PARALLEL VARIABLE DISTRIBUTION FOR CONSTRAINED OPTIMIZATION*

C. A. Sagastizábal[†] and M. V. Solodov[‡]

November 24, 2000 (revised July 3, 2001)

ABSTRACT

In the parallel variable distribution framework for solving optimization problems (PVD), the variables are distributed among parallel processors with each processor having the primary responsibility for updating its block of variables while allowing the remaining "secondary" variables to change in a restricted fashion along some easily computable directions. For constrained nonlinear programs convergence theory for PVD algorithms was previously available only for the case of convex feasible set. Additionally, one either had to assume that constraints are block-separable, or to use exact projected gradient directions for the change of secondary variables. In this paper, we propose two new variants of PVD for the constrained case. Without assuming convexity of constraints, but assuming block-separable structure, we show that PVD subproblems can be solved inexactly by solving their quadratic programming approximations. This extends PVD to nonconvex (separable) feasible sets, and provides a constructive way of solving the parallel subproblems. For inseparable constraints, but assuming convexity, we develop a PVD method based on suitable approximate projected gradient directions. The approximation criterion is based on a certain error bound result, and it is readily implementable. Using such approximate directions may be especially useful when the projection operation is computationally expensive.

Key words. Parallel optimization, variable distribution, constrained optimization, sequential quadratic programming, projected gradient.
AMS subject classifications. 90C30, 49D27.

---

* Research of the first author is supported by CNPq and FAPERJ. The second author is supported in part by CNPq Grant 300734/95-6, by PRONEX–Optimization, and by FAPERJ.
[†] INRIA-Rocquencourt, BP 105, 78153 Le Chesnay, France.
Also Instituto de Matemática Pura e Aplicada, Estrada Dona Castorina 110, Jardim Botânico, Rio de Janeiro, RJ 22460-320, Brazil.
Email : `sagastiz@impa.br`
[‡] Instituto de Matemática Pura e Aplicada, Estrada Dona Castorina 110, Jardim Botânico, Rio de Janeiro, RJ 22460-320, Brazil.
Email : `solodov@impa.br`

1 Introduction and Motivation

We consider parallel algorithms for solving constrained optimization problems

$$\min_{x \in \mathcal{C}} f(x)\,, \tag{1.1}$$

where $\mathcal{C}$ is a nonempty closed set in $\Re^n$, and $f : \Re^n \to \Re$ is a continuously differentiable function. Our approach consists of partitioning the problem variables $x \in \Re^n$ into p blocks $x_1, \ldots, x_p$, such that

$$x = (x_1, \ldots, x_p)\,, \quad x_l \in \Re^{n_l}\,, l = 1, \ldots, p\,, \quad \sum_{l=1}^{p} n_l = n\,,$$

and distributing them among p parallel processors. Note that in this notation we shall not explicitly account for possible re-arranging of variables.

In most parallel algorithms, an iteration usually consists of two steps: parallelization and synchronization, [2]. Consider for the moment the unconstrained case, corresponding to $\mathcal{C} = \Re^n$. Typically, the synchronization step aims at guaranteeing a sufficient decrease in the objective function, while the parallel step produces candidate points (or candidate directions) by simultaneously solving certain subproblems (P_l) for $l = 1, \ldots, p$. Each (P_l) is defined on a subspace of dimension (significantly) smaller than n, in such a way that these p subspaces "span" the whole variable space $\Re^n$. Most methods, such as Block-Jacobi [2], updated conjugate subspaces [10], coordinate descent [21], and parallel gradient distribution [14], define (P_l) as a minimization problem on the l-th block of variables, i.e., in $\Re^{n_l}$. More recently, Parallel Variable Distribution (PVD) algorithms, introduced in [6] and further studied and extended in [19, 20, 7], advocated subproblems (P_l) of slightly higher dimensions than n_l. In the l-th subproblem, in addition to the associated "primary" optimization variables x_l, there are $p-1$ "secondary" variables representing in a *condensed form* all the remaining $n - n_l$ problem variables (see Algorithm 1.1 below). Since those remaining $n - n_l$ variables are allowed to change only in a restricted fashion (in a subspace of dimension $p-1$), the additional computational burden of solving such enlarged subproblems is not big. The idea is that the "forget-me-not" terms add an extra degree of freedom yielding algorithms with better robustness and faster convergence. We refer the reader to [6, 22, 12] for numerical validation of PVD-type methods. We note that not having any variables in parallel subproblems completely fixed can be especially important in the constrained case, i.e., when $\mathcal{C} \neq \Re^n$ in (1.1). Without this, the method can simply fail. We shall return to this issue later on.

