Finding Good Starting Points for Solving Nonlinear Constrained Optimization Problems by Parallel Decomposition

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) formulated as *nonlinear programming* (NLP) and *mixed-integer NLP* (MINLP). By exploiting the localities of constraints, we first partition each problem by parallel decomposition into subproblems that are related by complicating constraints and complicating variables. We develop heuristics for finding good starting points that are critical for resolving the complicating constraints and variables. In our experimental evaluations of 255 benchmarks, our approach can solve 89.4% of the problems, whereas the best existing solvers can only solve 42.8%.

1 Introduction

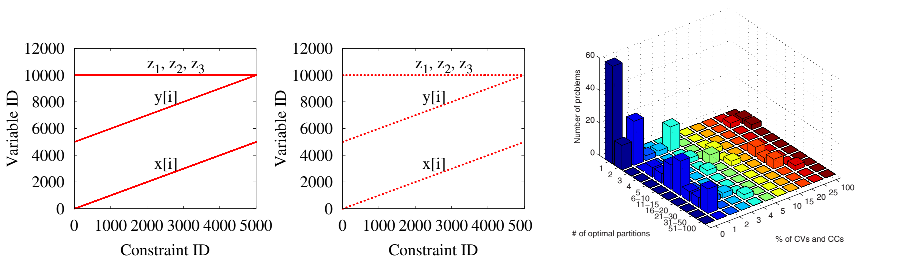
The NLP and MINLP problems studied in this paper have nonlinear and non-convex objective and constraint functions that are formulated as follows:

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

where $\mathbf{z} = (\mathbf{x}, \mathbf{y})$, and $\mathbf{x} \in \mathbb{R}^v$ and $\mathbf{y} \in \mathbb{D}^v$ 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 minimum* (CLM) solutions.

Consider ORTHRGDS, an NLP from the CUTE 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 $i = 1, \ldots, N$:

$$\begin{aligned} \min_{\mathbf{z}} \; f(\mathbf{z}) &= \sum_{i=1}^{N} \left((x[i] - xd[i])^2 + (y[i] - yd[i])^2 \right) \qquad (2) \\ \text{subject to } h_i(\mathbf{z}) &= \; 0 \text{ for } 1 \leq i \leq N, \\ \text{where } h_i(\mathbf{z}) &= \left((x[i]-z_1)^2 + (y[i]-z_2)^2 \right)^2 - \left((x[i]-z_1)^2 + (y[i]-z_2)^2 \right)(1+z_3^2)^2, \\ xd[i] &= (C_1 + \cos(2\pi(i-1)/N)) \cos(2\pi(i-1)/N)(1 + C_2 \cos(2\pi C_3 (i-1)/N)), \\ yd[i] &= (C_1 + \cos(2\pi(i-1)/N)) \sin(2\pi(i-1)/N)(1 + C_2 \cos(2\pi C_3 (i-1)/N)). \end{aligned}$$



a) NLP ORTHRGDS b) 50 sampled constraints c) Localities in 255 benchmarks

Fig. 1. The regular constraint structure of ORTHRGDS and NLP benchmarks studied

Here, $C_1 = 1+1.7^2$, $C_2 = 0.2$, and $C_3 = 237.1531$. The problem cannot be solved by SNOPT [1] and Lancelot, but can be solved by KNITRO and IPOPT [2].

ORTHRGDS belongs a large class of benchmark problems specified by an algebraic modeling language, like AMPL and GAMS, that uses indexes to define groups of variables and constraints with some common properties. Indexing is necessary when specifying large-scale problems because it will be very cumbersome to give a unique name for every variable and constraint. The use of indexed constraints leads to a large number of similar and related constraints.

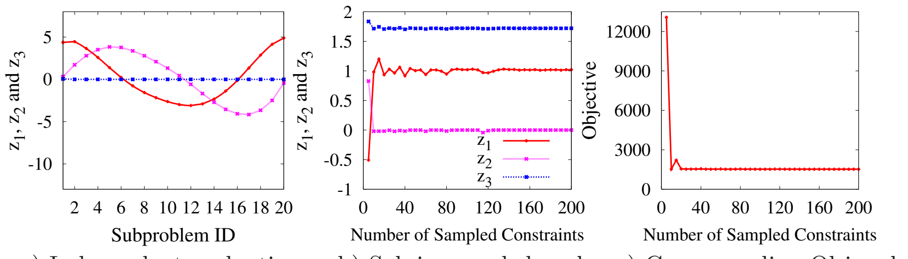
Figure 1a illustrates the regular structure of the constraints in ORTHRGDS. Using the indexed form, it shows a dot where a constraint on the x-axis is related to a variable on the y-axis. In particular, the i^{th} constraint $h_i(\mathbf{z}) = 0$ involves variables $x[i]$ and $y[i]$ and common variables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

