Pseudo-Cut Strategies for Global Optimization

Fred Glover[a]
Leon Lasdon[b]
John Plummer[c]
Abraham Duarte[d]
Rafael Marti[e]
Manuel Laguna[f]
Cesar Rego[g]

Abstract

Motivated by the successful use of a pseudo-cut strategy within the setting of constrained nonlinear and nonconvex optimization in Lasdon, et al. (2010), we propose a framework for general pseudo-cut strategies in global optimization that provides a broader and more comprehensive range of methods. The fundamental idea is to introduce linear cutting planes that provide temporary, possibly invalid, restrictions on the space of feasible solutions, as proposed in the setting of the tabu search metaheuristic in Glover (1989), in order to guide a solution process toward a global optimum, where the cutting planes can be discarded and replaced by others as the process continues. These strategies can be used separately or in combination, and can also be used to supplement other approaches to nonlinear global optimization. Our strategies also provide mechanisms for generating trial solutions that can be used with or without the temporary enforcement of the pseudo-cuts.

keywords: global optimization, metaheuristics, pseudo-cuts, tabu search, adaptive memory programming

---

[a] OptTek Systems, Inc., Boulder, CO 80302, USA (glover@opttek.com)
[b] Information, Risk, and Operations Management Department, The University of Texas at Austin, USA (Leon.Lasdon@mccombs.utexas.edu)
[c] Dept of CIS/QMST, McCoy College of Business Administration, Texas State University, USA (jcplummer@austin.rr.com)
[d] Departamento de Ciencias de la Computación, Universidad Rey Juan Carlos, Spain (Abraham.Duarte@urjc.es)
[e] Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Spain (Rafael.Marti@uv.es)
[f] Leeds School of Business, University of Colorado, Boulder, CO 80309, USA (Laguna@colorado.edu)
[g] School of Business Administration, University of Mississippi, University, MS 38677, USA

1. Introduction

We consider the constrained global optimization problem (*P*) expressed in the following general form:

(*P*) minimize $f(x)$
subject to:
$$G(x) \leq b$$
$$x \in \mathbb{R}^n$$

where *x* is an *n*-dimensional vector of decision variables, *G* is an *m*-dimensional vector of constraint functions, and without losing generality the vector *b* contains upper bounds for these functions. The set *S* is defined by simple bounds on *x*, and we assume that it is closed and bounded, i.e., that each component of *x* has a finite upper and lower bound.

We introduce strategies for solving (*P*) which are based on pseudo-cuts, consisting of linear inequalities that are generated for the purpose of strategically excluding certain points from being admissible as solutions to an optimization problem. The *pseudo* prefix refers to the fact that these inequalities may not be valid in the sense of guaranteeing that at least one globally optimal solution will be retained in the admissible set. Nevertheless, a metaheuristic procedure that incorporates occasional invalid inequalities with a provision for replacing them can yield an aggressive solution approach that can prove valuable in certain settings. The use of pseudo-cuts to create temporary restrictions in a search process was suggested in Glover (1989) in the context of a tabu search procedure. In this approach the cuts are treated in the same way as other restrictions imposed by tabu search, by drawing on a memory-based strategy to cull out certain cuts previously introduced and drop them from the pool of active restrictions. The present approach is particularly motivated by the work of Lasdon, et al. (2010), where a simplified instance of such strategies was found to be effective for improving the solution of certain constrained non-convex nonlinear continuous problems.

In the present paper we likewise assume the objective function of (*P*) is non-convex (hence a local optimum may not be a global optimum), and allow for non-convexity in the constraints. We also allow for the presence of integer restrictions on some of the problem variables under the provision that such variables are treated by means of constraints or objective function terms that permit them to be treated as if continuous within the nonlinear setting. In the case of zero-one variables, for example, a concave function such as $x_j(1 - x_j)$ may be used that is 0 when $x_j = 0$ or 1, and is positive otherwise. See Bowman and Glover (1972) for additional examples.

