2008 20th IEEE International Conference on Tools with Artificial Intelligence

Finding Good Starting Points For Solving Structured and Unstructured Nonlinear Constrained Optimization Problems*

Soomin Lee and Benjamin Wah
Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois, Urbana-Champaign
Urbana, IL 61801, USA
{lee203,wah}@illinois.edu

Abstract

In this paper, we develop heuristics for finding good starting points when solving large-scale nonlinear constrained optimization problems (COPs). We focus on nonlinear programming (NLP) and mixed-integer NLP (MINLP) problems with nonlinear non-convex objective and constraint functions. By exploiting the highly structured constraints in these problems, we first solve one or more simplified versions of the original COP, before generalizing the solutions found by interpolation or extrapolation to a good starting point. In our experimental evaluations of 190 NLP (resp., 52 MINLP) benchmark problems, our approach can solve 97.9% (resp., 71.2%) of the problems using significantly less iterations from our proposed starting points, as compared to 85.3% (resp., 46.2%) of the problems solvable by the best existing solvers from their default starting points.

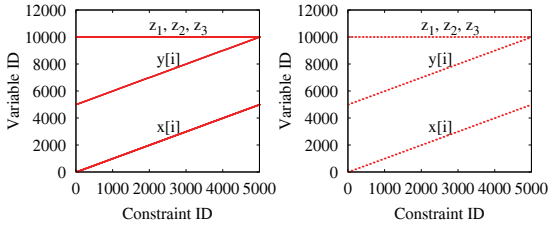
1 Introduction

The NLP and MINLP problems studied in this paper have nonlinear and non-convex objective and constraint functions and are formulated as follows:

$$(P_m): \quad \min_{\mathbf{z}} \quad f(\mathbf{z}) \qquad (1)$$
$$\text{subject to} \quad \mathbf{h}(\mathbf{z}) = \mathbf{0} \text{ and } \mathbf{g}(\mathbf{z}) \leq \mathbf{0},$$

where $\mathbf{z} = (\mathbf{x}, \mathbf{y})$, and $\mathbf{x} \in \mathbb{R}^v$ and $\mathbf{y} \in \mathbb{D}^w$ are, respectively, vectors of continuous and discrete variables; $f(\mathbf{z})$ is an objective function; $\mathbf{h}(\mathbf{z}) = (h_1(\mathbf{z}), \ldots, h_m(\mathbf{z}))^T$ is a vector of m equality constraint functions; and $\mathbf{g}(\mathbf{z}) = (g_1(\mathbf{z}), \ldots, g_r(\mathbf{z}))^T$ is a vector of r inequality constraint functions. Because no closed-form solution to (1) exists, we focus on finding its constrained local minima.

Consider ORTHRGDS, an NLP from the CUTEr benchmark suite. Its goal is to find values of $x[i]$, $y[i]$, z_1, z_2, and z_3 that minimize $f(\mathbf{z})$ and that satisfy $N = 5000$ non-convex equality constraints $h_i(\mathbf{z}) = 0$ for $i = 1, \ldots, N$:



a) NLP ORTHRGDS b) 50 sampled constraints

Figure 1. The regular constraint structure of ORTHRGDS

$$\min_{\mathbf{z}} f(\mathbf{z}) = \sum_{i=1}^{N} \left((x[i] - xd[i])^2 + (y[i] - yd[i])^2 \right) \qquad (2)$$
$$\text{subject to } h_i(\mathbf{z}) = 0 \quad \text{for } 1 \leq i \leq N,$$

where

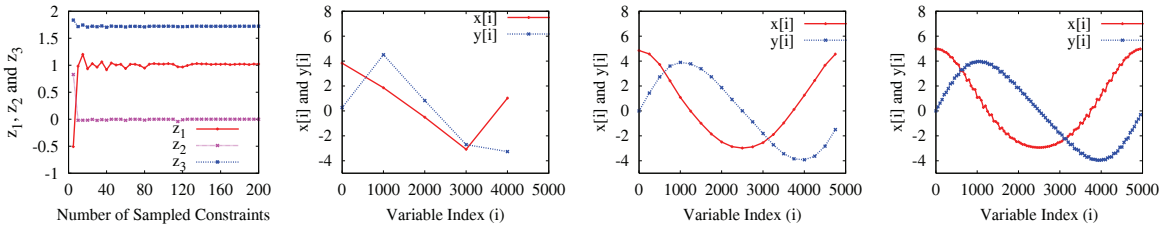
$$\begin{aligned} h_i(\mathbf{z}) &= \left((x[i] - z_1)^2 + (y[i] - z_2)^2 \right)^2 \\ &\quad - \left((x[i] - z_1)^2 + (y[i] - z_2)^2 \right) (1 + z_3^2)^2, \\ xd[i] &= (C_1 + \cos(p(i)) \cos(p(i))(1 + C_2 \cos(C_3 p(i))), \\ yd[i] &= (C_1 + \cos(p(i)) \sin(p(i))(1 + C_2 \cos(C_3 p(i))), \\ p(i) &= 2\pi(i-1)/N, \end{aligned}$$

and $C_1 = 1 + 1.7^2$, $C_2 = 0.2$, and $C_3 = 237.1531$. This problem cannot be solved by SNOPT [2] and Lancelot, but can be solved by Knitro [1] and Ipopt [5].

ORTHRGDS belongs a large class of benchmark problems specified by an algebraic modeling language, like AMPL (`http://www.ampl.com`) and GAMS (`http://www.gams.com`) that uses indexes to define groups of variables and constraints with some common properties. Indexing is useful when specifying large-scale problems because it will be cumbersome to give a unique name for every variable and constraint. The use of indexed constraints

*Proceedings of the IEEE International Conf. on Tools for AI, 2008.

1082-3409/08 $25.00 © 2008 IEEE
DOI 10.1109/ICTAI.2008.53



a) z_1, z_2 and z_3 b) 5 sampled constraints c) 20 sampled constraints d) 100 sampled constraints

Figure 2. An illustration of finding good starting points. a) The values of z_1, z_2, z_3 found from the simplified problem; b) - d) The variables converge with an increasing number of sampled constraints.