Parallel decomposition can be used to exploit the regular constraint structures in large-scale COPs and to decompose them into loosely coupled and much less complex subproblems. The original idea came from solving large-scale linear programming (LP) problems in the 1960s [3]. It partitions the state space of a problem in such a way that the combined state space is the cross product of the subproblem state spaces. The approach leads to the partitioning of variables into possibly overlapping subsets. Those variables that are shared among the subsets are called *complicating variables* (CVs), and those in one subset and not shared are called *local variables* (LVs). As a result of the partitioning of the variables, constraints are also partitioned. Those constraints involving variables in more than one subset are called *complicating constraints* (CCs), whereas those involving local variables in one subset are called *local constraints* (LCs).

A COP has good *localities* when its decomposed subproblems are loosely coupled by a small number of CVs and CCs. For instance, when ORTHRGDS is decomposed by parallel decomposition along the constraint dimension into 20 subproblems, its variables are partitioned into 20 subsets, where the j^{th} subset, $j = 1, \ldots, 20$, has variables $\{x[i], y[i], z_1, z_2, z_3\}$ for $250(j-1)+1 \leq i \leq 250j$. Hence, z_1, z_2, and z_3 are CVs; and all constraints are LCs. If the CVs and CCs can be effectively resolved, then parallel decomposition will be a good method for solving the COP because the complexity of each subproblem is much lower.

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

CUTE	Nonlinear optimization	`http://www.princeton.edu/~rvdb/bench.html`
NLP	Nonlinear programming	`http://plato.asu.edu/ftp/ampl-nlp.html`
NonSys	Nonlinear systems of equations	`http://plato.asu.edu/ftp/ampl_files/nonsys_ampl/`
MacMINLP	AMPL collection of MINLP	`http://www-unix.mcs.anl.gov/~leyffer/macminlp/`
COPS	Large-scale optimization problems	`http://www-unix.mcs.anl.gov/~more/cops/`



a) Independent evaluations b) Sol. in sampled prob. c) Corresponding Obj. val.

Fig. 2. Convergence behavior when ORTHRGDS is partitioned into 20 subproblems: a) the values of z_1, z_2, and z_3 do not converge when the subproblems are solved iteratively, using the values found in the last problem as the starting point in the next; b) the values of z_1, z_2, z_3 found from the sampled problem; c) the corresponding value of the objective function when fixing z_1, z_2, and z_3 found from the sampled problem

To show that many existing benchmarks have good localities that can be exploited by parallel decomposition, we have exhaustively tested the five benchmark suites in Table 1. After eliminating those unconstrained, linear, or quadratic problems, and small problems with less than 50 variables or constraints, we have 255 benchmarks whose localities are summarized in Figure 1c. The x-axis shows the optimal number of partitions found by the method in Section 2, and the y-axis shows the ratio between the total number of CVs and CCs and the total number of constraints, based on using the optimal number of partitions. Most of the problems have good localities with less than 1% of CVs and CCs. In some problems, the optimal number of partitions is one and no partitioning is needed.

One of the requirements when a COP is partitioned by parallel decomposition is that its CVs and CCs must converge when a solution is found. When the CVs are formulated as CCs, it results in a significant increase in the number of equality constraints that are hard to satisfy. On the other hand, when the subproblems are solved independently and the CVs are propagated from one subproblem to another during the solution process, they will not converge without a proper technique that ensures their consistency across the subproblems.

For example, Figure 2a illustrates that CVs z_i, $i = 1, 2, 3$, in ORTHRGDS will not converge when the z_i found in one subproblem are simply passed to the next as its initial values. The importance of using good starting points is also illustrated in solving EIGENA2 from CUTE. SNOPT can solve EIGENA2 when $q[i, j]$ for $1 \le i, j \le 10$, is initialized to an identity matrix (the default), but fails to find a feasible solution when $q[i, j]$ is initialized to a zero matrix.

