Report Number: wucse-2009-31

2009

Augmented Lagrangian Algorithms under Constraint Partitioning

You Xu and Yixin Chen

We present a novel constraint-partitioning approach for solving continuous nonlinear optimization based on augmented Lagrange method. In contrast to previous work, our approach is based on a new constraint partitioning theory and can handle global constraints. We employ a hyper-graph partitioning method to recognize the problem structure. We prove global convergence under assumptions that are much more relaxed than previous work and solve problems as large as 40,000 variables that other solvers such as IPOPT [11] cannot solve.

2009-31

Augmented Lagrangian Algorithms under Constraint Partitioning

Authors: You Xu, Yixin Chen

Corresponding Author: chen@cse.wustl.edu

Abstract: We present a novel constraint-partitioning approach for solving continuous nonlinear optimization based on augmented Lagrange method. In contrast to previous work, our approach is based on a new constraint partitioning theory and can handle global constraints. We employ a hyper-graph partitioning method to recognize the problem structure. We prove global convergence under assumptions that are much more relaxed than previous work and solve problems as large as 40,000 variables that other solvers such as IPOPT~\cite{ipopt} cannot solve.

Type of Report: Other

SOLVING LARGE-SCALE NONLINEAR PROGRAMMING PROBLEMS THOUGH CONSTRAINT PARTITIONING*

YOU XU AND YIXIN CHEN †

Abstract.
We present a novel constraint-partitioning approach for solving continuous nonlinear optimization based on augmented Lagrange method. In contrast to previous work, our approach is based on a new constraint partitioning theory and can handle global constraints. We employ a hyper-graph partitioning method to recognize the problem structure. We prove global convergence under assumptions that are much more relaxed than previous work and solve problems as large as 40,000 variables that other solvers such as IPOPT [11] cannot solve.

Key words. nonlinear programming, constraint partitioning, exact penalty

AMS subject classifications. 90C06, 90C30, 65K05

1. Introduction. Nonlinear optimization is an important problem that has abundant applications in science and engineering. In this paper, we study nonlinear programming problems formulated as follows:

$$P: \quad \min_x \quad f(x), \qquad \text{subject to} \quad h(x) = 0 \ \text{ and } \ g(x) \le 0, \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^n \to \mathbb{R}^r$ are twice continuously differentiable functions. We denote by $\mathcal{F} = \{x \in \mathbb{R}^n : g(x) \le 0, h(x) = 0\}$ the feasible set that we assume nonempty, by $\mathcal{G}_c$ and $\mathcal{L}_c$ the set of global solutions and the set of local solutions of Problem P. We further define $\mathcal{C} = \{h(x), g(x)\}$ as the constraints set and $\mathcal{X} = \{x_i, i = 1, \ldots, n\}$ as the variable set. Since "large-scale" is a vague term, in this paper, it refers to a large n and $m + r$.

Algorithm presented in this paper are designed to solve P by solving a sequence of specially constructed smaller-scale (in both variable number and constraint number) constrained nonlinear optimization problems which can be solved in parallel. By using partition, we aims to make the problem solver more scalable.

1.1. Observation of the problem structure. Our key observation is that most application-based NLPs have structured and localized arrangements of constraints. To formalize our observation and exploit the sparse nature of nonlinear optimization problems, we introduce the definition of related variable set.

DEFINITION 1.1. *For a given close form function f, the related variable set $V(f)$ is a set of variables involved in the close-form function f.*

Here is a concrete example. An NLP problem has five variables $\{x_1, \ldots, x_5\}$. The objective function f is $2x_1 + \sin(x_2) - x_4x_5$. Thus, $V(f)$ is $\{x_1, x_2, x_4, x_5\}$. This definition of related variable set of a function can be easily generalized to the a set of functions. For example. $V(f_1, f_2, ..., f_n)$ is defined as the union of $V(f_1), V(f_2), \ldots V(f_n)$. For a general nonlinear optimization problem P, we define the variable set $\mathcal{V}$ as the related variable set of this problem where $\mathcal{V} = V(f, g, h)$[1]. In the mean while, we define all the constraints as set $C = \{g, h\}$.

*This work was supported by Department of Energy

†Department of Computer Science, Washington University (chen@cse.wustl.edu).

[1]This definition is only valid for the functions that have a close form representation. This is well-defined as we do not deal with other type of functions in this paper.

Based on the definition of related variable set, we can number all constraints and variables and denote them by $c_1, c_2, \ldots, c_m$ and $x_1, x_2, \ldots, x_n$. To visualize this idea, we plot a *relationship graph* using constraints' indies as row and variables' indies as column. A dot is printed on (i, j) if and only if $x_j \in V(c_i)$.

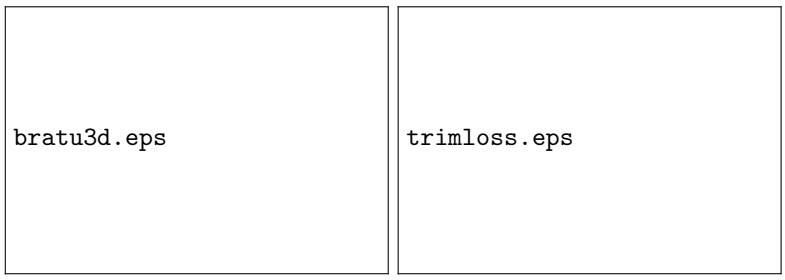

FIG. 1.1. *Constraint-Variable relationship for Bratu3D and Trimloss*

Figure 1a shows the constraint-variable relationship graph of a problem in CUTE library [3], Bratu3D problem. The coefficient matrix of Bratu3D is a well-known symmetric sparse matrix [4]. The corresponding relationship graph is also sparse, symmetric and has strong locality. Figure 1b again shows the constraint-variable relationship graph of a problem in CUTE library, Trimloss problem. It's a convex square root formulation of a non-convex MINLP arising from trim loss minimization in paper industry [3]. It is also sparse. For any given constraint, it is only related to a small set of variables.

Figure 1a, 1b and plenty of other problems in both CUTE library [3] and other application show that for nonlinear optimization problems, constraints are usually localized to a small set of variables, and this is usually because the spatial or temporal nature of the problem, as constraints are usually local spatial or temporal restrictions. Therefore, one intuitive idea is to reorder or regroup constraints and variables such that all constraints in one group are only related to a small set of variables. We use the term "partition" to denote such a regrouping procedure.

For some problems, partitioning strategies are straightforward. Take the "Distribution of Electrons on a Sphere" (ELEC) problem in COPS benchmark [5] as an example. Given n_p electrons, find the equilibrium state distribution (of minimal Coulomb potential) of the electrons positioned on a conduction sphere. The problem can be formulated as

$$\min f(x, y, z) = \sum_{i=1}^{n_p-1} \sum_{j=i+1}^{n_p} ((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i + z_j)^2)^{-\frac{1}{2}}, \tag{1.2}$$

$$\text{s.t.} \quad x_i^2 + y_i^2 + z_i^2 = 1, i = 1, \ldots, n_p \tag{1.3}$$

In this problem, all constraints are highly localized and only related to three variables which are the 3-D coordinates for electron. Since all constraints are localized to three related variables, when other variables are fixed, we can minimize the objective function for every three variables subject to related constraints without affecting other variables and constraints. Moreover, this minimization strategy has the potential to be implemented in parallel. This simple yet elegant strategy is surprisingly useful as we will show the detailed result in section 5.

Generally, for large scale NLP problems, localized arrangement of constraints and variables provides opportunities for reducing the computational cost. In the next section, we will show how to partition constraints and how to constraints that are not localized.

1.2. The constraint partitioning approach. Starting from a simple case, let's first give a formal definition of block separable constraint partition, which covers the constraint structures like in ELEC.