We make recourse to an independent algorithm to generate trial solutions to be evaluated as candidates for a global optimum, where as customary the best feasible candidate is retained as the overall "winner". The independent algorithm can consist of a directional search (based on gradients or related evaluations) as in Lasdon et al. (2010), or may be a "black box" algorithm as used in simulation optimization as in April et al. (2006) and Better et al. (2007).

2. Pseudo-Cut Form and Representation

Our pseudo-cut strategy is based on generating hyperplanes that are orthogonal to selected rays (half-lines) originating at a point x′ and passing through a second point x″, so that the hyperplane intersects the ray at a point x^o determined by requiring that it lies on the ray at a selected distance d from x′. The half-space that forms the pseudo-cut is then produced by the associated inequality that excludes x′ from the admissible half-space. We define the distance d by reference to the Euclidean (L2) norm, but other norms can also be used.

To identify the pseudo-cut as a function of x′, x″ and d, we represent the ray that originates at x′ and passes through x″ by

$$x = x' + \lambda(x'' - x'), \qquad \lambda \geq 0 \tag{1}$$

(Hence x′ and x″ lie on the ray at the points determined by λ = 0 and 1, respectively.)

A hyperplane orthogonal to this line may then be expressed as.

$$ax = b \tag{2.1}$$

where

$$a = (x'' - x') \tag{2.2}$$
$$b = \text{an arbitrary constant} \tag{2.3}$$

The specific hyperplane that contains a given point x^o on the ray (1) results by choosing

$$b = ax^0 \tag{2.4}$$

To identify the point x^o that lies on the ray (1) at a distance d from x′, we seek a value $\lambda = \lambda^o$ that solves the equation

$$d(x', x'') = \|x' - x^0\| = d \tag{3.1}$$

where

$$x^0 = x' + \lambda^0(x'' - x') \tag{3.2}$$

Consequently, by the use of (3.2) the desired value of λ^o is obtained by solving the equation

$$\sqrt{\sum_j \left(x_j' - x_j^0\right)^2} = d \tag{3.3}$$

For the value of λ^o and the hyperplane thus determined, the associated half-space that excludes x″ (and x′) is then given by

$$ax \geq ax^0 \qquad (4)$$

3. Pseudo-Cut Strategy

We make use of the pseudo-cut (4) within a 2-stage process. In the first stage x′ represents a point that is used to initiate a current search by the independent algorithm, and x″ is the point obtained at the conclusion of this search phase (e.g., x″ may be a local optimum). The distance d is then selected so that x^o lies a specified distance beyond x″.

In the second stage we take x′ to be the point x″ identified in the first stage, and determine x″ by applying the independent algorithm to the problem that results after adding the pseudo-cut generated in the first stage. In this case d is chosen so that x^o lies between x′ and x″ at a selected distance from x′.

The value of d in both of these cases may be expressed as a multiple m of the distance between the points currently denoted as x′ and x″, i.e.

$$d = m\|x'' - x'\| \qquad (5)$$

The multiple m is selected to be greater than 1 in the first stage and less than 1 in the second. Because the points x′ and x″ change their identities in the two stages, it is convenient to refer to the points generated in these stages by designating them as P0, P1, Q1, etc., as a basis for the following description. (We later identify additional variations based on choosing d, x′ and x″ in different ways.) The *pseudo-cut pool* (or simply *cut pool*) refers to all pseudo-cuts previously added that have not yet been discarded. The pool begins empty.

Together with the statement of the Pseudo-Cut Generation Procedure, we include parenthetical remarks, underlined and in italics, that identify specific accompanying diagrams to illustrate some of the key steps of the procedure. A complete pseudo-code of this procedure appears in the Appendix.

Pseudo-Cut Generation Procedure[1]

Stage 1:

(1.1) Let $x' = P0$ denote a starting point for the independent algorithm, let $x'' = P1$ denote the best point obtained during the current execution of the algorithm, and let $x^o = Q1$ be the point determined by (3) upon selecting a value $m > 1$ in (5). (See Note 1.) If x^o violates any pseudo-cut contained in the cut pool, remove this cut from the pool.