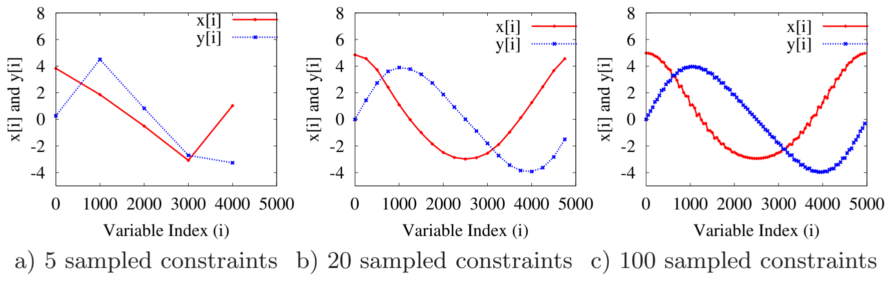


Fig. 3. The LVs converge with an increasing number of sampled constraints

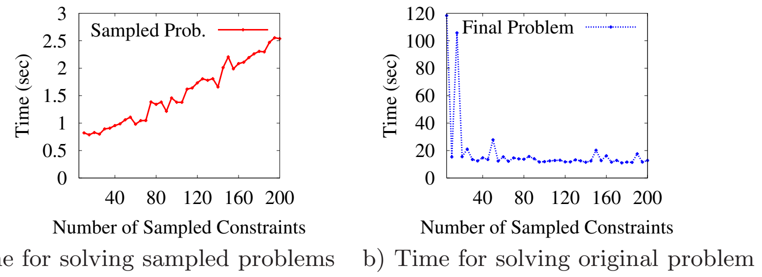


Fig. 4. Trade-offs between the time for solving a problem with sampled constraints and the time for solving the original problem using the starting point found

The convergence, however, can be significantly improved when the CVs and CCs are initialized properly, and simple techniques are employed to enforce their consistency. In this paper, we show that, for COPs with good structural localities, it is possible to find good starting points by exploiting their structures.

As an illustration, consider a simplified version of ORTHRGDS obtained by uniformly sampling the constraints in Figure 1a. (See Figure 1b when 50 constraints are sampled.) Figure 2b depicts the values of z_1, z_2, and z_3 when using SNOPT to solve ORTHRGDS of different numbers of sampled constraints. Although there are small fluctuations in their values, the CVs converge to some stable values when 20 or more constraints are sampled. This convergence behavior is also extended to LVs. Figure 3 shows the values of LVs $x[i]$ and $y[i]$ for three simplified problems. 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 usually 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 sampled constraints and the other variables interpolated as is shown in Figure 3, we can solve ORTHRGDS by parallel decomposition when we fix the value of its CVs throughout the process. Figure 2c shows that the objective value improves and converges when the number of sampled constraints increases.

In general, our approach of using fixed CVs may not lead to a feasible solution. Section 3 presents a technique that first relaxes the CVs to within some bounds and that gradually tightens the bounds until a feasible solution is found.

There is a trade-off between the time for solving a problem with sampled constraints (which increases as the number of constraints increases−Figure 4a), 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 4b). However, the time for solving a sampled problem is usually much smaller. Hence, we can simply choose the number of sampled constraints based on the convergence of the variables found. For example, Figure 2b shows that the solutions of ORTHRGDS converge when 20 or more constraints are sampled.

Our work is based on the theory of extended saddle points and the parallel decomposition algorithm summarized in Section 2 [4]. Our main contribution in this paper is on the development of algorithms for finding good starting points when solving NLPs and MINLPs by parallel decomposition. Without a good starting point, the previous parallel decomposition method gets stuck easily and does not converge.

2 CPOpt for Solving Constraint-Partitioned COPs

We summarize in this section a necessary and sufficient condition on the constrained local minima of P_m [5]. We also describe CPOpt, an implementation for resolving the CCs and CVs when P_m is partitioned by its constraints.

Definition 1. *Let* $a^+ = \max(a, 0)$. *The* penalty function *of* P_m *with penalty vectors* $\boldsymbol{\alpha} \in \mathbb{R}^m$ *and* $\boldsymbol{\beta} \in \mathbb{R}^r$ *is:*

$$L_m\big(\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}\big) = f(\mathbf{z}) + \boldsymbol{\alpha}^T |h(\mathbf{z})| + \boldsymbol{\beta}^T g(\mathbf{z})^+ = f(\mathbf{z}) + \sum_{i=1}^{m} \alpha_i |h_i(\mathbf{z})| + \sum_{j=1}^{r} \beta_j g_j(\mathbf{z})^+.$$

Without showing the details [5], we have proved that each *constrained local minimum* (CLM) of P_m must satisfy an *extended saddle point condition* (ESPC) and vice versa. Here, an *extended saddle point* (ESP) $(\mathbf{z}^*, \boldsymbol{\alpha}^{**}, \boldsymbol{\beta}^{**})$ is a a local minimum of $L_m\big(\mathbf{z}, \boldsymbol{\alpha}^{**}, \boldsymbol{\beta}^{**}\big)$ with respect to $\mathbf{z}$ and a local maximum of $L_m\big(\mathbf{z}^*, \boldsymbol{\alpha}, \boldsymbol{\beta}\big)$ with respect to $\boldsymbol{\alpha}^{**}$ and $\boldsymbol{\beta}^{**}$, when $\boldsymbol{\alpha}^{**}$ and $\boldsymbol{\beta}^{**}$ are non-negative and larger than some thresholds.