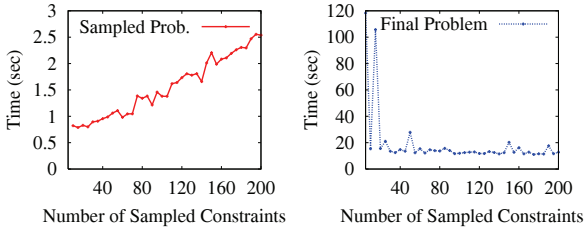
leads to a large number of similar and related constraints.

Figure 1a illustrates the regular structure of the constraints in ORTHRGDS. It shows a dot where a constraint on the x-axis is related to a variable on the y-axis. In particular, the i^{th} constraint $h_i(\mathbf{z}) = 0$ involves variables $x[i]$ and $y[i]$ with index i and common variables z_1, z_2, and z_3.

In this paper, we show for COPs with good structural properties that it is possible to find good starting points by exploiting their structures. Many COPs are supplied with default *starting points*, which can be used to test the ability of individual solvers to solve them. Our approach is to generate better starting points from those defaults and to show improved solution times or quality in solving the COPs.

As an illustration, consider a simplified version of ORTHRGDS obtained by uniformly sampling the constraints in Figure 1a. (See Figure 1b where 50 constraints are uniformly sampled.) Figure 2a depicts the values of z_1, z_2, and z_3 when solving by SNOPT the simplified problem of various numbers of sampled constraints. Although there are small fluctuations in their values, the variables converge to some stable values when 20 or more constraints are sampled. This convergence behavior is also extended to the other variables $x[i]$ and $y[i]$. Figures 2b-2d shows the values of $x[i]$ and $y[i]$ for three simplified problems, each with a different number of sampled constraints. The graphs show that the values of the variables of the same index are highly related. This *smoothness* property is typical in most of the AMPL benchmarks whose constraints are grouped in several forms and the parameters of the constraints in each group are closely related.

Using the starting points found by solving ORTHRGDS with 20 sampled constraints and by interpolating the values of the other variables (Figure 2c), we can solve ORTHRGDS in significantly fewer iterations and CPU times (8 iterations/1.635 sec. by Ipopt and 7 iterations/1.604 sec. by Knitro) when compared to those using the default starting point (16 iterations/3.359 sec. by Ipopt and 47 iterations/9.555 sec. by Knitro). The overhead for solving the problem with 20 sampled constraints is negligible (9 iterations/0.018 sec. by Ipopt and 5 iterations/0.00275 sec. by Knitro). Our approach also leads to a better solution than



a) Sampled problems b) Original problem

Figure 3. Trade-offs between the time for solving a problem with sampled constraints and the time for solving the original problem using the starting point found.

Knitro (1523.90 versus 1776.23).

There is a trade-off between the time for solving a problem with sampled constraints (which increases as the number of constraints increases−Figure 3a), and the time for solving the original problem using the starting points found (which decreases when the number of sampled constraints increases and the quality of the starting point found improves−Figure 3b). Because the time for solving a sampled problem is usually much smaller, we can simply choose the number of sampled constraints based on the convergence of the variables found. For example, Figure 2 shows that the solutions of ORTHRGDS converge when 20 or more constraints are sampled.

The importance of using good starting points is also illustrated in solving EIGENA2 from the CUTEr benchmark suite. EIGENA2 is easily solvable by SNOPT when the matrix variable $q[i, j]$, where $1 \leq i, j \leq 10$, is initialized to an identity matrix (the default). However, SNOPT fails to find a feasible solution when $q[i, j]$ is initialized to a zero matrix.

To show that many existing benchmarks have regular structures that can be exploited in finding good starting points, we have exhaustively tested the CUTEr, NLP, Nonsys, COPS, MacMINLP and MINLP Library suites (Table 1). Among those tested, we eliminate unconstrained, linear, or quadratic problems, and small problems with less than 50 variables or constraints. These result in 242 (190 NLP and 52 MINLP) benchmarks studied in this paper.

Table 1. NLP and MINLP benchmark suites studied in this paper.

CUTEr	Nonlinear optimization (revisited)	`http://www.numerical.rl.ac.uk/cuter-www/`
NLP	AMPL NLP benchmarks	`http://plato.asu.edu/ftp/ampl-nlp.html`
NonSys	Nonlinear systems of equations	`http://plato.asu.edu/ftp/ampl_files/nonsys_ampl/`
COPS	Large-scale optimization problems	`http://www-unix.mcs.anl.gov/~more/cops/`
MacMINLP	AMPL collection of MINLP	`http://www-unix.mcs.anl.gov/~leyffer/macminlp/`
MINLP Library	GAMS MINLP test models	`http://www.gamsworld.org/minlp/minlplib.htm`

Our main contribution in this paper is on the development of heuristics for finding good starting points that can be used for solving NLPs and MINLPs. Without a good starting point, a search may get stuck and does not converge. We describe in Section 2 our methods for sampling the constraints of indexed as well as non-indexed COPs. Section 3 summarizes some properties of the starting points found. We present in Section 4 our experimental results.

2 Generation of Good Starting Points

Our approach is to solve one or more simplified versions of a COP, whose solutions can be generalized to provide a good starting point to the original COP. In this section, we present three methods for handling indexed problems, followed by methods for discovering the structures of non-indexed problems.

Finding starting points for indexed problems. To effectively analyze the constraint structure of an indexed COP written in AMPL, we have developed a recursive descent parser (using a Perl module *ParseRecDescent* from CPAN: `http://www.cpan.org`) that can automatically parse its representation. Using the parser to identify the variables in the constraints, our system automatically generates one or more simplified AMPL problems with sampled constraints and solves them in order to find good starting points. The three methods described below are based on the unique structures discovered in these problems.

a) Constraint sampling and interpolations. For NLPs whose constraints are independent except for a few global shared variables, the constraints can be uniformly sampled in order to generate one or more simpler versions of the original NLP that are coupled through the shared variables. Using the approach described in Section 1 for solving ORTHRGDS, the NLPs with sampled constraints are solved until the shared variables converge.