DEFINITION 1.2. *For a given nonlinear optimization problem P with constraints $\mathcal{C} = \{c_i(x)\}$, a subset $\mathcal{S}$ of $\mathcal{P}(\mathcal{C})$ is a* block separable constraint partitioning *if and only if $\mathcal{S}$ is a partition of $\mathcal{C}$ and $V(\mathcal{S})$ is a partition of X. Thus, $\forall P_i, P_j$ in $\mathcal{S}$, $P_i \cap P_j = \emptyset$, $\bigcup P_i = \mathcal{C}$ and $V(Pi) \cap V(P_j) = \emptyset, \bigcup V(P_i) = \mathcal{V}$.*

For this case, we propose a strategy that can solve the original problem via resolving constraints once per partition. However, more often than not, there are some constraints that related to variables in different groups and there is no way to avoid. Again let's take a problem in CUTE library for example. The EIGMINB problem in CUTE is to find the eigvector of a given symmetric matrix A, that is, to find a unit vector q and scalar d such that $Aq = dq$ for which d is least. This problem is formulated with a constraint $\|q\| = 1$ which evolves all but one variable d. Other constraints also evolve both q and d. This constraints' structure prevents the block separable constraint partitioning proposed above. To handle this under our partition framework, we introduce the concept of general constraint partition as

DEFINITION 1.3. *For a given nonlinear optimization problem P with constraints $c_i(x)$, a subset $\mathcal{S}'$ of $\mathcal{P}(c_i)$ is a* general constraint partitioning *if $\mathcal{S}'$ is a partition of $\{c_i\}$ with $m+1$ parts and $\{V(\mathcal{S}'_j), j = 1, \ldots, m, m \in \mathbb{Z}^+\}$ is a partition of X.* Based on these two definitions, we proposed a problem partitioning and resolving framework to solve general NLP problems.

We first analyze the problem structure and find general constraint partition with $m + 1$ loosely coupled groups. According to our definition, all the constraints within the same group in first m groups are only related to a small portion of variables and any constraints that in different groups do not share any variable. Therefore, all the variables are hereby be partitioned into the first m non-intersect groups. The only difference between the definition of general constraint partition and block separable constraint partition is that we allow the $m+1$th group of constraints that are related to variables across different variable partitions. This relaxation gives us the flexibly to handle problems like EIGMINB where some constraints are related to all variables and there is no way to partition constraints into m independent groups with independent related variables. In the EIGMINB example, By putting constraint $\|q\| = 1$ to the $m+1st$ group, other constraints are now can be rearranged as an block separable partition. In this paper, we name all constraints reside in the $m+1$st group as global constraints. To the contract, all the constraints in the first m groups are correspondingly defined as local constraint. In section 3, we propose a constraints resolving strategy using a novel exact penalty theory proposed in [1] to solve nonlinear optimization problems and make full use of the local constraints. Our methods consists of two parts: 1) to exploit the problem structure and constraints locality to get general constraint partition, 2) iteratively solve smaller scale problems and ensure the solution quality via global constraint resolving.

DEFINITION 1.4. *The augmented Lagrange function $\mathcal{L}$ of P is:*

$$L(x, \lambda, c) = f(x) + \lambda h(x) + \frac{c}{2}\|h(x)\|^2, \tag{1.4}$$

where $c > 0$.

The rest of this paper is organized as following: In section 2, we briefly conclude the previous work and explained why a constraint partitioning approach is necessary. In section 3, we introduce our algorithm framework as well as the theoretical proof to support our approach based on much relaxed assumption. Then we introduce the system implementation briefly. Finally we get the conclusion that constraint partition framework is a promising approach in solving NLPs by showing some experimental results.

2. Related Work.

2.1. Existing parallel methods in optimization. In this section, we survey existing methods for solving large scale optimization problems in the manner of partitioning or decompose into subproblems.

Depending on different ways in dealing with constraints and the assumption over the convexity of the original problem, parallel algorithms for solving constrained optimization problems usually falls into three categories.

1. Unconstrained optimization with variable distributed

Algorithms in this category includes parallel variable distribution [6] and parallel gradient descending [7]. Those algorithm usually exploits the parallel structure in objective function of the unconstrained optimization problem. Usually, necessary information like gradients are distributed to p processors where each of them handle an independent part of variable set.

In parallel variable distribution algorithm, after variables are distributed among p processors, PVD introduced a 'forget-me-not' term to allow the remaining variables to change in a restricted fashion while solving a subproblem on a processing j with the primary responsibility for updating its own block of variables. While this idea allows parallel processor to obtain a better minimum. This algorithm is linearly convergence towards the global minimal point under the assumption that objective function f is strongly convex. In parallel gradient distribution algorithm, each processor takes its own block of variables and corresponding gradient information. One of the important features of this algorithm is that all subproblems can be solved using different approaches such as a descent, Newton, quasi-Newton or conjugated gradient algorithm under the unified framework with convergence proof.

Other decomposition algorithm that are applicable to solve unconstrained optimization problems includes the decomposition algorithm proposed by Tseng [10]. It assumes that original cost function is separable and convex. Although this condition is more restrictive than PVD, it can handle linear constraints that are not necessarily block separable, we will discuss this algorithm later.

Generally, algorithms in solving unconstrained optimization problems in a parallel manner motivates us to develop parallel algorithms for constrained cases. The proposed algorithm is similar to decomposition algorithm in the unconstrained cases and have the same linear convergence rate. Moreover, we can also handle constrained cases where the constraints might be nonlinear.

2. Constrained optimization with block-separable constraints.

This category contains various of algorithms as block separable constraints are naturally eligible for parallel computing. Usually subproblem P_l on processor l is defined as objective function f with variables other than block l fixed. It also only contains constraints in the current block l. As in block separable constraints problems, variables in each part are independent, objective function can be solved by solving

independent subproblems. The block Jacobi method [8], the updated conjugate subspaces method, the coordinate decent method and the parallel gradient distribution all fall in this category. As each processor solves its own subsystem and there is no communication between the processors until the synchronization step, these methods are highly parallel. Since this strategy neglects the global properties of the whole system, it might only gain a limited improvement during iteration.

A different family of algorithm that considers global properties includes Parallel Variable Distribution for convex and non-convex objective functions [6, 9]. Instead of solving subproblem independently, PVD solves the l-th problem with block variables as 'primary' variables and $p-1$ 'secondary' variables representing possible step size that x_k can move in block k other than block l while the search direction is pre-defined.

In conclusion, all parallel algorithms in this category shed lights on the development of parallel algorithm for general constraints. However, as they all rely on the assumption of the block separable structure, their applications are highly restricted. For instance, there are totally 194 large scale problems in CUTE library (with number of constraints and number of variables larger than 100) that has a sparse constraint-variable relationship (sparse factor ¡ 0.25). However, only 9 of them, namely, airport, ncvxqp4, ncvqp5, ncvqp6, static3, steenbra, steenbrb, steenbrd and steenbrf are block separable. For other sparse problems, although most of the constraints are block separable, there are some constraints that make the block separable structure impossible (for example, some constraints can involve all variables such that it can not be put in a smaller block). This observation motivates us to develop a general method in handling constraints.

3. Constrained optimization with general constraints.

As mentioned above, about 95% of the large scale sparse optimization problems in CUTE fall in to this category. The partitioning approaches for solving problems in these category can be classified as two families. The first one is to divide constraints into different processors and solve subproblems with less (and therefore easier) constraints. Parallel Constraint Distribution and Parallel proximal minimization are two representative approaches. Since for each subproblem, it involves the same number of variables as the original problem, they do not exploit the parallel and sparse nature of nonlinear optimization problems.

The other family is to handle the global constraints explicitly as an item in biased function and only handle block separable constraints explicitly. However, for general nonlinear convex optimization problems where constraints can be nonlinear and non-block separable, no numeric result was reported yet. Previously PVD [6] and PVDCO [9] both proposed this idea without further discussing. In this paper, we will report some preliminary numeric results in solving large scale nonlinear optimization problems.

2.2. Existing partitioning methods. In this section we briefly survey the existing methods in partitioning and solving large-scale nonlinear optimization problems.

Moreover, for nonlinear optimization problem general constraints, in the previous work, the block separable structure is *a prior*. However, in the real world problem, the constraints' structure needs to be analyzed to extract non-block separable constraints and partitioning the constraints into blocks. In this paper we first proposed a automatic partitioning method differ than some previous work that use a certain index [**?**] or bi-part graph partitioning.

3. Algorithm Framework. We assume that in all the cases below, objective function f and constraints h are all continuous and differentiable. We consider an NLP problem with general constraint partitioning as

$$\begin{aligned} \min \quad & f(x) \\ subject \ \ to \ \ & H_1^t(x) = 0 \\ & H_2(x) = 0, \end{aligned}$$

where $H_1^t (1 \leq t \leq m)$ and H_2 combined together is a general constraint partitioning.