ESPC can be applied to solve P_t, a version of P_m whose constraints can be partitioned by parallel decomposition into N subsets:

$$\begin{aligned} (P_t): \quad & \min_{\mathbf{z}(i)} \; f(\mathbf{z}) \\ \text{subject to } \; & h^{(i)}(\mathbf{z}(i)) = 0, \quad g^{(i)}(\mathbf{z}(i)) \le 0 \quad (\text{LC } i,\; i = 1, \ldots, N) \\ \text{and } \; & H(\mathbf{z}) = 0, \qquad\quad G(\mathbf{z}) \le 0 \qquad\quad (\text{CCs}). \end{aligned} \tag{3}$$

The corresponding ESPC can be decomposed into N necessary conditions, one for each subproblem, and another necessary condition on the CCs and CVs. Because finding an ESP of a subproblem is equivalent to solving a MINLP, the ESP search of the i^{th} condition can be formulated as the solution of the following:

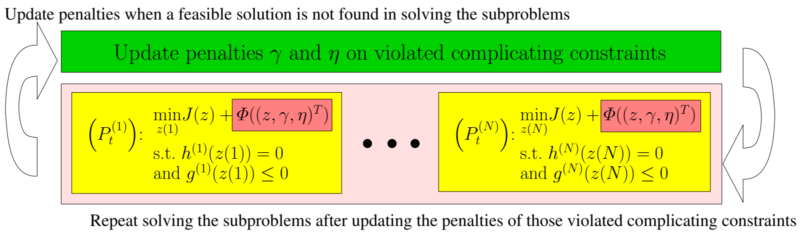

Update penalties when a feasible solution is not found in solving the subproblems
Update penalties γ and η on violated complicating constraints
$\left(P_t^{(1)}\right)$: $\min_{z(1)} J(z) + \Phi((z,\gamma,\eta)^T)$ s.t. $h^{(1)}(z(1)) = 0$ and $g^{(1)}(z(1)) \leq 0$
$\left(P_t^{(N)}\right)$: $\min_{z(N)} J(z) + \Phi((z,\gamma,\eta)^T)$ s.t. $h^{(N)}(z(N)) = 0$ and $g^{(N)}(z(N)) \leq 0$
Repeat solving the subproblems after updating the penalties of those violated complicating constraints

Fig. 5. Partition-and-resolve for finding a constrained local minimum of P_t

$$\left(P_t^{(i)}\right): \min_{\mathbf{z}(i)} f(\mathbf{z}) + \Phi(\mathbf{z}, \boldsymbol{\gamma}, \boldsymbol{\eta}) \text{ s.t. } h^{(i)}(\mathbf{z}(i)) = 0 \text{ and } g^{(i)}(\mathbf{z}(i)) \leq 0, \quad (4)$$

where $\Phi(\mathbf{z}, \boldsymbol{\gamma}, \boldsymbol{\eta}) = \boldsymbol{\gamma}^T |H(\mathbf{z})| + \boldsymbol{\eta}^T G(\mathbf{z})^+$ is the weighted sum of the CC functions. For simplicity, the CVs in (3) have been formulated as CCs in (4).

Figure 5 illustrates the concept. After decomposing P_t into N subproblems, the process iterates between solving the subproblems and re-weighting those violated CCs until a feasible solution to P_t is found.

Next, we summarize the design of CPOpt [4], which implements the approach in Figure 5. CPOpt has the following steps.

a) It first analyzes a COP represented in AMPL and decomposes its constraints by their localities in order to minimize the number of CCs across the subproblems. An *index vector* in an AMPL model is a finite ordered array of discrete elements that are used to index variables and constraints. This convenient representation allows a suitable partitioning of the constraints to be found by enumerating the different combinations of index vectors. Another benefit is that the overhead of finding the partitions is much smaller than that of clustering the constraints because the number of index vectors is very small (usually ≤ 5).

b) CPOpt find the proper granularity of partitioning by an iterative procedure. A *partitioning index vector* (PIV) of an AMPL model is defined to be an index vector used for partitioning the constraints. Using a PIV, an N partition is a collection of the disjoint subsets of the PIV, $S_1, \cdots, S_N$, where PIV $= S_1 \cup \cdots \cup S_N$. In general, the Cartesian products of the PIVs can be partitioned along one or more vectors into subsets. An interesting property observed from the benchmarks tested is that the best combination of the PIVs for a problem instance that leads to the minimum number of CCs is independent of the number of partitions N [4]. Hence, to determine the best combination, all possible combinations of PIVs are enumerated by fixing an arbitrary value of N.