To avoid enumerating all possible combinations of sampled constraints, we determine the number of sampled constraints as follows. As is depicted in Figure 2, the variables converge to some stable values as more constraints are sampled. Therefore, we start solving a simplified problem using a very small number (N_1) of sampled constraints. We then gradually increase the number of sampled constraints (N_i) until convergence is observed. Let $\mathbf{v_i}$ be the vector containing the values of the variables after solving a simplified problem with N_i sampled constraints. We stop generating new problems when the following condition is reached:

$$(1-a)|\mathbf{v_{i-1}}| - K \;\leq\; |\mathbf{v_i}| \;\leq\; (1+a)|\mathbf{v_{i-1}}| + K, \quad (3)$$

where $0 < a < 1$ and K is a constant in case $|\mathbf{v_i}|$ is very close to zero. When the value of $|\mathbf{v_i}|$ is within the bounds $(1 \pm a)|\mathbf{v_{i-1}}|$ or when N_i exceeds 10% of the total constraints, we stop and use $\mathbf{v_i}$ as the starting point. This approach is intuitively sound, as we expect $\mathbf{v_i}$ to converge to some stable form as more constraints are sampled.

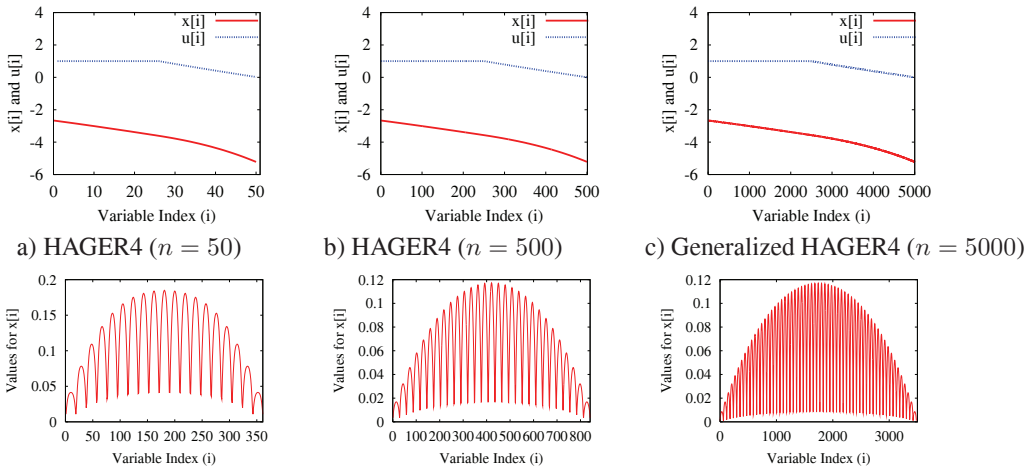
Using the values of variables found in the simplified NLP, we interpolate the values of those variables in the original NLP that are not solved in the simplified version. Based on the starting point found, we then solve the original NLP.

Overall, the approach is used in solving 34 (or 14.0%) of the 242 benchmarks tested (indicated by A in Table 2).

b) Constraint sampling and extrapolation. For an NLP whose constraints are related to their neighborhood constraints by some common variables, uniformly sampling the constraints of the problem may result in a new NLP with independent constraints that do not have common variables. For example, the problem HAGER4 from CUTEr has constraints of the form $(n-1)x[i]-nx[i-1]-\exp(t[i])u[i] = 0$ for $1 \leq i \leq 5000$, and the $i-1^{\text{st}}$ constraint is related to the i^{th} constraint by $x[i-1]$. This means that if every other constraint is sampled, then the resulting problem has independent constraints. In this case, solving the simplified problem will not lead to new insights on the starting point.

For these NLPs, instead of uniformly sampling the constraints, we generate a version using a subset of the contiguous constraints and eliminate those variables that are not in the sampled constraints. For example, in HAGER4, we generate a 500-constraint problem related to $x[0], \ldots, x[499]$. Figures 4a ($resp.$, 4b) shows the values of $x[i]$ and $u[i]$ found using the first 50 ($resp.$, 500) constraints.

After solving the simplified problem, we extrapolate the values of the variables found to those in the original problem. In the simplest case, the extrapolations can be done by assigning the value found for a variable in the simplified problem to the corresponding variable in the original problem and by interpolating the values of other variables in between. This works in HAGER4 when we assign $x[j]$, $j = 1, \ldots, 500$, in the simplified problem to $x[j']$, $j' = 10(j-1)+1$, in the original problem. Figure 4c shows the values of $x[i]$ and $u[i]$ that are extrapolated for the version of HAGER4 with 5,000 constraints.



a) HAGER4 ($n = 50$) b) HAGER4 ($n = 500$) c) Generalized HAGER4 ($n = 5000$)

d) EX1 ($n = 19$) e) EX1 ($n = 29$) f) Generalized EX1 ($n = 59$)

Figure 4. The common features of the solutions of two simplified versions of HAGER4 (*resp.*, EX1) and their generalization.

Overall, this approach is used in solving 119 (or 49.2%) of the 242 benchmarks (indicated by B1 in Table 2).

In more general cases, the solution of the simplified problem may be in a complex form, although the solution can be generalized using some features that have to be identified. For example, EX1 in the NLP suite has variables $x[i]$, $1 \leq i \leq n^2$ and $n = 59$. When we solved two simplified problems with $n = 19$ with $n = 29$, the solutions have, respectively, 19 and 29 peaks with a similar envelope (Figures 4d and 4e). When EX1 is solved with $n = 39$, we obtain a solution with 39 peaks and an envelope similar to that of $n = 29$. Hence, we generalize the solutions found to the initial solution of EX1 in Figure 4f, which has 59 peaks and the same envelope as that of $n = 39$.

Overall, this technique is used in solving 37 (or 15.3%) of the 242 benchmarks (indicated by B2 in Table 2).

c) Constraint relaxation for MINLPs. For MINLPs, we generate a good starting point by solving each as an NLP without the integrality requirement. We further apply the two approaches above in case that the relaxed MINLP is too large to be handled by an existing NLP solver.

This technique is used in solving 52 (or 21.5%) of the 242 benchmarks (indicated by C in Table 3).

Finding starting points for non-indexed problems. As is mentioned in Section 1, most of the problems written in AMPL or GAMS have indexed variables and constraints that can be exploited in generating simplified versions of these problems. However, some problems may not be indexed because they might have been converted from AMPL/GAMS to GAMS/AMPL by a converter. Moreover, the names of variables and constraints in some industrial benchmarks might have been concealed and replaced by some random names for security reasons.