1.1 General PVD Framework

To formalize the notion of primary and secondary variables, we need to introduce some notation. Let $\bar{l}$ denote the complement of l in the index set $\{1, \ldots, p\}$, where p is the number of parallel processors. Given some direction $d^i \in \Re^n$, which we shall call *PVD-direction*, we denote by $D^i_{\bar{l}}$ the $n_{\bar{l}} \times (p-1)$ block-diagonal matrix formed by placing the blocks $d^i_1, \ldots, d^i_{p-1}$

($d_t^i \in \Re^{n_t}$, $t = 1, \ldots, p-1$) of the chosen direction d^i along its block diagonal as follows:

$$D_{\bar{l}}^i := \begin{pmatrix} d_1^i & & & & & & \\ & d_2^i & & & & & \\ & & \ddots & & & & \\ & & & d_{l-1}^i & & & \\ & & & & d_{l+1}^i & & \\ & & & & & \ddots & \\ & & & & & & d_p^i \end{pmatrix}. \tag{1.2}$$

Note that if I denotes the identity matrix of appropriate dimension, the linear transformation

$$A_l^i := \begin{pmatrix} I_{n_l \times n_l} & 0_{n_l \times p-1} \\ 0_{n_{\bar{l}} \times n_l} & D_{\bar{l}}^i \end{pmatrix}$$

maps $(x_l, \mu_{\bar{l}}) \in \Re^{n_l} \times \Re^{p-1}$ into $(x_l, D_{\bar{l}}^i \mu_{\bar{l}}) \in \Re^n$. In the unconstrained case, more general transformations can be used, which give rise to a fairly broad *parallel variable transformation* framework discussed in [7]. However, the theory of [7] does not appear to extend to the constrained case, which is the focus of the present paper.

We describe next the basic PVD algorithm [6, 19, 20].

Algorithm 1.1 (PVD) *Start with any $x^0 \in \mathcal{C}$. Choose a PVD-direction $d^0 \in \Re^n$. Set $i := 0$.*
Having x^i, check a stopping criterion. If it is not satisfied, compute x^{i+1} as follows.
Parallelization. *For each processor $l \in \{1, \ldots, p\}$ compute a (possibly approximate) solution $(y_l^i, \mu_{\bar{l}}^i) \in \Re^{n_l} \times \Re^{p-1}$ of*

$$(P_l) \qquad \begin{cases} \min_{x_l, \mu_{\bar{l}}} \psi_l^i(x_l, \mu_{\bar{l}}) := f(x_l, x_{\bar{l}}^i + D_{\bar{l}}^i \mu_{\bar{l}}) \\ (x_l, x_{\bar{l}}^i + D_{\bar{l}}^i \mu_{\bar{l}}) \in \mathcal{C}\,. \end{cases}$$

Synchronization. *Compute $x^{i+1} \in \mathcal{C}$ such that*

$$f(x^{i+1}) \leq \min_{l \in \{1,\ldots,p\}} \psi_l^i(y_l^i, \mu_{\bar{l}}^i).$$

Set $i := i+1$; choose new PVD-direction d^i; and repeat.

∎

The idea of PVD algorithm is to balance the reduction in the number of variables for each (P_l) with allowing just enough freedom for the change of other (secondary) variables. Due to this, the parallel subproblems better approximate the original problem that has to be resolved. Note that because the secondary variables can change only along chosen fixed directions, their inclusion does not significantly increase the dimensionality of the parallel subproblems (P_l). Indeed, the number of variables in (P_l) is $n_l + p - 1$, only $p-1$ more than if the secondary variables were excluded. Of course, in the context of parallel computing it is reasonable to assume that p is small relative to n_l.