Based on the PIVs selected and a convex relationship between the number of partitions and the total solution time, CPOpt finds the optimal number of partitions by a binary search. Starting from a large number of partitions, one subproblem in the set is solved to estimate the overhead of solving all the subproblems in the set, while ignoring the overhead of resolving the CCs (a lower-bound estimate). Then, the number of partitions is reduced by half and the procedure repeated until the estimated overhead starts to increase. The number of partitions found at that point is very close to the optimal number [4].

c) CPOpt then calls an existing solver to solve each partitioned subproblem and resolves those violated CCs iteratively by updating their penalties. After solving each subproblem, γ and η on the violated CCs are increased by some constant rates. To avoid getting stuck in infeasible stationary points of the penalty function, periodic decreases of γ and η are allowed, which "lower" the barrier in the penalty function and allow local descents with respect to z to escape from an infeasible region. The process is repeated until all the constraints are satisfied or the maximum number of iterations is exceeded.

One of the difficulties encountered in CPOpt is that the PIVs alone may not fully exploit the constraint locality of a problem. For example, the benchmark *WM_CFy* from the NLP suite has 8,709 variables and 11,338 constraints and cannot be solved by SNOPT in over 3,600 sec. The best PIV leads to 2,646 CCs, which are too many to be handled. A close observation of its CCs leads to some common expressions and operators that can be further partitioned into multiple LCs with a small number of CVs. For example, when the constraints are partitioned by PIV $i = \{1, \ldots, 2N\}$ into two disjoint subsets $S_1 = \{1, \ldots, N\}$ and $S_2 = \{N+1, \ldots, 2N\}$, the CC $\left(\max_{i=1,\ldots,2N} x_i\right) / \sqrt{\sum_{i=1}^{2N} y_i^2} \leq 0$ can be rewritten into $\max\{\left(\max_{i=1,\ldots,N} x_i\right), t_{2,1}\} / \sqrt{\sum_{i=1}^{N} y_i^2 + t_{2,2}} \leq 0$. This is an LC in the first subproblem with two CVs $t_{2,1} = \max_{i=N+1,\ldots,2N} x_i$ and $t_{2,2} = \sum_{i=N+1}^{2N} y_i^2$ computed in the second subproblem. A similar LC can be constructed in the second subproblem using two CVs in the first subproblem. When *WM_CFy* is partitioned into six subproblems in this way, each can be solved in 5 sec. by SNOPT.

3 Generation of Good Initial Starting Points

Our goal is to derive a simplified version of the original COP whose solution can be extended to provide a good starting point. This is done by solving one or more simplified versions of the COP and by generalizing the solutions found. In this section, we present three methods based on the problem characteristics.

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 2b, the CVs 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 some convergence is observed. Let $\mathbf{v}_i$ be the vector containing the values of the CVs after solving a simplified problem with N_i sampled constraints. Convergence is then determined as follows:

$$(1-a)|\mathbf{v}_{i-1}| - K \quad \leq \quad |\mathbf{v}_i| \quad \leq \quad (1+a)|\mathbf{v}_{i-1}| + K, \quad 0 < a < 1, \qquad (5)$$

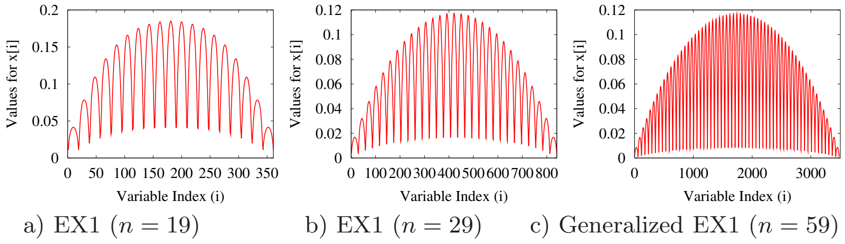


Fig. 6. The common features of the solutions of three simplified versions of EX1

where K is a constant in case $|\mathbf{v}_i|$ is very close to zero. If $|\mathbf{v}_i|$ is within the bound $(1 \pm a)|\mathbf{v}_{i-1}|$ or if N_i exceeds 10% of the total constraints, then 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 solved in the simplified NLP, we interpolate the values of variables in the original NLP that are not solved in the simplified version into a starting point. While fixing the CVs, we then solve the original NLP by parallel decomposition. Since no feasible solution may be found when the CVs are fixed, we relax the CVs so that they can be solved within some bounds. Once a feasible solution is found, we gradually tighten the bounds until the CVs are consistent across the subproblems. Overall, the approach is used in solving 34 (or 13.3%) of the 255 benchmarks tested (indicated by A in Table 2).