Since the indexing of variables and constraints is critical when sampling constraints and when interpolating and extrapolating variables, we have developed techniques to recover the indexes of those non-indexed problems, assuming that these problems were originally indexed.

Using the parser we have developed for analyzing AMPL problems, we analyze each constraint in a non-indexed problem and group those of the same form together. Within a group of constraints, we extract variables and reorder them by their names. We further assume that variables at the same ordinal location in each group of constraints were indexed by the same variable name. After recovering the indexed form, we apply the same techniques earlier to generate good starting points. Note that this approach will not work if the variables were shuffled and randomly indexed.

As an example, consider CATMIX800 (with 2,403 variables and 1,600 constraints) from the NLP suite. The benchmark has two different types of constraints. After proper parsing, grouping, and reordering, its constraints can be represented in Figure 5a. We can rewrite this problem into an indexed form by assigning arbitrary names on variables and constraints. Here, variables $x_1 \ldots x_{801}$, $x_{802} \ldots x_{1602}$ and $x_{1603} \ldots x_{2403}$ are, respectively, assigned to $X[i]$, $Y[i]$ and $Z[i]$ for $0 \leq i \leq I$ and $I = 800$ (see Figure 5b). Lastly, we generate two simplified problems with $I = 100$ and $I = 200$ using technique B1 discussed in this section. Figure 6 shows the convergence behavior for the variables in CATMIX800 when I is extended from 100 to 200.

3 Analysis of the Starting Points Found

Due to space limitation, we informally analyze the quality of the starting points in this section.

a) Convergence of the objective function. Let p_i^* be the global minimum of P_i, a problem with i sampled con-

$$
\begin{array}{ll}
e_2: & x_{802} + x_{803} + C(x_1 x_{802} + x_2 x_{803} - 10(x_1 x_{1603} + x_2 x_{1604})) = 0 \\
 & \ldots \\
e_{801}: & x_{1601} + x_{1602} + C(x_{800} x_{1601} + x_{801} x_{1602} - 10(x_{800} x_{2402} + x_{801} x_{2403})) = 0 \\
e_{802}: & x_{1603} + x_{1604} - C(x_1 x_{802} + x_2 x_{803} - 9(x_1 x_{1603} + x_2 x_{1604}) - x_{1603} - x_{1604}) = 0 \\
 & \ldots \\
e_{1601}: & x_{2402} + x_{2403} - C(x_{800} x_{1601} + x_{802} x_{1602} - 9(x_{800} x_{2402} + x_2 x_{2403}) - x_{2402} - x_{2403}) = 0
\end{array}
$$

a) The constraints of CATMIX800 after parsing and reordering

$$
\begin{array}{ll}
E_1: & Y[i] + Y[i+1] + C(X[i]Y[i] + X[i+1]Y[i+1] - 10(X[i]Z[i] + X[i+1]Z[i+1])) = 0 \\
E_2: & Z[i] + Z[i+1] - C(X[i]Y[i] + X[i+1]Y[i+1] - 9(X[i]Z[i] + X[i+1]Z[i+1]) - Z[i] - Z[i+1]) = 0
\end{array}
$$

b) The constraints of CATMIX800 with recovered indexes, $0 \leq i \leq 800$

Figure 5. CATMIX800: an example non-indexed NLP benchmark.

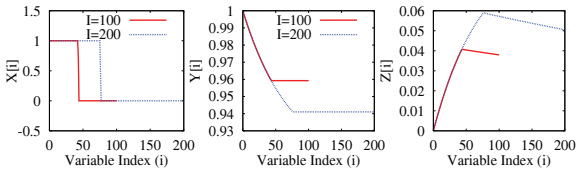


Figure 6. The generalizable solutions of two simplified versions of CATMIX800

straints, and p_N^* (or p^*) be the global minimum of P_N, the original problem with N constraints. Obviously, $p_i^* \leq p_j^*$ for $i \leq j$, where the constraints of P_i is a subset of those of P_j. Therefore, the optimal value of each simplified problem is a lower bound for p_N^*. That is,

$$p_0^* \leq p_1^* \leq \ldots \leq p_i^* \leq \ldots \leq p_j^* \leq \ldots \leq p_N^*. \quad (4)$$

Let p_i be a local minimum solution of P_i found by some solver, where $p_i \geq p_i^*$. Unlike (4), there is no similar relation between p_i and p_j, where the constraints of P_i is a subset of those of P_j, because p_i may be greater than or smaller than p_j. Figure 7a illustrates this fact that the solutions of the sampled problems found by Ipopt [5] do not increase monotonically as the number of constraints grows.

b) Smoothness and convergence of the starting points. In generating starting points for regularly indexed problems, we assume that the values of variables of the same index are highly related. Those variables have a *smoothness* property when the mean squared error (MSE) between their optimal and the linearly interpolated values decreases monotonically as the number of sampled variables increases beyond some threshold. As is discussed in Section 1, this smoothness property has been found in many AMPL benchmarks, where each problem has constraints that are grouped and the parameters in each group are closely related. With this property, our proposed techniques in Section 2 generally work well because they are based on linear interpolations.

Figure 7b illustrates a) the values of $x[i]$, $2001 \leq i \leq 3000$, found by Ipopt (Ipopt Sol: Orig. Prob.), b) the linearly interpolated values of 50 sampled final solutions (Ipopt Sol: Sampled), and c) the variable values found by solving a problem with 50 sampled constraints (Sampled Prob: 50 Cons). Figure 7c shows that the MSE between (a) and (b) for $1 \leq i \leq 5000$ decreases monotonically when the number of samples exceeds 250 and converges to nearly zero when the number of samples is 500. In contrast, the MSE between (a) and (c) oscillates around 0.1, and the MSE between (a) and the default starting point is 8.0664.

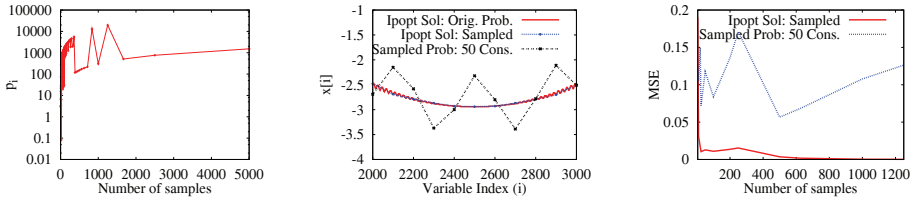
4 Experimental Results and Conclusions

In this section, we test the effectiveness of the starting points generated by our proposed methods on three leading NLP solvers (SNOPT [2], Ipopt [5] and Knitro [1]) and two leading MINLP solvers (FilMINT [3] and MINLP_BB [4]).