Unlike the traditional unconstrained optimization method which transforms all the constraints to a part of the unconstrained objective function, we only transform H_2 to objective while maintaining $H_1(x)$ still as constraints as P':

$$\begin{aligned} \min f(x) + \lambda H_2(x) + \frac{1}{2}\|H_2(x)\|^2 \\ subject \ \ to \ \ H_1^t(x) = 0. \end{aligned}$$

P' now has block separable constraints.

For problem P', we will later prove in Lemma 2 that the solution of P when c is sufficiently large is indeed the solution to the original problem P. The discussion in Chapter 2.4 of [1] shed the light on the proof. The basic idea is to use find a partition of x as

$$x = [x_1, x_2]$$

and denote x_2 by x_1 via system $H_2(x_1, x_2) = 0$. Therefore, the local minimal of P' in augmented Lagrange iteration is equivalent to the unconstrained cases. The rational behind this is that the constrained problem can be solved quite satisfy and possibly even more easily that the problem of unconstrained minimization of the ordinary augmented Lagrangian [1]. The other reason is that we can solve the problem in a partitioned manner by solving even smaller subproblems.

Now that we have problem P', the challenge is to solve P' efficiently via the augmented Lagrange iteration. Parallel variable distribution with constraints is a good way to solve it, but instead of employing the SQP method to solve subproblems, we directly call existing solvers to find the KKT point for subproblems and do not care too much about the underlying method in solving subproblems.

First we define subproblem P_t as

$$P_t : \qquad \min \quad f(x_t) \tag{3.1}$$

$$subject \quad to \quad H_1^t(x_t) = 0 \tag{3.2}$$

where $x_t = V(H_1^t)$ is a subset of x and $f(x_t) = f(x)$ where all variables except in x_t are fixed to a value. That is, subproblem x_t only aims to optimize the objective function on a small partition of variables instead of the complete variable space x. Depends on the number of the partition and the size of the partition, the subproblem can be significantly smaller than the original problem. Given the fact that general nonlinear optimization problem is NP-complete, the time required in solving a smaller subproblem can be exponentially smaller than solving the original problem.

In practice, there are two ways to utilize the solution of the subproblems. One way to to solve all subproblems one by one and use the variable value updating from

the previous subproblem as the fixed value in the next iteration. In this manner, we prove the result that every accumulation point generate by this procedure is the stationary point of the original problem. Moreover, if every time the subproblem solver yields the strict minimal value, then the accumulation point is the strict local minimal.

In practice, since stage search are performed in parallel on each subspace, for a given starting point x_0, every stage tries to solve subproblem P_t on subspace $\mathcal{X}_i$. We denote the solution on stage t by x_0^t and the distance vector from x_0^t to x_0 by d_k. Since each solution is collected through parallel stage search, different strategies can be used to put the solution together as the next point moving from x_0. We might use cubic spline interpolation which is similar to SQP method. In this paper, we simply choose next point as $x = (x_0 + d_1 + d_2 + d_t + \ldots + d_N)$. If $f(x) < f(x_0)$, we adopt x as the next starting point in the parallel stage search. Otherwise we pick d_i where $d_i = \arg\min f(x_0 + d_i)$. Therefore, the parallel stage search will always find a new point to reduce the function value unless every $f_{x_0^t}$ in stage search equals to $f(x_0)$.

We will prove later that similar to the result in **??**, the procedure in Figure **??** generates fixed points that are necessary but not sufficient to satisfy **??**. Hence, in [**?**] and [**?**], constraint partitioned simulated annealing are used to escape from infeasible local minimal of the $\updownarrow_1$-penalty function. However, in certain cases, we can proof that the stationary point obtained by our search process is actually the feasible point and thus the local minimum of the objective function in according to the ESPC condition. One of the trivial example is that the objective function f is linear and there is no global constraints. To optimize a linear function $f = \sum_{i=1}^{n} \omega_i x_i$ without global constraints is equivalent to optimized N subproblems with partitioned local constraints and partitioned linear objection function $\sum_{jinV_t} \omega_j x_j$ separately. In the next section we will further formalize this result based on block separable constraint partitioning framework.

3.1. The convergence of our algorithm. In this section we present a proof of global convergence of the partitioning and resolving algorithm of Figure 3 when applied to problem P. We start from the simplest cases where Problem P has block separable constraint partition. These problems are **without** global constraints. This category contains lots of problem such as AIRPORT in CUTE, ELEC in COPS and have significant application in real world. We shall make use of the following assumptions on Problem P and establish some lemmas based on problems with block separable constraint partition. Later we will show how to solve problems with general constraint partition in the framework of block separable constraint partition.

Standard Assumption

Assumption 1. The feasible set $\mathcal{F}$ is compact on $\mathbb{R}^n$ (This implies non-emptiness).

Assumption 2. The objective function f takes a finite number of values on $\mathcal{L}_c$

Assumption 3. The EMFCQ (extended Mangasarian-Fromowitz constraint qualification) holds, namely, $\nabla h_j(x), j = 1, \ldots, r$ are linearly independent and there exists a $z \in \mathbb{R}^n$ such that

$$\begin{aligned} \nabla g_i(x)^{'} z < 0, \quad & i \in I_+(x), \\ \nabla h_j(x)^{'} z = 0, \quad & j = 1, \ldots, r. \end{aligned}$$

It can be noted that EMFCQ implies the MFCQ that the gradients of the active inequality constraints and the gradients of the equality constraints are positive-linearly

independent at an optimal point x^* for the original problem. It's obvious that this condition also implies the MFCQ condition for each subproblem at each stage search phase.

Consider a set of Problem $P_I \subset P$:

$$P_I : \quad \min_{x \in \mathbb{R}^N} \quad f(x), \qquad (3.3)$$
$$\text{subject to} \quad h(x) = 0 \ \text{ and } \ g(x) \leq 0,$$

with block separable constraint partition where constraint set $\{h(x)\}$ and $\{g(x)\}$ can be partitioned into N groups with non-intersective related variable set. We try to find local minima of P_I by iteratively solving a series of subproblems P_t defined as

$$P_t : \quad \min_{x_t \in \mathcal{F}_t} \quad f(x_t), \qquad (3.4)$$

where t indicates the index of constraint partitions.

Assume Problem P_I has compact feasible set $\mathcal{F}$. Since P_I has block separable constraint partition, $\mathcal{F}$ can be decomposed as $\mathcal{F}_1 \times \mathcal{F}_2 \times \ldots \times \mathcal{F}_t$. Thus, if every x_t is a solution to P_t, namely, $x_t \in \mathcal{F}_t$, we have $x = (x_1, \ldots, x_N) \in \mathcal{F}$. Hence, x is a feasible solution to P_I.

Thus, the only concern is optimality. To prove this, we make an assumption that at every stage, the solution to subproblem P_I satisfies the necessary KKT condition. We prove the below two lemma that solution sequence generated by stage search algorithm has accumulative point and those accumulative points satisfy KKT condition.

Lemma 1 For Problem P with block separable constraint partition and an non-trivial objective function $f \in C^1$ with tight lower bound f^* in compact feasible set $\mathcal{F}$, the point sequences visited by parallel stage search algorithm has accumulative points.

Proof. As the objective function f is lower bounded and at every stage and restoration phase never occurs, the parallel stage search algorithm defined in Figure 1 will always generate next point with function value f' less than or equals to the current f. Thus, during the search, f will monotonically decreasing or keep unchanged. As objective function f is lower bounded, will finally converge to a finite value, say, f^*.

Lemma 2 (Local optimality) For continuous optimization problem P and standard assumptions holds. If P has block separable constraint partition with N partitions and therefore can be partitioned as P_t in stage search, for a point x^*, if each $x^*|_t$ is the KKT point of subproblem P_t if and only if x^* is the KKT point of P.