b) Constraint sampling and extrapolation. For an NLP whose constraints are related to their neighboring constraints by some common variables, uniformly sampling the constraints may result in a new problem with independent constraints that do not have common variables. For example, the problem DTOC6 from CUTE has constraints of the form $y[i] - y[i+1] + \exp(x[i]) = 0$ for $1 \leq i \leq 5000$, and the i^{th} constraint is related to the $i+1^{\text{st}}$ constraint by $y[i+1]$. 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 any insights on the starting point of the original problem. For these NLPs, instead of uniformly sampling the constraints, we generate a version using a subset of their contiguous constraints and eliminate those variables that are not in the sampled constraints. For instance, in DTOC6, we generate a problem with 500 constraints that are related to variables $y[1], \ldots, y[500]$.

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. This works in DTOC6 when we assign $y[j]$ in the sampled problem to $y[j']$, $j' = 10(j-1)+1$, in the original problem. This is used in solving 128 (or 50.2%) of the 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 to be identified. For example, EX1 of the NLP suite has variables $x[i]$, $1 \leq i \leq n^2$ and $n = 59$. When we solved three simplified problems with $n = 19,\ 29,\ 39$, the solutions have, respectively, 19, 29, and 39 peaks with a similar envelope (Figures 6a and 6b). Hence, we generalize the initial solution of EX1 to have 59 peaks and the same envelope as that of $n = 39$ (Figure 6c). This technique is used in solving 36 (or 14.1%) of the benchmarks (indicated by B2 in Table 2).

c) Constraint relaxation for MINLPs. For MINLPs, we generate a good starting point by solving it 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 30 (or 11.8%) of the benchmarks (indicated by C in Table 2).

To effectively analyze the constraint structure of a COP written in AMPL, we have developed a recursive descent parser (using the Perl module *ParseRecDescent* from `www.cpan.org`) that can automatically parse its representation and identify the PIVs and the variables in the constraints. The automated CPOpt rewrites the original AMPL program into subproblems in AMPL, generates one or more AMPL programs with sampled constraints, solves the sampled problems to generate good starting points, solves some subproblems in order to determine the optimal granularity, and implements the approach in Section 2 to find CLMs.

4 Analysis of the Starting Points Found

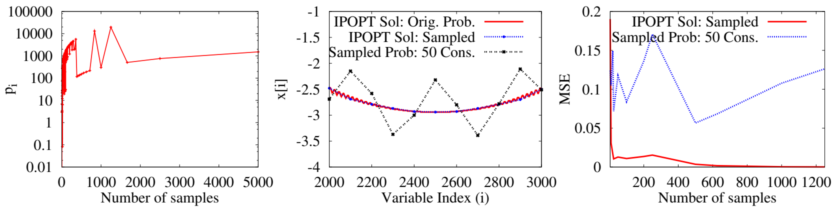
Due to space limitation, we informally analyze the quality of the starting points.

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

$$p_0^* \leq p_1^* \leq \ldots \leq p_i^* \leq \ldots \leq p_j^* \leq \ldots \leq p_N^*. \tag{6}$$

Let $p_i \geq p_i^*$ be a local minimum solution of P_i found by some solver. Unlike (6), there is no similar relation between p_i and p_j when 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 [2] 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.



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

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

With the smoothness property, our proposed techniques in Section 3 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.

5 Experimental Results

We compare the performance of CPOpt to that of other leading solvers, using the 255 NLP and MINLP benchmarks described in Section 1.

Table 2 summarizes those problems that can be solved by CPOpt but cannot be solved by SNOPT (an NLP solver using SQP) [1] and FilMINT (a MINLP solver using branch-and-cut and filterSQP). These are generally large problems, and CPOpt can solve each in under 5 minutes (33 minutes for space-960)

However, there are 27 (or 10.6%) benchmarks that cannot be solved by CPOpt. In particular, LUKVLE2 from the NLP suite has a poor locality. The best PIV leads to $10(N-1)$ CVs when partitioned into N subproblems, which are too many to be handled efficiently. All the other problems (LUKVLI2, TOP1-15x05, TOP1-30x10, TOP1-60x20, CATMIX400) not solvable by CPOpt have similar characteristics. There are also 21 problems that cannot be solved by SNOPT even for significantly simplified versions.[1]

Figure 8 compares the normalized time and quality of the 109 problems that can be solved by CPOpt as well as SNOPT or FilMINT. These problems are smaller problems that require very small amount of execution times. In general,

[1] Problems not solvable by SNOPT, even for simplified versions, include EX6, CONT6, CONT7, CONT8, CONT5_1, CONT5_2_1, CONT5_2_2, CONT5_2_3, CONT5_2_4, CONT5_400, TWOD, POROUS1, POROUS1_L, SEMICON1, SEMICON1_L, SEMICON2, LUKVLE4, LUKVLE6, LUKVLI6, TETRA4, TETRA5.