The experiments were conducted on the benchmark suites shown in Table 1. As is mentioned in Section 1, we study 242 (190 NLP and 52 MINLP) problems, after eliminating relatively easy problems (unconstrained, linear, quadratic, and small problems with less than 50 variables or constraints). Among the problems studied, 99 NLP problems can be solved by SNOPT (an NLP solver using SQP), and 10 MINLP problems can be solved by FilMINT (a MINLP solver using branch-and-cut and filterSQP).

Tables 2 and 3 summarize the results on the the remaining 133 (91 NLP and 42 MINLP) problems that cannot be solved by SNOPT or FilMINT. Note that there are 45 more problems in the MINLP library that are not shown in Table 3. These problems were discarded because they have similar characteristics. For example, we have only tested fo9 because fo9, fo9_ar2_1, fo9_ar25_1, fo9_ar3_1 and fo9_ar5_1 are all similar block layout design problems of various aspect ratios. We use the number of iterations as a metric for evaluating solution time, since CPU times vary across different computers (except for FilMINT that only reports CPU times). Also, all the results should be compared within a solver and not across solvers.

For the 91 NLP benchmark in Table 2, SNOPT (*resp.*, Ipopt and Knitro) can solve 31 (*resp.*, 17 and 30) more



a) Objective value p_i b) Variable $x[i]$ c) MSE of starting points

Figure 7. The convergence of the objective values and the MSE of sampled solutions.

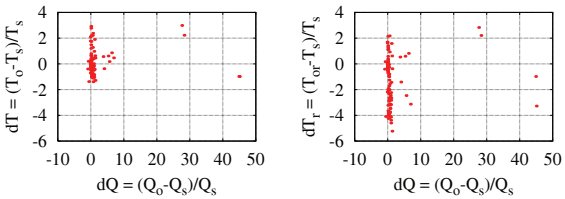
problems using the new starting points. For the other 74 ($resp.$, 61) problems that can be solved from their default starting points, Ipopt ($resp.$, Knitro) can solve them with 35.1% ($resp.$, 14.7%) less iterations from the new starting points. For each solver, we have also highlighted those results that are better when using our proposed starting points.

For the 42 MINLP benchmarks in Table 3, FilMINT ($resp.$, MINLP_BB) can solve 4 ($resp.$, 10) more problems using the new starting points. For the 12 that can be solved from both their default and the proposed starting points, 4.9% less iterations are needed when using the new points.

There are 31 (16 NLP and 15 MINLP) problems that cannot be solved by any solvers: lukvle2 and lukvli2 in the NLP suite are likely to be infeasible, since no known solutions exist for them; bratu2d, bratu2d_l, bratu2dt, bratu2dt_l, bratu3d, cbratu2d, cbratu2d_l, cbratu3d, porous1, porous1_l, porous2 and porous2_l from Nonsys and cont4_400 and twod from the NLP suite are too large, and the processes were killed due to a lack of memory. However, the starting point generated for the 12 problems in Nonsys that cannot be solved by any of the three solvers from their default starting points were close enough to satisfy all the constraints, and we consider them solved in zero iteration by our approach in Table 2.

Figure 8 compares the normalized time and quality of the 109 problems that can be solved by our approach as well as by SNOPT or FilMINT. These problems are smaller problems that require very small amount of execution times. In general, due to the additional overheads for finding good starting points, our approach is slower in solving these problems (Figure 8a). However, when only the times for resolution are considered, our approach is much faster in solving most of them (Figure 8b). Note that the normalized solution quality (dQ) of four of the problems found by our approach is much worse than that of SNOPT or FilMINT. This is due to the fact that those solutions found by SNOPT or FilMINT (Q_s) are very close to zero.

In conclusion, by using the better starting points generated by our proposed methods in this paper, most of the tested problems can be solved in significantly less iterations. This is due to the fact that the starting points found by our approach are much closer to the feasible space. In our experimental evaluations of 190 NLP ($resp.$, 52 MINLP) benchmarks, our approach can solve 97.9% ($resp.$, 71.2%) of the problems with significantly less iterations, as compared to 85.3% ($resp.$, 46.2%) of the problems solvable by the best existing solvers from their default starting points.



a) Total time vs quality b) Resolution time only vs quality

Figure 8. Normalized time and quality of the 109 problems that can be solved by our approach as well as by SNOPT or FilMINT. (Q_0: our quality; Q_s: SNOPT quality; T_0: our time; T_s: SNOPT time.) Our approach performs better than SNOPT when dQ and dT are negative.

In the future, we plan to automate the process of generating good starting points and integrate it as a preprocessor to existing NLP and MINLP solvers. Also, our current results are based on starting points generated by SNOPT (the only available solver on our local computer), and we plan to employ other solvers besides SNOPT in our approach for generating starting points.

References

[1] R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: an integrated package for nonlinear optimization. In G. D. Pillo and M. Roma, editors, *Large-scale nonlinear programming*, pages 35–59. Springer-Verlag, 2006.

[2] P. E. Gill, W. Murray, and M. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM J. on Optimization*, 12:979–1006, 2002.

[3] S. L. K. Abhishek and J. T. Linderoth. FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs. Technical Report ANL/MCS-P1374-0906, Mathematics and Comp. Sci. Division, Agronne National Lab., 2008.

[4] S. Leyffer. Mixed integer nonlinear programming solver. `http://www-unix.mcs.anl.gov/~leyffer/solvers.html`, 2002.

[5] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Programming*, 106(1):25–57, 2006.