(1.2) Add the pseudo-cut (4) to the cut pool and apply the independent algorithm starting from the point Q1. Let Q2 denote the best point of the current execution. If Q2 = Q1, then increase the value of m to determine a new Q1 by (3) that replaces the previous cut that was generated for a smaller m value, and then repeat step (1.2) (without increasing an iteration counter). Otherwise, if Q2 differs from Q1, proceed to step (1.3). (See Note 2.)

(1.3) If Q2 does not lie on the hyperplane $ax = ax^o$ associated with the current pseudo-cut (4) then redefine P0 = Q1, P1 = Q2, and return to step (1.1). *(Fig. 1(a) shows this case and Fig. 1(b) shows this case after returning to step (1.1).)* Otherwise, if Q2 lies on $ax = ax^o$, then proceed to Stage 2. (See Note 3.) *(Fig. 1(c) shows this case.)*

Stage 2:

(2.1) Remove the pseudo-cut (4) just added in step (1.2) and replace it with a new one determined as follows. Let $x' = P1$ and $x'' = Q2$, and determine a point $x^o = R1$ by (3) and (5), where m is chosen to satisfy $1 > m > 0$. (See Note 4 for choosing m large enough but less than 1.) If x^o violates any pseudo-cut contained in the cut pool, remove this cut from the pool.

(2.2) Add the new pseudo-cut (4) to the cut pool and apply the independent algorithm starting from the point R1. Let R2 denote the best point of the current execution. (a) If R2 = R1, then redefine P0 = Q1, P1 = Q2. Otherwise, (b) if $R2 \neq R1$ *(Diagram 2.1 shows this case)*, then whether or not R2 lies on the cut hyperplane, redefine P0 = P1 and P1 = R2. In either case (a) or (b), return to step (1.1) of Stage 1. (See Note 5.) *(Diagram 2.1.1 shows this case, inherited from (b), while Fig. 1(f) shows the case inherited from (a). Both of these two diagrams also show the new P0, P1 and Q1, and the new pseudo-cut produced at step (1.1).)*

We observe that each time the method returns to step (1.1) in the Pseudo-Cut Generation Procedure, whether from step (1.3) or step (2.2), the current designation of P0 and P1 is compatible with the original designation, i.e., P0 always represents a point that has been used to start the independent algorithm and P1 represents the resulting best solution found on the current (most recent) execution of the algorithm.

[1] A complete pseudo-code for this procedure appears in the Appendix.

We also remark that when the method specifies that the independent algorithm should start from Q1 in step (1.2) or from R1 in step (2.2), it may be preferable to start the method from a point slightly beyond this intersection with the current pseudo-cut hyperplane, to avoid numerical difficulties that sometimes arise in certain nonlinear methods if starting solutions are selected too close to the boundaries of the feasible region.

Illustrative Diagrams

The diagrams that illustrate several main components of the procedure are as follows.

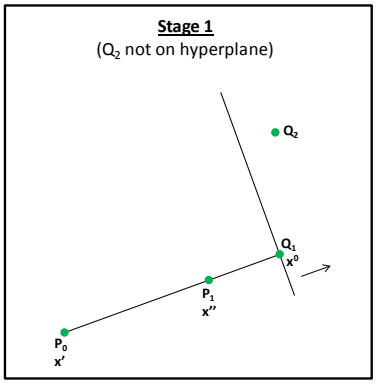

Fig. 1(a)

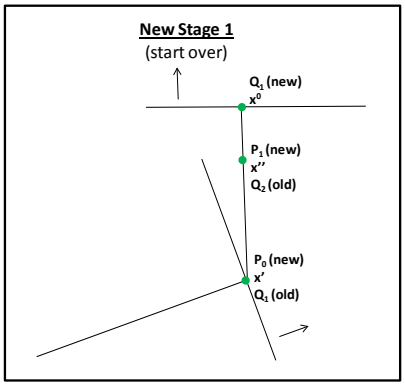

Fig. 1(b)