Proof. The Lagrangian function associated with Problem P is the function $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^r \to \mathbb{R}$ defined by

$$L(x, \lambda, \mu) = f(x) + \lambda' g(x) + \mu' h(x).$$

Under the assumption of the regularity condition, the *Karush-Kuhn-Tucker* (KKT) *triplet* for Problem P is a triplet $(x^*, \lambda^*, \mu^*) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^r \to \mathbb{R}$ such that $x^* \in \mathcal{F}$ and

$$\nabla_x L(x^*, \lambda^*, \mu^*) = 0, \qquad (3.5)$$
$$\lambda' g(x^*) = 0, \qquad (3.6)$$
$$\lambda \geq 0. \qquad (3.7)$$

Similarly, start from the KKT condition for subproblem P_t, we have

$$\nabla_{V_t} f(x_t^*) + \lambda_t^{'} \nabla_{V_t} g(x_t^*) + \mu_t \nabla_{V_t} h(x_t^*) = 0 \tag{3.8}$$

$$\lambda_t^{'} \nabla_{V_t} g(x^*|_t) = 0 \tag{3.9}$$

$$\lambda_t^{'} \geq 0 \tag{3.10}$$

We concatenate all N conditions over subproblem P_t by concatenating all λ_t and μ_t as λ^* and μ^*. Note that $\nabla_V f(x^*) = (\nabla_{V_t} f(x_1^*, \ldots, \nabla_{V_t} f(x_N^*)^T$, all N conditions can be rewrite collectively as

$$\nabla_x = 0, \nabla_V f(x^*) + \lambda^{*'} \nabla_V g(x^*) + \mu_t \nabla_{V_t} h(x^*) = 0 \lambda' g(x^*) = 0, \lambda \geq 0.$$

where $x^* = (x_1^*, \ldots, x_1^*)$, $\lambda^* = (\lambda_1, \ldots, \lambda_N)$ and $\mu_1, \ldots, \mu_N)$.

Remark: The proof of the above Lemma can also be drawn from ESP Theory developed in [12]. ESP condition is more general than KKT condition as it is capable for non-continuous programming. wh **Lemma 2'**. Under our general assumption, starting from any initial point x_0 in $\mathbb{R}^n$, the stationary or accumulative point of our algorithm satisfies KKT necessary condition.

Proof: We prove this by contradiction. Let's assume x^* as the stationary or accumulative point of algorithm. Since $\mathcal{F}$ is compact, x^* is well defined and $x^* \in \mathcal{F}$. Under our assumption, MFCQ holds at x^*. Thus, if x^* does not satisfy KKT necessary condition, since all $g_i(x^*)$ and $h_j(x^*)$ are local constraints and are satisfied during the stage search, the only possibility for x^* to violate the KKT condition is there doesn't exist constant $\mu_i > 0$ and γ_j such that

$$\nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{r} \gamma_j \nabla h_j(x^*) = 0.$$

Thus, $\nabla f \neq 0$, there exists at least one point x^{**} along the gradient direction of x^*, such that $f(x^{**}) < f(x^*)$ and $g(x^{**}) < 0$, $h(x^{**}) = 0$. We denote p by $x^{**} - x^*$ and further expand $p\nabla f$ as

$$p\nabla f = \sum_{i=1}^{n} p_i \frac{\partial f}{\partial x_i} = \sum_{t=1}^{m} p_t \nabla f_{X_t}$$

where p_t and f_{X_t} are the vector and gradient projected on space X_t. Thus, at least one $p_t \nabla f_{X_t} < 0$. This contradicts our assumption that at x^*, every subproblem already satisfies KKT condition.

Remark 2. This lemma insures us that our constraint-partitioning framework will output an local optimal solution for objective functions in $\mathcal{C}^1$ if we can solve every subproblem optimally. It also provides the theoretical proof of partition large problems into several subproblems and solve them independently when there is no global constraints. For some large-scale problems in the benchmark such as AIRPORT and ELEC, we use this approach to separate original problems into smaller parts and then call the same solver to solve subproblems iteratively until the objective value f keep unchanged. As shown in Section 5, amazingly it boosts the solving speed up to 9 times without any other optimization techniques applied to solver as in every optimization step in our framework, we significantly reduce the computational cost to evaluate objective function and gradient. Additionally, as guaranteed by this lemma, our solution quality are equally good as the solution given by other solvers.

It's also easy to see from Lemma 2 that if the feasible set F is convex and the objective function f is convex and second order differentiable, then by using Lemma 2, our algorithm also guarantees the global optimality given that every solver will provide local optimal with respect to each subproblem.

Till now, we established the lemma under the assumption that there is no global constraints involved, namely, problem P_I. At every stage in the algorithm for solving P_I, x is always a feasible solution to the original problem. However, in general problem P, when global constraints are involved, x might violated some global constraints and therefore not necessarily be the feasible solution to the original problem. To handle this and solve P in this framework, we put the violation of global constraints as a penalty item in objective function in the hope of solving the optimization and feasibly problem in a single shot. To show how we deal with separable and non separable constraints, we first define a sightly different problem P_Ω as

DEFINITION 3.1. *(Nonlinear Problem on Ω) Assume the general nonlinear optimization problem P_Ω*

$$\begin{aligned} P_\Omega: \qquad & \min_x \quad f(x), \\ & \textit{subject to} \quad h(x) = 0 \quad \textit{and} \quad g(x) \leq 0, \\ & x \in \Omega \end{aligned} \tag{3.11}$$

where Ω is a closed set and contains block separable constraints.

Our aim is to solve a biased problem defined as

$$\begin{aligned} P'_\Omega: \qquad & \min_x \quad f(x) + \rho\|h(x), g^+(x)\|, \\ & x \in \Omega \end{aligned} \tag{3.12}$$

where Ω is contains all block separable constraints.

Traditionally, exactly penalty method is defined in solving an unconstrained subproblem with biased objective function. The classic result of exact penalty method can be written as [**?**] the proposition below. The detailed proof can be found on [**?**].

Proposition 1 Assume that $x^* \in \mathbb{R}^n$ satisfies the second-order sufficient conditions for a local minimizer of problem P_Ω. Let $\lambda^* \in \mathbb{R}^m$, $\mu^* \in \mathbb{R}^r$ and η^* be the Lagrange multiplier vectors associated to h, g and constraints set Ω, respectively. For $\rho > \bar{\rho} = \|\lambda^*, \mu^*\|^{1/p}$, and $\rho_1 > \|\eta^*\|^{1/p}$, x^* is a strict unconstrained local minimizer of $P(x, \rho, \rho_1) = f(x) + \rho\|h(x), g^+(x)\|^{1/q} + \rho_1\omega(x)$, where $\omega(x)$ is the penalty function for Ω and $1/p + 1/q = 1$.

Based on this proposition, we now prove an important lemma which was first proved in [**?**] that exact penalty method can be applied to constrained cases.

Lemma 2 Let $x^* \in \mathbb{R}^n$ satisfies the second-order sufficient conditions for a local minimizer of problem P_Ω. There exist $0 < \bar{\rho} < \infty$ such that for $\rho > r\bar{h}o$, x^* is a strict local minimizer of P'_Ω where the definition of norms satisfies the same condition in Proposition 1.

Proof.

Proof. We consider the problem

$$\begin{aligned} P'_\Omega: \qquad & \min_x \quad f(x) + \rho\|h(x), g^+(x)\| + \rho_1\omega(x), \\ & x \in \mathbb{R}^n \end{aligned} \tag{3.13}$$

where $\rho, \rho_1 > 0$ and $\omega(x)$ is as defined in Proposition 1. Since x^* satisfies the second-order sufficient conditions for P_Ω, we know from Proposition 1 that x^* is a strict local

minimizer of 3.13. Therefore, there exists ϵ such that for all $x \in \mathcal{B}(x^*, \epsilon) \equiv \{x \in \mathbb{R}^n | \|x - x^*\| \leq \epsilon\}$,

$$f(x^*) < f(x) + \rho \|h(x), g^+(x)\| + \rho_1 \omega(x) \quad \forall x \in \mathcal{B}(x^*, \epsilon) \cup \Omega,$$

since $\omega(x) = 0$ for all $x \in \Omega$. Thus, for $\rho > \bar{\rho} = \|\lambda^*, \mu^*\|^{1/p}$, x^* is a strict local minimizer of P_Ω. □

Hence, to solve the original problem P, we can transform it to P_Ω and then use constraint partitioning approach we proposed for block-separable constraints to solve that. However, the only difficulty here is that in our algorithm assumption, the objective function is differentiable. Here the biased function is convex and continuous but not continuous. We further make the objective function differentiable on each segments. Other methods that make the objective function, for example, to use the continuous differentiable exact penalty function can also be used here. We use non-differentiable function as we don't want to involve too much complexity to the nonlinear objective function.