Table 2. Results on solving NLP benchmarks from the CUTEr, NLP, and Nonsys suites using SNOPT [2], Ipopt [5], and Knitro [1] with default and our proposed starting points. Results on Ipopt and Knitro were obtained by submitting jobs to the NEOS server (`http://www-neos.mcs.anl.gov/`); results of the other solver were collected on an AMD 2-GHz computer with RedHat Linux 3.4.6 and 2 GB memory and a time limit of 3,600 sec. All timing results refer to the number of iterations used by each solver and should only be compared within a solver. For each instance, n_c (*) and n_v are, respectively, the number of (*nonlinear) constraints and variables; and n_s is the number of sampled constraints (for technique A), or the number of variables in the simplified problem (for B), or nil (for C where MINLP is relaxed to NLP). "Tech." refers to the method in Section 2 for generating starting points; "‡" refers to maximization instead of minimization problems; "f" means that no feasible solutions were found.; "t" means that the process was killed due to time limit.; and "m" means that the process ran out of memory. For each solver, we have shaded those results (time and quality) that are better when our proposed starting point as compared to that of using the default starting point.

Benchmark	n_c (*)	n_v	n_s	Tech.	Default		Proposed		Default		Proposed		Default		Proposed	
					Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol
CUTEr (Not Solvable by SNOPT in 3,600 sec.)					SNOPT				Ipopt				Knitro			
dtoc1l	9990(0)	14985	1495	B1	f	f	f	f	6	125.34	6	125.34	9	125.34	5	125.34
dtoc2	3996(3996)	5994	598	B1	f	f	f	f	10	0.51	8	0.51	260	0.53	48	0.51
dtoc4	9997(4997)	14996	749	B1	f	f	f	f	3	2.87	2	2.87	3	2.87	1	2.87
dtoc6	5000(5000)	10000	1001	B1	f	f	f	f	11	134851.00	6	134851.00	12	134850.62	6	134850.62
hager2	5000(0)	10000	201	B1	f	f	f	f	1	0.43	1	0.43	2	0.43	2	0.43
hager4	5000(0)	10000	1001	B1	f	f	f	f	10	2.79	8	2.79	7	2.79	7	2.79
orthrdm2	2000(2000)	4003	10	A	f	f	f	f	6	155.53	5	155.53	5	155.53	5	155.53
orthregd	5000(5000)	10003	50	A	f	f	f	f	6	1523.90	8	1523.90	6	1523.90	8	1523.90
orthrgdm	5000(5000)	10003	50	A	f	f	f	f	6	1513.80	7	1513.80	6	1513.80	6	1513.80
orthrgds	5000(5000)	10003	20	A	f	f	f	f	16	1523.90	8	1523.90	47	1776.23	7	1523.90
svanberg	5000(5000)	5000	500	B1	f	f	f	f	30	8361.42	18	8361.42	15	8361.42	11	8361.42
catenary	166(166)	496	54	B1	f	f	f	f	56	-348403.00	227	-406699	2928	-348403.16	1822	-406699
dtoc5	4999(4998)	9998	999	B1	f	f	f	f	4	1.54	2	1.54	3	1.54	1	1.54
hadamard	256(128)	65	17	B1	f	f	1	1.00	8	1.00	7	1.00	3	1.00	7	1.00
kissing	903(903)	127	67	B1	f	f	f	f	350	0.85	350	0.85	62	0.85	62	0.85
orthregc	5000(5000)	10005	50	A	f	f	f	f	14	189.60	12	192.71	11	189.60	9	192.72
ubh5	14000(2000)	19997	2010	B1	f	f	f	f	5	1.12	4	1.12	t	t	2	1.12
NLP (Not Solvable by SNOPT in 3,600 sec.)					SNOPT				Ipopt				Knitro			
cont5	3717(236)	3953	517	B1	f	f	1748	0.00	21	0.55	47	0.55	12	0.55	15	0.55
cont6	3717(236)	3953	517	B1	f	f	2727	0.01	21	0.01	39	0.01	14	0.01	19	0.01
cont7	2597(2401)	2793	517	B1	f	f	8005	3.00	f	f	f	f	10	0.26	14	0.26
cont8	2597(2401)	2793	517	B1	f	f	6778	3.00	f	f	f	f	12	0.16	15	0.16
ex4	3717(3481)	7198	801	B1	f	f	893	0.00	13	0.08	17	0.08	6	0.08	7	0.08
ex6	3717(3481)	7198	801	B1	f	f	f	f	18	-4.22	23	-4.22	11	-4.22	19	-4.22
lukvle1	49998(49998)	50000	500	B2	f	f	f	f	6	6.23	1	6.23	6	6.23	0	6.23
lukvle2	49993(49993)	50002	502	B2	f	f	f	f	m	m	f	f	m	m	m	m
lukvle3	2(2)	50002	502	B2	f	f	0	65.00	9	65.12	1	65.12	t	t	0	65.12
lukvle4	49998(49998)	50002	502	B2	f	f	f	f	16	242908.00	1	242908.00	20	243157.77	0	242907.68
lukvle5	49996(49996)	50002	502	B2	f	f	f	f	15	2.64	2	0.37	19	2.64	1	0.37
lukvle6	24999(24999)	49999	500	B2	f	f	f	f	f	f	f	f	14	3144226.08	1	3144226.08
lukvle7	4(4)	50002	502	B2	f	f	f	f	17	-66139.60	2	-66139.60	20	-66139.62	2	-66139.62
lukvle8	49998(49998)	50000	1000	B2	f	f	f	f	13	4130070	13	4130070	t	t	2	4130073.56
lukvle9	6(6)	50000	50	B2	f	f	1	4994.00	21	4994.67	1	4994.67	8352	4994.67	0	4994.67
lukvle10	49998(49998)	50000	500	B2	f	f	f	f	13	17677.20	1	17677.20	94	17677.25	1	17677.24
lukvle11	33330(33330)	49997	497	B2	f	f	0	0.00	8	0.00	0	0.00	t	t	0	0.00
lukvle12	37497(37497)	49997	997	B2	f	f	f	f	7	77203.90	2	77203.90	6	77203.88	1	77203.88
lukvle13	33330(33330)	49997	497	B2	f	f	f	f	23	401793.00	5	401784.00	t	t	1	401784.15
lukvle14	33330(33330)	49997	497	B2	f	f	f	f	21	380425.00	3	380425.00	17	380424.85	2	380424.85
lukvle15	37497(37497)	49997	997	B2	f	f	0	0.00	53	0.00	0	0.00	t	t	0	0.00
lukvle16	37497(37497)	49997	997	B2	f	f	0	0.00	7	0.00	0	0.00	28	0.00	0	0.00
lukvle17	37497(37497)	49997	997	B2	f	f	0	71433.00	8	71433.10	1	71433.10	8	71433.15	0	71433.15
lukvle18	37497(37497)	49997	997	B2	f	f	0	59982.00	13	59982.00	1	59982.00	12	59982.01	0	59982.01
lukvli1	49998(49998)	50000	50	B2	f	f	f	f	3000	472.98	12	0.00	t	t	0	0.00
lukvli2	49993(49993)	50000	502	B2	f	f	f	f	f	f	m	m	m	m	m	m
lukvli3	2(2)	50000	500	B2	f	f	0	11.00	10	11.58	5	11.58	9	11.58	2	11.58
lukvli4	49998(49998)	50000	500	B2	f	f	f	f	26	20102.10	6	20102.10	t	t	3	20102.05
lukvli5	49996(49996)	50000	500	B2	f	f	f	f	f	f	f	f	t	t	9	0.53
lukvli6	24999(24999)	49999	500	B2	f	f	f	f	f	f	f	f	16	3144227.58	2	3144225.96
lukvli7	4(4)	50000	500	B2	f	f	f	f	22	-18633.90	7	-18633.90	69	-18633.85	2	-18633.84
lukvli8	49998(49998)	50000	500	B2	f	f	f	f	183	5023590.00	93	4129430.00	50	6161014.81	42	4129611.13
lukvli9	6(6)	50000	50	B2	f	f	f	f	39	4994.67	13	4994.67	t	t	70	4994.67
lukvli10	49998(49998)	50000	500	B2	f	f	f	f	92	17677.20	4	17677.20	44	17677.24	4	17677.24
lukvli11	33330(33330)	49997	497	B2	f	f	f	f	41	0.00	17	0.00	17	0.00	18	0.00
lukvli12	37497(37497)	49997	997	B2	f	f	0	0.00	28	0.00	23	0.00	t	t	0	0.00
lukvli13	33330(33330)	49997	497	B2	f	f	47	2.00	23	0.00	22	0.00	t	t	6	0.00
lukvli14	33330(33330)	49997	497	B2	f	f	f	f	23	380425.00	22	72632.00	16	380425.00	16	72631.98
lukvli15	37497(37497)	49997	997	B2	f	f	45	1.00	f	f	21	0.00	24	6.34	15	0.00
lukvli16	37497(37497)	49997	997	B2	f	f	0	3.00	24	0.00	32	0.00	16	0.00	0	0.00
lukvli17	37497(37497)	49997	997	B2	f	f	8	0.00	75	1.55	21	0.39	55	8.94	24	0.39
lukvli18	37497(37497)	49997	997	B2	f	f	0	5.00	19	0.00	12	0.00	13	0.00	13	0.01
cont5_1	40200(200)	40400	2651	B1	f	f	f	f	14	2.72	15	2.72	t	t	8	2.72
cont5_2_1	40200(0)	40400	2651	B1	f	f	f	f	37	0.00	34	0.00	18	0.00	20	0.00
cont5_2_2	40200(200)	40400	2651	B1	f	f	f	f	50	0.00	41	0.00	t	t	21	0.00
cont5_2_3	40200(200)	40400	2651	B1	f	f	f	f	50	0.00	45	0.00	30	0.00	24	0.00
cont5_2_4	40200(40000)	40400	2651	B1	f	f	f	f	14	0.07	9	0.00	29	0.07	11	0.07
cont5_400	160400(0)	160800	2651	B1	f	f	f	f	m	m	f	f	m	m	m	m
twod	129850(0)	132304	9641	B1	f	f	f	f	m	m	f	f	m	m	m	m

