, M.G., Fazel, M. and S. Boyd. 2007. “Portfolio optimization with linear and fixed transaction costs.” *Annals of Operations Research. Financial Optimization,* 152,1:341–365.

SAS Institute Inc. 2014. SAS/OR® 13.2 User’s Guide: Mathematical Programming. Cary, NC: SAS Institute Inc.

Spatz, R. and T. Zlupko. 2015. “Portfolio Construction with OPTMODEL.” *Proceedings of the SAS Global Forum 2015 Conference*, Dallas, TX. Available at http://support.sas.com/resources/papers/proceedings15/3225-2015.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Taras Zlupko
University of Chicago
taras.zlupko@crsp.ChicagoBooth.edu
http://crsp.chicagobooth.edu/