Now we introduce the projection theorem as a lemma here without proof. Detailed proof can be found in [2].

Lemma 3 Let $\mathcal{C}$ be a closed convex set and $\|.\|$ be the Euclidean norm, the projection mapping $f : \mathbb{R}^n \to \mathcal{C}$ defined by $f(x) = [x]^+$ is continuous and non-expansive, i.e.,

$$\|[x]^+ - [y]^+\| \leq \|x - y\|, \quad \forall x, y \in \mathbb{R}^n$$

Lemma 4 For every $x \in X$ and $z \in \mathbb{R}^n$, the function $g : [0, \infty) \to \mathbb{R}$ defined by

$$g(s) = \frac{\|[x + sz]^+ - x\|}{s}, \forall s > 0,$$

is monotonically non-increasing.

Proof. We take two scalars s_1 and s_2 with $s_2 > s_1 > 0$, and we show that

$$\frac{\|[x + s_2 z]^+ - x\|}{s_2} \leq \frac{\|[x + s_1 z]^+ - x\|}{s_1}$$

To simplify the proof, we define $y = s_1 z$ and $\gamma = s_2/s_1$, we further denote $x + y$ and $x + \gamma y$ by a and b. The inequality is written as

$$\|\bar{b} - x\| \leq \|\gamma \bar{a} - x\|$$

where $\bar{x}$ is the projections of point x on X.

We first consider some special cases. If $\bar{a} = x$, then as $s_1 < s_2$, clearly $\bar{b} = x$, so the equation holds. Also if $a \in X$, then $\bar{a} = a = x + y$, so equation above becomes $\|\bar{b} - x\| \leq \gamma \|y\| = \|b - x\|$, which again holds according to Lemma 3. Finally if $\bar{a} = \bar{b}$, then the equation also holds. Therefore, we only consider the case where $\bar{a} \neq \bar{b}$, $\bar{a} \neq x$, $\bar{b} \neq x$, $a \notin X$.

Let H_a and H_b be the two hyperplanes that are orthogonal to $\bar{b} - \bar{a}$ and pass through $\bar{a}$ and $\bar{b}$, respectively. Since $(\bar{b} - \bar{a})'(b - \bar{b}) \geq 0$ and $(\bar{b} - \bar{a})'(a - \bar{a}) \leq 0$, we have that neither a nor b lie strictly between the two hyperplanes H_a and H_b. Furthermore, x lies on th esame side of H_a as a, so $x \notin H_a$. denote the intersections of the line $\{x + \alpha(b - x) | \alpha \in \mathbb{R}\}$ with H_a and H_b by s_a and s_b, respectively. Denote

the intersection of the line $\{x + \alpha(\bar{a} - x) | \alpha \in \mathbb{R}\}$ with H_b by ω, we have

$$\gamma = \frac{\|b - x\|}{\|a - x\|} \geq \frac{\|s_b - x\|}{\|s_a - x\|} \tag{3.14}$$

$$= \frac{\|\omega - x\|}{\|\bar{a} - x\|} = \frac{\|\omega - \bar{a}\| + \|\bar{a} - x\|}{\|\bar{a} - x\|} \tag{3.15}$$

$$\geq \frac{\|\bar{b} - \bar{a}\| + \|\bar{a} - x\|}{\|\bar{a} - x\|} \geq \frac{\|\bar{b} - x\|}{\|\bar{a} - x\|} \tag{3.16}$$

Theorem 1a For every $x \in X$ there exists a scalar $s_x > 0$ such that

$$f(x) - f(x(s)) \geq \sigma \nabla f(x)'(x - x(s)), \quad \forall s \in [0, s_x],$$

where $x_s = [x - s\nabla f(x)]^+$.

Proof. We know that if z is the projection of y over convex set C, we have $(x-z)'(y-z) \leq 0$ for any $x \in C$. We plug in $x - s\nabla f(x)$ as y and $x_s = [x - s\nabla f(x)]^+$ as z, we have

$$(x - x(s))'(x - s\nabla f(x) - x(s)) \leq 0, \quad \forall x \in X, s > 0.$$

Hence,

$$\nabla f(x)'(x - x(s)) \geq \frac{\|x - x(s)\|^2}{s} \quad \forall x \in X, s > 0.$$

If x is stationary, the conclusion holds with s_x being any positive scalar, so assume that x is not stationary. Thus, $\|x - x(s)\| \neq 0$ for all $s > 0$. We also know that by mean value theorem that for all $x \in X$ and $s \geq 0$,

$$f(x) - f(x(s)) = \nabla f(x)'(x - x(s)) + (\nabla f(\xi_s) - \nabla f(x))'(x - x(s)),$$

where ξ_s lies on the line segment joining x and $x(s)$. Therefore, the equation can be rewritten as

$$(1 - \sigma)\nabla f(x)'(x - x(s)) \geq (\nabla f(x) - \nabla f(\xi_s))'(x - x(s)),$$

However, from Lemma 4, we have for all $s \in (0, 1]$,

$$\nabla f(s)'(x - x(s)) \geq \frac{\|x - x(s)\|^2}{s} \geq \|x - x(1)\| \cdot \|x - x(s)\|.$$

Therefore, the equation satisfied for all $s \in (0, 1]$ such that

$$(1 - \sigma)\|x - x(1)\| \geq (\nabla f(x) - \nabla f(\xi_s))' \frac{x - x(s)}{\|x - x(s)\|}.$$

Remark 1 (Partitioned search in convex subproblems). For block separable constraint partition problem P_I, if each region $\mathcal{F}_t$ is convex and objective function f is convex defined on a convex $\mathcal{D}$ with standard assumption holds, stage search algorithm converges to global optimal point of P_I.

Now we study a constraint partitioning algorithm that can efficient solve large-scale nonlinear optimization problems based on augmented Lagrange method. We

consider the classic nonlinear optimization problem with all equal constraints. The original problem P_o is defined as:

$$\min \quad f(x) \tag{3.17}$$
$$\text{subject} \quad \text{to}: \quad H_1(x) = 0 \tag{3.18}$$
$$H_2(x) = 0 \tag{3.19}$$

where H_1 contains all constraints that can be partitioned and H_2 are global constraints as we stated. We transform problem P_o to an Augmented Lagrange problem with partical constraints as P_L.

$$\min \quad f(x) + \lambda H_2(x) + \frac{1}{2} c \|H_2\|^2 \tag{3.20}$$
$$\text{subject} \quad \text{to}: \quad H_1(x) = 0 \tag{3.21}$$

There are plenty of related works talked about how to solve this problem P_L using existing solver. In our proposed algorithm, as argued in the previous section, we are trying to solve problem P_L in a partitioned manner, namely, to solve a series of smaller subproblems. The partitioned subproblem P_L^t where $t = 1, \ldots, r$.

$$\min \quad f(x) + \lambda H_2(x) + \frac{1}{2} c \|H_2\|^2 \tag{3.22}$$
$$\text{subject} \quad \text{to}: \quad H_1^t(x) = 0 \tag{3.23}$$

where H_2^t is the t-Th block of in the block separable constraints H_2.

4. Algorithm. In order to solve the problem above, we consider the following algorithmic models.

Initialization;
while *constraint violation larger than* η^* **do**
  while $\|\nabla L_P\| \geq \omega^k$ **do**
  end
  if $\|H_2(x^{(k)})\| < \eta^*$ $\nabla L_P \leq \omega^*$ **then**
    output $x^{(k)}$ as the solution;
  end
  if $\|H_2(x^{(k)})\| < \eta^k$ **then**
    $\lambda^{(k+1)} = \lambda^{(k)} + c * H_2(x^{(k)})$;
    $c^{(k+1)} = c^{(k)}$;
    $k = k + 1$;
  else
  end
end

Note that in the initialization step, an initial vector of Lagrange multiplier estimates λ_0 is given. The positive constants are specified as $c^0 = 1$. ω^k and η^k are two positive sequences with 0 as the limit. ω_* and η_* are pre-defined positive scalar that is close to 1 and $\tau > 1$.