Table 2. (continued).

Benchmark	n_c (*)	n_v	n_s	Tech.	Default		Proposed		Default		Proposed		Default		Proposed	
					Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol	Time	Sol
NLP (Not Solvable by SNOPT in 3,600 sec.)					SNOPT				Ipopt				Knitro			
catmix800	1600(1600)	2401	303	B1	f	f	f	f	15	-0.05	15	-0.05	8	-0.05	9	-0.05
ex8_2_1	31(25)	55	15	B1	f	f	f	f	t	t	f	f	140	-73873259870.00	26	-3673965181.00
ex8_2_2	1943(1875)	7510	1512	B1	f	f	f	f	f	f	56	-552.67	30	-552.67	34	-552.67
ex8_2_3	3155(3125)	15636	3136	B1	f	f	5540	-3730.83	71	-3731.08	64	-3731.08	28	-3731.08	29	-3731.08
ex8_2_4	81(75)	55	35	B1	f	f	f	f	f	f	f	f	f	f	10000	-4027.6
glider50	605(551)	654	121	B1	f	f	f	f	f	f	f	f	723	-249.95	19	-249.99
glider100‡	1205(1101)	1304	231	B1	f	f	f	f	f	f	f	f	2989	250.00	39	249.99
glider200‡	2405(2201)	2604	561	B1	f	f	f	f	f	f	f	f	605	249.97	2251	249.99
glider400‡	4805(4401)	5204	561	B1	f	f	f	f	f	f	f	f	111	250.00	1081	274.61
Nonsys (Not Solvable by SNOPT in 3,600 sec.)					SNOPT				Ipopt				Knitro			
bratu2d	61504(61504)	62500	10000	B1	f	f	0	0.00	m	m	0	0.00	m	m	0	0.00
bratu2d_l	248004(248004)	250000	1000	B1	f	f	0	0.00	m	m	0	0.00	m	m	0	0.00
bratu2dt	61504(61504)	62500	10000	B1	f	f	0	0.00	m	m	0	0.00	m	m	0	0.00
bratu2dt_l	248004(248004)	250000	10000	B1	f	f	0	0.00	f	f	0	0.00	t	t	0	0.00
bratu3d	27000(27000)	32768	512	B1	f	f	0	0.00	m	m	0	0.00	m	m	0	0.00
cbratu2d	123008(123008)	62500	10000	B1	f	f	0	0.00	f	f	0	0.00	m	m	0	0.00
cbratu2d_l	316808(316808)	160000	10000	B1	f	f	0	0.00	m	m	0	0.00	f	f	0	0.00
cbratu3d	11664(11664)	16000	125	B1	f	f	0	0.00	m	m	0	0.00	m	m	0	0.00
chemrctb	50000(49998)	50000	500	B1	f	f	0	0.00	3	0.00	2	0.00	3	0.00	2	0.00
porous1	61504(61504)	65000	10000	B1	f	f	0	0.00	f	f	0	0.00	m	m	0	0.00
porous1_l	248004(248004)	250000	19603	B1	f	f	0	0.00	m	m	0	0.00	m	m	0	0.00
porous2	61504(61504)	65000	10000	B1	f	f	0	0.00	f	f	0	0.00	m	m	0	0.00
porous2_l	248004(248004)	250000	19603	B1	f	f	0	0.00	m	m	0	0.00	m	m	0	0.00
semicon1	50000(50000)	50002	500	B1	f	f	0	0.00	51	0.00	82	0.00	36	0.00	1	0.00
semicon1_l	100000(100000)	100002	1000	B1	f	f	290	0.00	50	0.00	81	0.00	m	m	0	0.00
semicon2	50000(50000)	50002	500	B1	f	f	0	0.00	27	0.00	1	0.00	t	t	1	0.00