Table 2. Result on solving NLP and MINLP benchmark suites in Table 1. The experiments were performed on an AMD 2-GHz computer with RedHat Linux 3.4.6 and 2 GB memory. For each instance, n_c(*), n_v, and n_p are, respectively, the number of (*nonlinear) constraints, variables, and partitions; 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 3 for generating starting points; and Iter. is the number of iterations for resolving the violated CCs.

Benchmark	n_c(*)	n_v	n_p	n_s	Tech.	Iter.	Time	Sol	Benchmark	n_c(*)	n_v	n_p	n_s	Tech.	Iter.	Time	Sol
CUTE (Not Solvable by SNOPT in 3,600 sec.)							CPOpt		CUTE (Not Solvable by SNOPT in 3,600 sec.)							CPOpt	
dtoc1l	9990(0)	14985	10	1495	B1	1	22.50	155.69	dtoc2	3996(3996)	5994	10	598	B1	1	7.65	0.51
dtoc4	9997(4997)	14996	10	749	B1	1	18.27	10.74	dtoc6	5000(5000)	10000	10	1001	B1	1	40.75	6.76e5
hager2	5000(0)	10000	20	201	B1	1	10.17	0.00	hager4	5000(0)	10000	20	1001	B1	1	11.66	3.46
orthrdm2	2000(2000)	4003	10	10	A	1	10.13	155.96	orthregd	5000(5000)	10003	25	50	A	1	23.36	1524.21
orthrgdm	5000(5000)	10003	20	50	A	1	20.47	1513.99	orthrgds	5000(5000)	10003	20	20	A	1	18.58	1545.35
catenary	166(166)	496	15	54	B1	1	4.95	-5.45e5	dtoc5	4999(4998)	9998	20	999	B1	1	14.54	1.50
hadamard	256(128)	65	1	16	B1	1	5.98	1.01	kissing	903(903)	127	1	36	B1	1	65.55	1.20
orthregc	5000(5000)	10005	20	50	A	1	19.72	190.79									
NLP (Not Solvable by SNOPT in 3,600 sec.)							CPOpt		NLP (Not Solvable by SNOPT in 3,600 sec.)							CPOpt	
cont5	3717(236)	3953	1	517	B1	1	13.95	0.55	ex4	3717(3481)	7198	1	802	B2	1	8.45	0.08
lukvle1	49998(49998)	50000	10	500	B2	1	22.31	6.23	lukvle3	2(2)	50002	1	502	B2	1	3.14	65.12
lukvle5	49996(49996)	50002	100	502	B2	1	203.67	0.37	lukvle7	4(4)	50002	100	502	B2	1	351.98	-6.61e4
lukvle8	49998(49998)	50000	2	1000	B2	1	6.41	4.13e6	lukvle9	6(6)	50000	100	50	B2	1	366.62	4994.66
lukvle10	49998(49998)	50000	2	500	B2	1	6.35	1.77e4	lukvle11	33330(33330)	49997	2	497	B2	1	4.83	0
lukvle12	37497(37497)	49997	50	997	B2	1	123.42	7.72e4	lukvle13	33330(33330)	49997	40	497	B2	1	210.26	4.02e5
lukvle14	33330(33330)	49997	40	497	B2	1	225.75	3.80e5	lukvle15	37497(37497)	49997	2	997	B2	1	4.61	0
lukvle16	37497(37497)	49997	2	997	B2	1	4.39	0	lukvle17	37497(37497)	49997	10	997	B2	1	22.71	7.14e4
lukvle18	37497(37497)	49997	5	997	B2	1	12.96	6.00e4	lukvli1	49998(49998)	50000	2	50	B2	1	6.30	0
lukvli3	2(2)	50000	1	500	B2	1	3.06	11.58	lukvli4	49998(49998)	50000	100	500	B2	1	369.26	2.01e4
lukvli5	49996(49996)	50000	2	500	B2	1	10.28	0.37	lukvli7	4(4)	50000	100	500	B2	1	365.34	-1.86e4
lukvli8	49998(49998)	50000	2	500	B2	1	8.75	4.13e-6	lukvli9	6(6)	50000	100	50	B2	1	353.80	4994.67
lukvli10	49998(49998)	50000	2	500	B2	1	8.69	1.77e4	lukvli11	33330(33330)	49997	2	497	B2	1	6.48	0
lukvli12	37497(37497)	49997	10	997	B2	1	19.64	0	lukvli13	33330(33330)	49997	3	497	B2	1	6.98	1.08e-8
lukvli14	33330(33330)	49997	40	497	B2	1	264.02	7.10e4	lukvli15	37497(37497)	49997	2	997	B2	1	4.62	0
lukvli16	37497(37497)	49997	2	997	B2	1	4.60	0	lukvli17	37497(37497)	49997	100	997	B2	1	255.70	1.92e-5
lukvli18	37497(37497)	49997	2	997	B2	1	4.49	0	WM_CFy	11338(2268)	8709	6	1184	B1	1	6.25	1.25
Nonsys (Not Solvable by SNOPT in 3,600 sec.)							CPOpt		Nonsys (Not Solvable by SNOPT in 3,600 sec.)							CPOpt	
bratu2d	61504(61504)	62500	1	10000	B1	1	26.02	0	bratu2d_l	248004(248004)	250000	1	10000	B1	1	23.53	0
bratu2dt	61504(61504)	62500	1	10000	B1	1	26.27	0	bratu2dt_l	248004(248004)	250000	1	10000	B1	1	26.27	0
bratu3d	27000(27000)	32768	1	512	B1	1	2.79	0	cbratu2d	123008(123008)	62500	1	10000	B1	1	59.69	0
cbratu2d_l	316808(316808)	160000	1	10000	B1	1	55.38	0	cbratu3d	11664(11664)	16000	1	125	B1	1	1.99	0
chemrctb	50000(49998)	50000	1	500	B1	1	2.53	0	porous2	61504(61504)	62500	1	10000	B1	1	29.29	0
porous2_l	248004(248004)	250000	1	10000	B1	1	29.34	0	semicon1	50000(50000)	50002	2	500	B1	1	30.32	0
MacMINLP (Not Solvable by FilMINT in 3,600 sec.)							CPOpt		MacMINLP (Not Solvable by FilMINT in 3,600 sec.)							CPOpt	
c-reload-q-104	3338(3130)	12906	13		C	42	360.60	-1.24	c-reload-q-49	1430(1332)	3292	7		C	10	68.43	-1.14
space-960-ir	3617(960)	2657	30		C	21	170.08	7.70e6	space-960	8417(960)	15137	30		C	20	2100.08	7.84e6
stockcycle	97(0)	480	6		C	2	5.90	1.20e5	trimlon6	72(12)	168	6		C	5	18.09	15.6
trimlon12	72(12)	168	12		C	47	347.27	98.7	trimloss5	90(5)	161	5		C	6	69.88	10.7
trimloss6	154(7)	345	6		C	8	72.94	22.15	trimloss7	154(7)	345	7		C	4	57.30	26.80
trimloss12	372(12)	800	12		C	52	348.98	150.85									