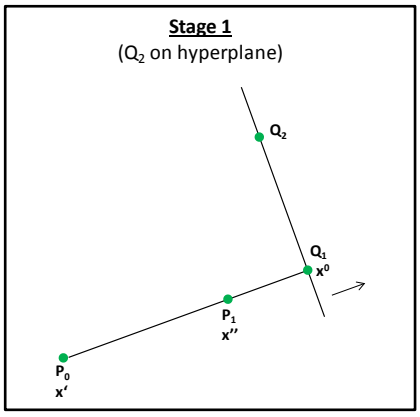

Fig. 1(c)

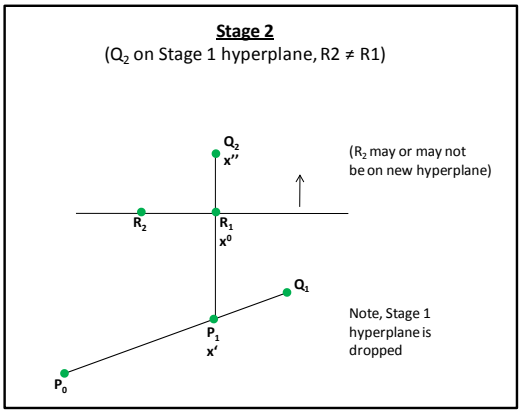

Fig. 1(d)

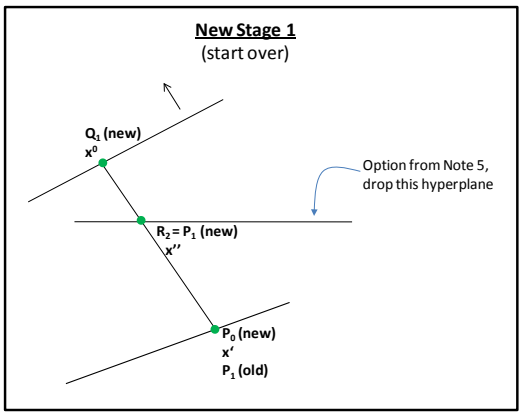

Fig. 1(e)

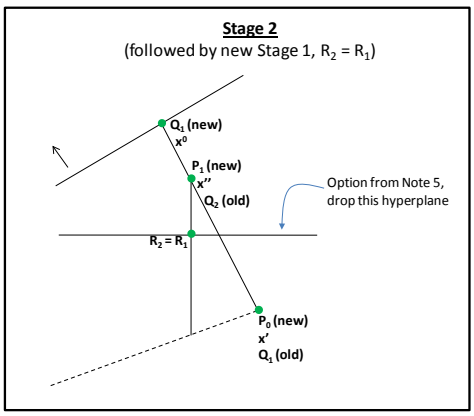

Fig. 1(f)

A Rule for Dropping Pseudo-Cuts: We allow for pseudo-cuts to be dropped (removed from the cut pool) by a rule that goes beyond the simple provision for dropping cuts already specified in the algorithm. We consider the pseudo-cuts to have the same character as tabu restrictions that are monitored and updated in the short term memory of tabu search. We propose the use of two tabu tenures t_1 and t_2 for using such memory, where t_1 is relatively small (e.g., $1 \leq t_1 \leq 5$) and t_2 is selected to be larger (e.g., $7 \leq t_2 \leq 20$). (The indicated ranges are for illustrative purposes only.) Each pseudo-cut not dropped by the instructions stipulated in the algorithm will be retained for t_1 iterations (executions of step (1.1)) after the cut is created, and then dropped after this number of iterations whenever the cut becomes non-binding (the current solution x" produced by the independent algorithm does not lie on the cut hyperplane). However, on any iteration when no cut is dropped (either directly by the algorithm or by this rule), a second rule is applied by considering the set of all cuts that have been retained for at least t_2 iterations. If this set is non-empty, we drop oldest cut from it (the one that has been retained for the greatest number of iterations).