Table 3. Result on solving MINLP benchmarks from the MacMINLP and MINLP Library suites using FilMINT [3] and MINLP_BB [4] with default and our proposed starting points. The results on MINLP_BB were obtained by submitting jobs to NEOS (`http://www-neos.mcs.anl.gov/`), and those of FilMINT were collected on an AMD 2-GHz computer with RedHat Linux 3.4.6 and 2 GB memory and a time limit of 3,600 sec. The timing results for FilMINT and MINLP_BB are, respectively, in sec. and the number of objective-function evaluations and should only be compared within a solver. The other notations are the same as those in Table 2.

Benchmark	n_c (*)	n_v(#)	n_s	Tech.	Default		Proposed		Default		Proposed	
					Time	Sol	Time	Sol	Time	Sol	Time	Sol
MacMINLP (Not Solvable by FilMINT in 3,600 sec.)					FilMINT				MINLP_BB			
c-reload-14a‡	308(258)	342(168)	0	C	f	f	f	f	1203	1.01	8	1.01
c-reload-14e‡	308(258)	342(168)	0	C	f	f	f	f	100	1.03	20	1.02
c-reload-q-24‡	632(584)	968(576)	0	C	f	f	f	f	122	0.00	5	0.00
c-reload-q-25‡	658(608)	1033(625)	0	C	f	f	f	f	1155	1.12	2159	1.12
c-reload-q-49‡	1430(1332)	3292(2401)	0	C	f	f	f	f	f	f	f	f
c-reload-q-104‡	3338(3130)	12906(10816)	0	C	f	f	f	f	f	f	f	f
c-sched2‡	137(0)	400(308)	0	C	f	f	f	f	95295	146205.96	t	t
feedloc	247(147)	89(37)	0	C	f	f	f	f	30	0.00	31	0.00
space-25	235(25)	893(750)	0	C	f	f	f	f	t	t	f	f
space-25-r	160(25)	818(750)	0	C	f	f	f	f	f	f	f	f
space-960	8417(960)	15137(9600)	0	C	f	f	f	f	f	f	f	f
space-960-ir	3617(960)	2657(960)	0	C	f	f	1548.17	48000000.00	f	f	f	f
stockcycle	97(0)	480(480)	0	C	f	f	f	f	f	f	t	t
trimlon6	72(12)	168(168)	0	C	f	f	113.97	17.10	429520	15.30	f	f
trimlon12	72(12)	168(168)	0	C	f	f	f	f	f	f	f	f
trimloss5	90(5)	161(131)	0	C	f	f	f	f	t	t	t	t
trimloss6	154(7)	345(289)	0	C	f	f	f	f	f	f	837285	19.20
trimloss7	154(7)	345(289)	0	C	f	f	f	f	t	t	f	f
trimloss12	372(12)	800(644)	0	C	f	f	f	f	t	t	t	t
MINLPLib (Not indexed, not Solvable by FilMINT in 3,600 sec.)					FilMINT				MINLP_BB			
4stufen	94(33)	145(48)	0	C	f	f	f	f	f	f	4853	116329.67
beuster	109(46)	153(51)	0	C	f	f	f	f	f	f	307807	116329.67
deb9	566(426)	735(10)	0	C	f	f	f	f	1292	116.58	1360	116.58
enpro56	192(2)	128(73)	0	C	f	f	f	f	f	f	59869	263428.30
ex1266a	53(6)	48(48)	0	C	f	f	f	f	f	f	f	f
feedtray2	197(147)	64(12)	0	C	f	f	f	f	7	0.00	15	0.00
feedtray	91(62)	98(7)	0	C	f	f	f	f	10	0.00	8	0.00
fo7	211(14)	112(42)	0	C	f	f	f	f	t	t	678258	35.16
fo8	273(16)	144(56)	0	C	f	f	f	f	t	t	666549	36.90
fo9	343(18)	180(72)	0	C	f	f	f	f	f	f	2385472	58.42
gasnet	67(42)	86(10)	0	C	f	f	f	f	f	f	1040	0.00
gastrans	125(24)	89(15)	0	C	f	f	f	f	164	89.09	132	89.09
lop97ic	87(39)	1626(1531)	0	C	f	f	f	f	t	t	t	t
m7	211(14)	112(42)	0	C	f	f	935.58	106.76	t	t	t	t
nuclear14a	633(584)	992(600)	0	C	f	f	f	f	51	-1.13	154	-1.13
o7	211(14)	112(42)	0	C	f	f	f	f	f	f	t	t
oil	1417(412)	1361(19)	0	C	f	f	f	f	f	f	f	f
product	1031(132)	948(92)	0	C	f	f	f	f	t	t	87891	-2139.51
super1	1580(372)	1264(31)	0	C	f	f	21.18	10.23	f	f	f	f
tloss	53(6)	48(48)	0	C	f	f	f	f	1429	16.30	1455	16.30
tls5	90(5)	161(136)	0	C	f	f	f	f	t	t	1318698	12.30
waste	1882(1230)	1425(400)	0	C	f	f	f	f	t	t	f	f
waterx	54(16)	70(14)	0	C	f	f	f	f	1091	983.02	1389	945.19