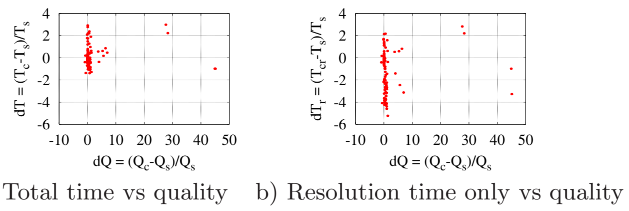


Fig. 8. Normalized time and quality of the 109 problems that can be solved by CPOpt as well as SNOPT or FilMINT

due to the additional overheads for partitioning, finding good starting points, and resolving violated CCs, CPOpt is slower in solving these problems (Figure 8a). However, if only the times for resolution are considered, then CPOpt is much faster in solving most of them (Figure 8b). Note that the normalized quality (dQ) of CPOpt for four problems is much worse when the solutions found by SNOPT or FilMINT (Q_s) are very close to zero.

References

1. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM review 47(1), 99–131 (2005)
2. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Programming 106(1), 25–57 (2006)
3. Dantzig, G.B., Wolfe, P.: Decomposition principle for linear programming. Operations Research 8, 101–111 (1960)
4. Wah, B.W., Chen, Y.X.: Solving large-scale nonlinear programming problems by constraint partitioning. In: van Beek, P. (ed.) CP 2005. LNCS, vol. 3709, pp. 697–711. Springer, Heidelberg (2005)
5. Wah, B., Chen, Y.X.: Constraint partitioning in penalty formulations for solving temporal planning problems. Artificial Intelligence 170(3), 187–231 (2006)