The following additional observations are relevant.

Note 1. The values chosen for m are a key element of the cut generation strategy in its present variation, and will depend on such things as the sizes of basins of attraction in the class of problem considered. Within step (1.1), m may be chosen to be a selected default fraction greater than 1, but bounded from below by a value that assures x^o will lie a certain minimum distance beyond x".

Note 2. To avoid numerical problems, it is appropriate to require that Q2 differ from Q1 by a specified amount in step (1.2) in order to be considered “not equal” to Q1. Also, the increase in the value of m in step (1.2) can be chosen either as a default percentage increase or as an amount sufficient to assure that d grows by a specified value independent of this percentage. This value of m drops back to its original value whenever the method re-visits step (1.1), but if a succession of increases in step (1.2) causes the distance separating Q1 from P1 to exceed a specified threshold (anticipated to render all feasible solutions for the original problem inadmissible relative to the pseudo-cut (4) at step (1.2)), then the procedure may be terminated or re-started from scratch from a new

initial starting solution x′ = P0 produced by a multi-start procedure, e.g. as described in Ugray, et. al (2009).

Note 3. In step (3.3) we require the point Q2 to lie a certain minimum distance from the hyperplane $ax = ax^o$ in order to be considered as not lying on the hyperplane.

Note 4. The value of m in step (2.1) is assumed to be chosen to prevent the point Q1 from satisfying the pseudo-cut (4) produced in step (2.2). It suffices to choose m so that the distance of R1 from P1 is as least as great as the distance of Q1 from P1. (If this distance is the same, then R1 and Q1 will lie on a common hyper-sphere whose center is P1, and the pseudo-cut (4) of (2.2) is produced by a tangent to this hyper-sphere.)

Note 5. An interesting possible variation in Step (2.2) that reduces the number of pseudo-cuts maintained, and hence constrains the search space less restrictively, is to drop the latest pseudo-cut (4) (that led to determining R2) before returning to (1.1) to generate the new pseudo-cut. (The cut thus dropped is not immediately relevant to the next step of the search in any event.) Another variation is to make sure that d is large enough to render the most recent Q2 infeasible relative to the pseudo-cut. This variation will avoid cases where sometimes Q2 may be revisited as a local optimum. (The procedure may be monitored to see if multiple visits to the same Q2 point occur, as a basis for deciding if the indicated variation is relevant.)

Finally, we observe that a simplified version of the Pseudo-Cut Generation Procedure can be applied that consists solely of Stage 1, with the stipulation in step (1.3) that the pseudo-cut (4) is generated and the method returns to (1.1) in all cases.

4. Determination of the distance *d* by exploiting quick objective function and directional evaluations.

In a context where a computational method exists that can relatively quickly calculate the objective function value for the point x^o, and in addition can fairly quickly calculate whether a given direction is an improving direction, the value of d that determines x^o can be determined implicitly rather than explicitly.

This is done by generating a number of successively larger candidate values for the scalar weight λ^o, starting from $\lambda^o > 1$ for Step (1.1) of the Pseudo-Cut Generation Procedure, and starting from $\lambda^o > 0$ otherwise. For each candidate value of λ^o, we then check whether one or more of the following conditions hold for the associated x^o vector. (It is assumed that terms like *feasible improving direction* and *stronger improving direction* are understood and need not be defined.)

> *Condition 1(a*). There exists a feasible improving direction from x^o that lies in the region satisfying the pseudo-cut (4).

Condition 1(b). The direction from x^o on the ray for $\lambda > \lambda^o$ is a feasible improving direction.

Condition 2(a). The improving direction from Condition 1 (for a given choice of 1(a) or 1(b)) is stronger than any feasible improving direction that does not lie in the region satisfying the pseudo-cut (4).

Condition 2(b). The improving direction from Condition 1 (for a given choice of 1(a) or 1(b)) is stronger than the direction from x^o on the ray for $\lambda < \lambda^o$ (automatically satisfied the latter is not a feasible improving direction).