5. Proof. In this section, we will prove three major results. 1, we will prove that the inner iteration will terminate in finite steps to find $x^{(k+1)}$. 2, we will prove that this method will converge to a local minimal x^* of P when stated from a neighborhood of x^*, namely, the local convergence result. 3, if there is a feasible solution for this problem and the function is lower bounded, under some mild assumptions, every limited point of the sequence $x^{(k)}$ is a KKT point of problem P. This result is usually called global convergence attribute.

The whole theory of our propositions are based on this assumption:

General Assumption Vector x^* is a strict local minimum and a regular point of the original problem P.

We first scratch out the local convergence theory for the original augmented la grange method (i.e. all the constraints are transformed into objective function with corresponding Lagrange multiplier.)

Lemma 1 Suppose we have general assumption holds and let $\bar{c}$ be a position scalar such that

$$\nabla^2_{xx} L_{\bar{c}}(x^*, x^*) > 0.$$

For two positive scalars ϵ and δ We draw a set D around x^*, for all (λ, c, α) in the set D defined by

$$D = \{(\lambda, c, \alpha) | (|\lambda - x^*|^2/c^2 + |\alpha|^2)^{1/2}) < \delta, \bar{c} \leq c\},$$

(1). there exists a unique vector $x_\alpha(\lambda, c)$ within $S(x^*; \epsilon)$ satisfying

$$\nabla_x L_c[x_\alpha(\lambda, c), \lambda] = \alpha$$

(2). The function x_α is continuously differentiable in the interior of D, and for all $(\lambda, c, \alpha) \in D$, we have

$$|x(\lambda, c) - x^*| \leq M(|\lambda - x^*|^2/c^2 + |\alpha|^2)^{1/2}.$$

.

(3). for all $(\lambda, c, \alpha) \in D$, we have

$$\|\tilde{\lambda} - x^*\| \leq M(\|\lambda - x^*\|^2/c^2 + \|\alpha\|^2)^{1/2},$$

where $\tilde{\lambda} = \lambda + c * h$.

Proof

We consider the system of equations in $(x, \tilde{\lambda}, \lambda, c, \alpha)$

$$\nabla f + \nabla h \tilde{\lambda} = \alpha$$

$$h(x) = (\lambda - \tilde{\lambda})/c = 0$$

By introducing a new variable t and define γ as $1/c$, we can rewrite the system as

$$\nabla f + \nabla h\tilde{\lambda} = \alpha$$

$$h(x) + t + \gamma x^* - \gamma\tilde{\lambda} = 0$$

We now define a continuously differentiable functions $\hat{x}$, $\hat{\lambda}$ as the function to variable γ and t. Now we have

$$\nabla f(\hat{x}) + \nabla h(\hat{x})\hat{\lambda} = \alpha$$

$$h(\hat{x}) + t + \gamma x^* - \gamma\hat{\lambda} = 0$$

We differentiable the above system with respect to t, γ and α. We obtain

$$\begin{bmatrix} \nabla_t\hat{x} & \nabla_\gamma\hat{x} & \nabla_\alpha\hat{x} \\ \nabla_t\hat{\lambda} & \nabla_\gamma\hat{\lambda} & \nabla_\alpha\hat{\lambda} \end{bmatrix} = A(t,\gamma,\alpha)\begin{bmatrix} 0 & 0 & I \\ -I & \hat{\lambda} - x^* & 0 \end{bmatrix},$$

where A is the inverted Jacobi an matrix of the system.

Now, we have for all t, γ and α such that $\|(t,\alpha)\| < \delta$ and $\gamma \in [0, 1/\bar{c}]$,

$$\begin{bmatrix} \bar{x} - x^* \\ \bar{\lambda} - x^* \end{bmatrix} = \int_0^1 A(\zeta t, \zeta\gamma, \zeta\alpha)\begin{bmatrix} 0 & 0 & I \\ -I & \hat{\lambda} - x^* & 0 \end{bmatrix}\begin{bmatrix} t \\ \gamma \\ \alpha \end{bmatrix} d\zeta$$

We know that A is bounded. Let μ be such that $|A| \leq \mu$ and take δ sufficiently small to ensure that $\mu\delta < 1$. We have

$$(|\hat{x} - x^*|^2 + |\hat{\lambda} - \lambda|^2)^{1/2} \leq \mu(|t,\alpha| + max|\hat{\lambda} - x^*|\gamma).$$

We denote $max|\hat{\lambda} - x^*|\gamma$ by κ and obtain that

$$|\hat{\lambda} - x^*| \leq \mu|(t,\alpha)| + \mu\gamma\kappa$$

Therefore, $\kappa \leq \frac{\mu}{1-\mu\gamma}|t,\alpha|$ for $\mu\gamma < 1$. We plug in this inequality to and get

$$(|\hat{x} - x^*|^2 + |\hat{\lambda} - \lambda|^2)^{1/2} \leq \left(\mu + \frac{\mu^2\gamma}{1-\mu\gamma}\right)|t,\alpha| \leq \frac{\mu}{1-\mu\delta}|t,\alpha|.$$

For sufficiently small δ, we have

$$(|\hat{x} - x^*|^2 + |\hat{\lambda} - \lambda|^2)^{1/2} \leq 2\mu|t,\alpha|$$

As $t = 1/c$, we finally have

$$|x - x^*| \leq M(|\lambda - x^*|^2/c^2 + |\alpha|^2)^{1/2}$$

and

$$|\lambda - x^*| \leq M(|\lambda - x^*|^2/c^2 + |\alpha|^2)^{1/2}$$

Q.E.D.

Lemma 1 stats the local convergence of general Lagrange multiplier method in inexact cases. Now we extend this to handle constraints explicitly. Namely, the augmented Lagrange function L only constrains part of the constraints. Other constraints are explicitly written as constraints.

Some other works handle the simple bound constraints. Here we propose a method that can handle general constraints. The idea is similar to [2].

This lemma draws the same idea of [3], Chapter 2, page 143, with the assumption that every step can be solved inexactly.

Lemma 2 Assume the standard assumption holds. Additionally, the gradient of each constraint at x^* are linearly independent.

Problem

$$minL_{1,c}(x, \lambda_1)$$

$$subjecttoh_2(x) = 0$$

is solved in an inexact manner. Namely, we have

$$\nabla L_{1,c}(x, \lambda_1) \leq \epsilon_1$$

$$|h_2(x)| \leq \epsilon_2$$

Then there exist M such that

$$|x - x^*| \leq M * max\left\{|\lambda - x^*|/c, |\epsilon_1|, |\epsilon_2|\right\}$$

and

$$|\lambda - x^*| \leq M * max\left\{|\lambda - x^*|/c, |\epsilon_1|, |\epsilon_2|\right\}$$

proof The proof is straightforward. The idea is to divide x as (x_1, x_2) and assume without loss of generality that ∇h_2 is non-singular. Then using the implicit function theorem, it's possible to solve near x^* the system of equations

$$h(x_1, x_2) = 0$$

and obtain x_2 in terms of x_1 as an implicit function $\Phi(x_1)$. To make it more preciously, we now consider a system of three equations as

$$\nabla f + \nabla h_1 \hat{\lambda} + \nabla h_2 \lambda_2 = 0,$$

$$h_1(x) + t + \gamma x_1^* - \gamma \tilde{\lambda} = \epsilon_1,$$

$$h_2(x) = \epsilon_2$$

Similar to the proof of Lemma 1. Now we have Jacobi matrix with respect to $(x, \lambda_1, \lambda_2)$ computed as (x^*, x^*, x_2^*) is

$$\begin{bmatrix} \nabla^2_{xx} L_0(x^*, x^*) & \nabla h_1(x^*) & h_2(x^*) \\ \nabla h_1(x^*)^T & -\gamma I & 0 \\ \nabla h_2(x^*)^T & 0 & 0 \end{bmatrix}$$

As our assumption ensures that this Jacobi matrix is non-singular for $\gamma = 0$. Actually for $\gamma \in [0, 1/\bar{c}]$, this matrix is non-singular. Therefore, by using the implicit function theorem mentioned in section 1.3 of the book, x_2 can denoted as a function of x_1 and h_2. The below is as the same as Lemma 1

6. Global Convergence. Lemma 1 and 2 states the local convergence result of our algorithm. However, a more important result is the global convergence proof, namely, every accumulation point of the sequence generated by this algorithm will converge to a feasible KKT point to the original problem under some mild assumptions, or the algorithm will stop at some step states that the original problem doesn't have a feasible solution. The global convergence result ensures the robustness of the algorithm.