The conditions 1(b) and 2(b) are more restrictive than 1(a) and 2(a), respectively, but are easier to check. Condition 2 is evidently more restrictive than Condition 1.

For a selected condition, we then choose the first (smallest) candidate λ^o value (and associated x^o) for which the condition is satisfied. This choice then indirectly determines the distance d.

5. Choosing the points x′ and x″

We have previously indicated that x′ is customarily chosen as a point that initiates the search of the independent algorithm, and x″ denotes the best point determined on the current pass of the algorithm, as where x″ may denote a local optimum. We now consider other choices that can be preferable under various circumstances.

It is possible, for example, that an effort to determine a point x^o according to Condition 1 or 2 of the preceding section will not be able to identify a feasible point that qualifies. In this case, it may be preferable to reverse the roles of x′ and x″ to seek a qualifying x^o on the ray leading in the opposite direction. Moreover, it may be worthwhile to examine the option of reversing the roles of x′ and x″ in any event, where the ultimate choice of which point qualifies as x′ will depend on the evaluation of the point x^o that is generated for each case.

Still more generally, the collection of candidate points from whose members a particular pair of points x′ and x″ will be chosen can be generated by a variety of considerations, including those used in composing a Reference Set in Scatter Search (see, for example, Glover, Laguna and Marti (2000) and Marti, Glover and Laguna (2006)). Likewise the criteria for selecting x′ and x″ from such a collection can also incorporate criteria from Scatter Search. Here, however, we suggest three alternative criteria.

Criterion 1. Let x′(i) and x″(i), i = 1,…,i*, identify the points used to determine previous pseudo-cuts (i.e., those successfully generated and introduced at some point during the search). Let x*(i) identify the point on the ray from x′(i) through

x″(i) that lies a unit distance from x′(i). Finally for a candidate pair of points x′ and x″, let x* denote the point on the ray from x′ through x″ that lies a unit distance from x′. From among the current pairs x′ and x″, we select the one such that x* maximizes the minimum distance from the points x*(i), i = 1,…,i*.

Criterion 2. Choose the candidate pair x′ and x″ by the same rule used in Criterion 1, except that x*(i) is replaced by the point x^o(i) (the "x^o point" previously determined from x′(i) and x″(i)), and x* is likewise replaced by the point x^o determined from the currently considered x′ and x″.

Criterion 2 allows for the possibility that x′ and x″ may lie on the same ray as generated by some pair x′(i) and x″(i), provided the point x^o lies sufficiently distant from the point x^o(i). This suggests the following additional criterion.

Criterion 3. Employ Criterion 1 unless the minimum distance of the selected point x* from the points x*(i), i = 1,…,i* falls below a specified threshold, in which case employ Criterion 2.

A variant on Criterion 3 is to employ Criterion 1 except where the minimum distance determined from Criterion 2 exceeds a certain lower bound, where this latter may be expressed in terms of the minimum distance obtained for Criterion 1.

6. Additional Considerations for Choosing x^o

To this point we have assumed that x^o will lie beyond x″ on the ray leading from x′ through x″, on each execution of Step (1.1) of the Pseudo-Cut Generation Procedure. However, in some case, as in the customary application of Scatter Search, it may be preferable to select a point x^o that lies between x′ and x″. We add this possibility as follows.

First, we stipulate that the candidate values for λ^o lie in the interval $0 < \lambda^o < 1$. Second, we apply Condition 1 or Condition 2 (in either the (a) or (b) form)) to determine a value λ^o_{min} which is the least λ^o value that satisfies the condition (assuming such a value exists in the interval in the interval $0 < \lambda^o < 1$). Next, we examine the candidate λ^o values in the reverse direction (from larger to smaller) in the interval $\lambda^o_{min} < \lambda^o < 1$, and choose one of the following Reverse Conditions as a basis for choosing a particular candidate value.

Reverse Condition 1(a). There exists a feasible improving direction from x^o that lies in the region not satisfying the pseudo-cut (4).

Reverse Condition 1(b). The direction from x^o on the ray for $\lambda < \lambda^o$ is a feasible improving direction.

Reverse Condition 2(a). The improving direction from Reverse Condition 1 (for a given choice of 1(a) or 1(b)) is stronger than any feasible improving direction that lie sin the region satisfying the pseudo-cut (4).

Reverse Condition 2(b). The improving direction from Reverse Condition 1 (for a given choice of 1(a) or 1(b)) is stronger than the direction from x^o on the ray for $\lambda > \lambda^o$ (automatically satisfied if the latter is not a feasible improving direction).

Finally, we identify the first (largest) λ^o candidate value satisfying the selected Reverse Condition, denoted by λ^o_{max} (provided such a value exists in the indicated interval), and choose $\lambda^o = (\lambda^o_{min} + \lambda^o_{max})/2$. This final λ^o value is the one used to find a point strictly between between x' and x'' from which to launch a new search. This search can optionally be constrained by adding a pseudo-cut (4) for x^o determined from $\lambda^o = \lambda^o_{min}$ (or from a "reverse" pseudo-cut determined from $\lambda^o = \lambda^o_{max}$).

From among the various candidate values x^o identified for launching a new search as above, and also from among those that may be identified from applying Condition 1 or 2 for $\lambda^o > 1$ (allowing x' and x'' to be interchanged), one may ultimately choose the option such that x^o receives a highest evaluation. This evaluation can be in terms of objective function value (possibly considering directional improvement), or in terms of maximizing the minimum distance of x^o from points in a Reference Set. By such a use of a Reference Set, the approach can foster diversity in conjunction with the search for improvement. In fact, the indicated strategies can be used to create rules for a version of Scatter Search that differs from more customary forms of the method. It should be noted that these strategies for choosing x^o vectors can be used without bothering to introduce pseudo-cuts. For example, such a strategy can be employed for some initial duration of search to produce x^o trial solutions, and then the pseudo-cuts can subsequently be invoked to impose greater restrictiveness on the search process.

7. Conclusion

The proposed collection of pseudo-cut strategies for global optimization expands the options previously available for guiding solution processes for non-convex nonlinear optimization algorithms. These strategies can be used to supplement other approaches for solving such problems, or can be used by themselves. The mechanisms proposed for generating trial solutions can similarly be used in a variety of ways, and may even be used independently of the pseudo-cuts themselves. The demonstration that an exceedingly simplified instance of a pseudo-cut strategy succeeded in enhancing a non-convex optimization method in Lasdon, et al. (2010) suggests the potential value of more advanced pseudo-cut strategies as described here, and of empirical studies for determining which combinations of these strategies will prove most effective in practice. The use of pseudo-cuts reinforces the theme of joining mathematically based exact methods for convex problems with special strategies capable of modifying these methods to enable them to solve non-convex problems. In this guise, the proposals of this paper offer a chance to create a wide range of new hybrid algorithms that marry exact and metaheuristic procedures.

References

April, J., M. Better, F. Glover, J. Kelly and M. Laguna (2006) “Enhancing Business Process Management with Simulation-Optimization,” *Proceedings of the 2006 Winter Simulations Conference*, L.F. Perrone, F.P. Wieland, J.Liu, B.G. Lawson, D.M. Nicol, and R.M. Fujimoto, eds.

Better, M., F. Glover and M. Laguna (2007) "Advances in Analytics: Integrating Dynamic Data Mining with Simulation Optimization," *IBM Journal of Research and Development*, Vol. 51, No. 3/4, pp. 477-487.

Bowman, V.J. and F. Glover (1972) “A Note on Zero-One Integer and Concave Programming,” *Operations Research*, Vol. 20, No. 1, pp. 182-183.

Glover, F. (1989) "Tabu Search - Part I," *ORSA Journal on Computing*, Vol. 1, No. 3, pp. 190-206.