The proof of this is similar to the proof appears in paper [1].

General assumptions:

The iterates $\{x^k\}$ considered lie within a closed, bounded domain Ω.

The MFCQ is hold (need to expand)

Lemma 3. Suppose that $1/c^k$ converges to 0 as k increases when Algorithm 1 is executed. Then λ/c convergences to 0.

Proof. We know that if $1/c$ converges to 0, step 3 (the penalty increase step) must be executed infinitely often. Therefore, a sub-sequence $\{k_i\}$of indies of the iteration can be picked where the step 3 is executed. And we know that, according to our updating rule,$c_k > 2^\beta$.

Now we consider how the Lagrange multiplier estimates change between two successive iterations indexed int eh set K.

$$\lambda^{k+j} = \lambda^k + \sum_{l=1}^{j-1} c^{k+l} h(x)$$

and

$$c_{k+1} = c_{k+1} = c_{k+j} = \eta c_k$$

Now that the summation is null if $j = 1$. Suppose that $j > 1$. We know that step 2 must executed and hence, we have the constraint violation that

$$h(x) < \epsilon_k$$

Plug it in to the equation above, we can get

$$\lambda^{k+j} \leq \lambda^k + \sum_{l=1}^{j-1} c^{k+l} h(x) \leq \lambda + 2\eta 1/c$$

Thus, we obtain that

$$\lambda^{k+j}/c \leq \eta \lambda^k + 2\eta 1/c$$

Therefore, if we let k increase, λ/c converges to zero.

Lemma 4. The violation of the constraints h is bounded by this equation

$$h(x^k \leq (\alpha_1 + \|\lambda - \lambda^*\|/c + \alpha_2 \|x^k - x^*\|/c$$

Proof: as we know that $h = 1/c(\lambda^{k+1} - \lambda^k)$. According to Lemma 2, we have the bound of $\lambda - \lambda^*$. Therefore, we have the bound on h.

Lemma 5. Suppose that all the assumption holds and at point x^* which is an accumulation point given by algorithm. Additionally, if we have $h(x^*) = 0$. Then, x^* is a KKT point of problem P. In the meanwhile, λ^* is the corresponding multiplier.

Note that x^* is an accumulative point. Therefore, there exist a sequence x^k with x^* as the limit point. Now we investigate the gradient of Lagrange function L at x^*. We all know that gradient of L approaches 0 when k approaches infinity according to the definition of our algorithm. The only thing we need to proof is the multiplier. Note that according to the assumption, the Jacobi matrix of the constraint h is nonsigular. Therefore, we define a new λ' which is $-J^+\nabla L$. We use the λ' to replace the λ^* in Lemma 2 without affect the result. In this case, as k approaches to infinity, λ converges to λ^*.

Now let's prove our global convergence result.

Theorem 1. Suppose all the assumption holds on every accumulative points x^* of sequences generated by our algorithm without the failure in step 2, then x^* is the KKT point of the original problem.

Proof There are two cases

1) If c is bounded. The Step 2 must be executed every iteration from k sufficiently large. But this implies that the constraint is always satisfied. Then according to Lemma 5, the accumulative point x^* is indeed the KKT point.

2) If c converges to infinity, we know that lemma 4 states that h is bounded by the equation. Therefore, h must be 0 when c approaches infinity. Therefore, the same conclusion can be drawn that x^* is the KKT point.

7. Algorithm Implementation. In this section, we briefly introduce our implementation details about our algorithm. We do believe that technical details should be well presented in introducing a novel algorithm framework. We briefly introduce the idea of constraint partitioning and

7.1. Strategies for partitioning constraints into subproblems. If we define R_{global} to measure the effectiveness of using different strategies for partitioning the constraints of a problem. Since the time to solve a partitioned problem is largely driven by the overhead in resolving its inconsistent global constraints, we define R_{global} to the the ratio of the number of global constraints to the total number of all constraints. Note that the metric is heuristic because the exact overhead depends on the difficulty of resolving the inconsistent global constraints and not on the number of global constraints. We can also introduce some weighted metric to the constraints, for example, we are willing to leave linear constraints as global constraints instead of nonlinear constraints as linear inconsistency tends to be easy to resolve. However, this is beyond the scope of this paper at this time.

As our goal is to partition the constraints in such a way that minimizes the overall search time, an efficient constraint partitioning algorithm is essential. Since the enumerations of all possible ways of partitioning is computationally prohibitive, our approach is to convert the constraint partitioning problem to a graph or hypergraph partitioning problem. In the graph model, we first formulate all the variables as the vertexes and the if two variables appear in a constraint, we add an edge in between these two variables (vertexes) with some weights, and then we minimize the total sum of the weights of the edges that cross different partitions. In the hypergraph model, we first formulate all the variables as the vertexes in the hyper-graph and all the constraints as hyper-edges in the hyper-graph. In this model, our goal is

to minimize the edge cuts cross different partitions, which is actually equivalent to minimize the number of global constraints, or R_{global}.

After studying the existing partition er in different fields and having tried different problem formulations graph, hyper-graph and community graph partitioning problem. We finally adopt the hyper-graph model and employ hMETIS to partition the hyper-graph based on varies of experiments. Note that the general graph or hyper-graph partitioning problem are NP-hard problems so the results of the partitioning algorithm might vary even with the same input due to some randomized heuristics.The observation here is that almost every existing graph/hyper-graph partitioner is fast enough (in comparing with the time to solve the whole NLP problem) so that we can try to partition the problem in several different ways and try to pick the optimal one. That is to say, although the one-time result may not reflect the average performance under the some partition parameter, we can still build an outer iteration that will repeat the partitioning procedure if the result is better than the known average as we are trying to find the optimal partitioning strategy.

7.2. Main Components. Our implementation contains three parts: Problem Analyzer (PA) is to analysis the problem structure and decide how to partition the problem. PA reads common modeling file like AMPL or GAMS file. PA is the solvers interface to the real world problems and provides a flexible interface to be hooked in the AMPL environment or other environments. In the implementation of the NL file analyzer, we noticed the work of Dr. AMPL, a meta solver for optimization which is able to find the approximate solution, as will as convexity analysis. Dr. AMPL is supposed to be an open source software distributed under the terms of the GNU Lesser General Public License. However, it is not yet available to the public at this time. Therefore, we follow the idea of Dr. AMPL and implement a simplified version of the Dr. AMPL which could help us set the initial points, analysis the problem structure and decide the number of partitions as well as the basic solver selection procedure. By working on these modules, we also get a byproduct: a very simple environment/API for problem analysis. It provides the problem structure plot, objective and constraint evaluation and other functionality where user can analysis the problem in just few commands or lines of code.

The Automated Partitioning Module (APM) is to partition the constraints in a way that minimizes the overall search time. In APM, there are several different components including a metric to measure the quality of partitioning, an algorithm to partition the constraints and optimize the metric as we mentioned above. Instead of simply use hMETIS, In our implementation, we employ several heuristics to handle arrowhead structure and decide the number of partitions.

The Basic Solver Interface reconstructs a series of new subproblems and employ other CNLP solvers to solver the subproblems after the partitioning. The purpose of the BSI is provide a neat yet powerful interface that can hook other solvers. Now the BSI implements a protocol to supply subproblems in AMPL nl file to the basic solver and retrieve the solution from existing solvers. As a general-purpose interface, the design and implementation of the interface is a challenge. We will discuss this in the later sections.

The Penalty Control Module (PCM) dynamically updates the penalty values in hope of resolving violated global constraints. The update rate of each penalty is proportional to the violation of the corresponding global constraints.

7.3. Reduce computational cost for each subproblem. As we've shown in this paper, for some large-scale problems, the subproblem only evolves a small number

of variables. While we can assign other variables as constants and pass the huge nl file to the subproblem solver, we can reduce the file size as well as the computational cost of the subproblem solver via a pre-processing called "pre-calculation". Actually, nl file consists of all the nonlinear expressions stored as the directed acyclic graph (DAG). Therefore, simply replace the non-local variables with constants does not reduce the computational cost as the subproblem solver still needs to evaluate the whole DAG every time in the iteration. However, after the assignment of the non-local variables to the constants, at some level of the DAG, the value of this node is fixed. For example, if we have $sin(x_1)$ where $x_1 = 1.0$, we do not need to evaluate $sin(1)$ every time as the value is fixed. Therefore, we "pre-calulate" the value of some nodes and reduce the whole DAG subtree to a single constant node, which significantly reduced the size of the nl file and the computational cost. In the elec problem from COPs benchmark, we finally reduce the size of the NL file to 1/9 of the original size and therefore reduce the total solution time to about 1/8.

8. Experimental results. In this section, we compare the performance of CPRE to that of other leading solvers. In the CNLP benchmarks, we use IPOPT, one of the leading solver in optimization. Our benchmarks includes problems from COPS, ASU benchmark and CUTE.

To get some insight into computational properties of our approach, we considered test problems taken from COPS. In particular, we choose Elec, Shape and Polygen.

8.1. Problem Elec. Given n_p electrons, find the equilibrium state distribution of the electrons positioned on a conducting sphere. If n_p points are denoted by (x_i, y_i, z_i), the problem can be formulated as: [5]

$$\min \sum_{i=1}^{n_p-1} \sum_{j=i+1}^{n_p} ((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)^{-\frac{1}{2}}, \tag{8.1}$$

$$\text{subject to} \quad x_i^2 + y_i^2 + z_i^2 = 1, \quad i = 1, \ldots, n_p \tag{8.2}$$

Obviously this problem is block separable where we can cluster constraints and variables along index i and it has many local minima. Theoretical results show that the number of local minima grows exponentially with n_p and determining the global minimum is computationally difficult. Thus, solvers are usually expected to find local minimum. We will show that partitioned algorithm can handle up to 3000 variables where other existing solver can only handle no more than 300 variables.

8.2. Problem CamShape. This problem tries to maximize the area of a convex cam with constraints on the curvature and on the radius of the cam. We assume that the shape of the cam is circular over an angle of $\frac{6}{5}\pi$ of its circumference with radius r_{min}. The design variables, r_i, $i = 1, \ldots, n$, represent the radius of the cam at equally spaces angles distributed over an angle of $\frac{2}{5n}\pi$.

The problem can be formulated as

$$\min \quad -\pi r_v^2 (\frac{1}{n} \sum_{i=1}^{n} r_i) \tag{8.3}$$

$$\text{s.t.} \quad 2r_{i-1}r_{i+1}cos(\theta) \leq r_i(r_{i-1} + r_{i+1}), i = 0, \ldots, n+1, \tag{8.4}$$

$$\text{s.t.} \quad -\alpha * \theta \leq r_{i+1} - r_i \leq \alpha * \theta, \quad i = 0, \ldots, n+1, \tag{8.5}$$

$$\tag{8.6}$$

The first constraint group are convexity constraints where $r_{-1} = r_0 = r_{min}$, $r_{n+1} = r_{max}$, $r_{n+2} = r_n$ and $\theta = 2\pi/5(n+1)$. The second group of constraints are curvature requirements. It's easy to show that these constraints are block separable if we remove two adjacent constraints from each constraint group. Namely, if we remove constraints in the form of $g_1(r_{k-1}, r_k, r_{k+1}) \leq 0$ and $g_2(r_k, r_{k+1}, r_{k+2}) \leq 0$, then the remaining constraints can be divided into two blocks with all constraints related to x_i, $i \leq k$ in first block and x_i, $i \geq k+1$ as the other block. Our algorithm handles g_1 and g_2 as global constraints and partition the original problem into subproblems.

TABLE 8.1
Results on MacMINLP

CUTE	CPRE-IPOPT		IPOPT	
ID	Sol.	Time	Sol.	Time
aug2dcqp	312	27	302	0.1
drcav1lq		1190.234		M
drcav2lq		1088.5		I
drcav3lq		662.1		1039.021
ncvxqp4		1.298		12.98
ncvxqp5		3.039		13.58
ncvxqp6		15.025		18.31
bratu3d		7.86		11.449
britgas		21.34		33.962
gausselm		35.921		51.131
nuffield2		21.129		23.033
nuffield		931.1		I
nuffield2_trap		18.9		47.055
semicon1		16.21		17.817
EIGMAXB	-	-	2.43	9.1
HADAMARD	-	-	1.21	27.21
KISSING	-	-	0.77	794.2
VANDERM1	-	-	0	19.2
VANDERM3	-	-	0	26.9
VANDERM4	-	-	0	24.16
OPPRLOC	550	0.11	550	0.01
OPTCNTRL	-16.42	2.12	-16.42	0.02

TABLE 8.2
Results on some COPS problems

COPS	IPOPT		CPRE	
ID	Sol.	Time	Sol.	Time
ELEC300	42145.09	27	42131.5	22
SHAPE1200	4.273	5.42	4.273	6.23
Polygon300	-	-	7.623492	891.2

REFERENCES

[1] Dimitri P. Bertsekas. *Constrainted Optimization and Lagrange Multiplier Methods*. Athena Scientific, 1982.

[2] Dimitri P. Bertsekas. *Nonlinear Programming: 2nd Edition*. Athena Scientific, 1999.
[3] I. Bongartz, A. R. Conn, Nick Gould, and Ph.L. Toint. CUTE: Constrained and unconstrained testing environment. *ACM Trans. on Mathematical Software*, 21(1):123–160, 1995.
[4] T. Davis. The university of florida sparse matrix collection. http://www.cise.ufl.edu/research/sparse/matrices.
[5] E. Dolan, J. J. Moré, and T. S. Munson. Benchmarking optimization software with COPS 3.0. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 2004.
[6] Michael C. Ferris and Olvi L. Mangasarian. Parallel variable distribution. *SIAM Journal on Optimization*, 4(4):815–832, 1994.
[7] O. L. Mangasarian. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Control and Optimization*, 33(6):1916–1925, 1995.
[8] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Springer, 2002.
[9] C. A. Sagastizábal and M. V. Solodov. Parallel variable distribution for constrained optimization. *Computational Optimization Applications*, 22(1):111–131, 2002.
[10] Paul Tseng and Dimitri P. Bertsekas. Relaxation methods for problems with strictly convex costs and linear constraints. *Math. Oper. Res.*, 16(3):462–481, 1991.
[11] A. Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, 2002.
[12] B. Wah and Y. X. Chen. Fast temporal planning using the theory of extended saddle points for mixed nonlinear optimization. *Artificial Intelligence*, accepted for publication 2005.

9. Conclusion. In this paper, we proved the global convergence of our augmented lagrange based algorithm.

REFERENCES

[1] Dimitri P. Bertsekas. *Constrainted Optimization and Lagrange Multiplier Methods*. Athena Scientific, 1982.
[2] Dimitri P. Bertsekas. *Nonlinear Programming: 2nd Edition*. Athena Scientific, 1999.
[3] I. Bongartz, A. R. Conn, Nick Gould, and Ph.L. Toint. CUTE: Constrained and unconstrained testing environment. *ACM Trans. on Mathematical Software*, 21(1):123–160, 1995.
[4] T. Davis. The university of florida sparse matrix collection. http://www.cise.ufl.edu/research/sparse/matrices.
[5] E. Dolan, J. J. Moré, and T. S. Munson. Benchmarking optimization software with COPS 3.0. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 2004.
[6] Michael C. Ferris and Olvi L. Mangasarian. Parallel variable distribution. *SIAM Journal on Optimization*, 4(4):815–832, 1994.
[7] O. L. Mangasarian. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Control and Optimization*, 33(6):1916–1925, 1995.
[8] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Springer, 2002.
[9] C. A. Sagastizábal and M. V. Solodov. Parallel variable distribution for constrained optimization. *Computational Optimization Applications*, 22(1):111–131, 2002.
[10] Paul Tseng and Dimitri P. Bertsekas. Relaxation methods for problems with strictly convex costs and linear constraints. *Math. Oper. Res.*, 16(3):462–481, 1991.
[11] A. Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, 2002.
[12] B. Wah and Y. X. Chen. Fast temporal planning using the theory of extended saddle points for mixed nonlinear optimization. *Artificial Intelligence*, accepted for publication 2005.