Glover, F., M. Laguna and R. Marti (2000) "Fundamentals of Scatter Search and Path Relinking," *Control and Cybernetics*, Vol. 29, No. 3, pp. 653-684.

Lasdon, L., A. Duarte, F. Glover, M. Laguna and R. Marti (2010) “Adaptive Memory Programming for Constrained Global Optimization,” *Computers and Operations Research 37*, pp. 1500-1509.

Martí, R., F. Glover and M. Laguna (2006) “Principles of Scatter Search,” *European Journal of Operational Research* 169, pp. 359-372.

Ugray, Z., L. Lasdon, J. Plummer, and M. Bussieck (2009), “Dynamic filters and Randomized Drivers for the Multi-start Global Optimization Algorithm MSNLP,” *Optimization Methods and Software 24*, pp. 635-656.

Appendix – Pseudo-code for the Pseudo-Cut Method (Initial Simplified Version)

Initialization

1. Let *TabuCutList* be the memory list of pseudo-cuts with *TabuTenure* size
2. Let m = 0.1, *MaxIter* = 5 and *GlobalIter*= 20
   Maxpert = 5 and *Iter1* = 0. % Maxpert is a limit on *pert*
   % The preceding values in 2. are suggestive only
3. Let $P0$ be a *random point and Best_f = f(P0)*

While (*Iter*1 < *GlobalIter*)

4. *TabuCutList* = ∅
5. *improved* = FALSE
6. *pert* = *m* and *Iter2 = 0*
7. $P1$ = LS($P0$, *f(x),G,S*)

If (*f*($P1$) < *Best_f*)

8. *Best_f* = *f*($P1$)

While (*Iter*2 < *MaxIter* **and** ||P1 – P0|| > *minDist*)

//Begin Stage 1

9. $Q1 = P0 + (1 + pert)*(P1 - P0)$
10. Remove from *TabuCutList* the cuts violated at $Q1$. If no cuts are removed and if there are any cuts retained more than t_2 iterations, drop the oldest. (Disregard this last instruction if 9 and 10 are reached from 16, below.)
11. Add to *TabuCutList* the cut *pcut*($Q1$): $(P1 - P0)\, x \geq (P1 - P0)\, Q1$
12. $Q2$ = LS($Q1$, *f(x),G*$\cup$*TabuCutList, S*)

If (*f*($Q2$) < *Best_f)*

13. *improved* = TRUE
14. *Best_f = f*($Q2$); *P0 = Q1; P1 = Q2; pert = m*
15. go to 22

If (||*Q1 – Q2*|| < *minDist*)

16. Drop the cut just added in 11, set *pert = pert + m*. If *pert* > *Maxpert*, go to 23. Otherwise, return to 9.

Else

17. *pert = m*

If ($Q2$ does not satisfy *pcut*($Q1$) with equality (**and** $\|Q2 - Q1\| >$ *minDist*))

18. $P0 = Q1$; $P1 = Q2$; Proceed to 22 (increase Iter2 and repeat Stage 1)

//**Begin Stage 2**

(Here $Q2$ satisfies *pcut*($Q1$) with equality and $\|Q2 - Q1\| >$ *minDist*)

19. Drop the cut just added in 11 and record P01 = P1 and Q01 = Q1.
20. $P0 = P1$; $P1 = Q2$; $Q1 = P0 + (1 - pert)*(P1 - P0)$
21. Execute instructions 10 – 15, but without dropping any cuts in 10 other than those violated at Q1.

    If (||*Q1 – Q2*|| < *minDist*)

    21.1. P0 = Q01 (P1 is unchanged);

    Else (||*Q1 – Q2*|| ≥ *minDist*)

    21.2. P0 = P01; P1 = Q2

22. *Iter*2 = *Iter*2 + 1 (Return to 9 if *Iter2* < *MaxIter*.)

23. Generate a new point $P0$ by a diversification step, *Iter1=Iter1+1* (and return to 4 if *Iter1* < *GlobalIter*)