The synchronization step in Algorithm 1.1 may consist of minimizing the objective function in the affine hull, subject to feasibility, of all the points computed in parallel by the p processors. This would require solving a p-dimensional problem, which is again small compared to the original one. If (1.1) is a convex program, one can alternatively define x^{i+1} as a convex combination of the candidate points. In principle, for convergence purposes, any point with the objective function value at least as good as the smallest computed by all the processors is acceptable. As for PVD-directions, they are typically some easily computable feasible descent directions for the objective function f at the current iterate x^i, e.g., quasi-Newton or steepest descent directions in the unconstrained case.

Algorithm 1.1 is a rather general framework, which has to be further refined and specialized to obtain implementable/practical versions. In this respect, the two principal questions are:

- how to set up parallel subproblems; e.g., how to choose PVD-directions, and
- how to solve each parallel subproblem, including some criteria for inexact resolution.

When (1.1) is unconstrained, some of these issues have been addressed in [19], which contains improved convergence results (compared to [6]), including linear rate of convergence. These results, as well as useful generalizations such as algorithms with inexact subproblem solution and a certain degree of asynchronization, were obtained by imposing natural restrictions (of sufficient descent type) on the PVD-directions. An even more general framework for the unconstrained case was developed later in [7], where subproblems are obtained via certain nondegenerate transformations of the original variable space. These transformations can be very general, and there need not even be a distinction between primary and secondary variables. However, this approach does not seem to extend to the constrained case. It seems also that intuitively justifiable transformations do have some kind of primary-secondary variable structure. For those reasons, in the present paper we shall restrict our consideration to specific transformations of the form (1.2).

When (1.1) is a constrained optimization problem, many questions are still open, especially for a nonconvex feasible set $\mathcal{C}$. When $\mathcal{C}$ is convex with block-separable structure (i.e., $\mathcal{C}$ is a Cartesian product of closed convex sets), it was shown in [6] that every accumulation point of the PVD iterates satisfies the first-order necessary optimality conditions for problem (1.1). It was further stated that in the case of inseparable convex constraints, the PVD approach may fail. This conclusion was supported by a counter-example, which we reproduce below:

$$\begin{array}{ll} \min & x_1^2 + x_2^2 \\ \text{s.t.} & x_1 + x_2 - 2 \geq 0\,. \end{array}$$

This strongly convex quadratic program has a unique global solution at $\bar{x} = (1,1)$. Consider any point $\hat{x} \neq \bar{x}$) such that $\hat{x}_1 + \hat{x}_2 = 2\,, \hat{x}_1 \geq 0\,, \hat{x}_2 \geq 0$, and observe that

$$\hat{x}_1 = \arg\min_{x_1}\{x_1^2 + \hat{x}_2^2 \mid x_1 + \hat{x}_2 - 2 \geq 0\},$$
$$\hat{x}_2 = \arg\min_{x_2}\{\hat{x}_1^2 + x_2^2 \mid \hat{x}_1 + x_2 - 2 \geq 0\}.$$

Therefore, if we apply Algorithm 1.1 using $\hat{x}$ as its starting point and fixing the secondary variables, then this PVD variant will stay at this same nonoptimal point $\hat{x}$, thus failing to solve the original problem. This shows that in the constrained case, a special care should be taken in setting up the parallel subproblems.

For a general convex feasible set, it was shown in [20] that using the projected gradient direction $d(x) := x - P_{\mathcal{C}}[x - \nabla f(x)]$ for secondary variables does the job (here $P_{\mathcal{C}}[\cdot]$ stands for the orthogonal projection map onto the closed convex set $\mathcal{C}$). Specifically, it was established that setting $d^i := d(x^i)$ would ensure convergence of PVD methods for problems with general (inseparable) convex constraints. Some criteria for inexact subproblem solution were also given in [20]. When $\mathcal{C}$ is a polyhedral set, computing the projected gradient direction requires solving at every synchronization step of Algorithm 1.1 a single quadratic programming problem. For this, a wealth of fast and reliable algorithms is available [11, 5]. It should be noted that in the case of nonlinear constraints, the task of computing the projected gradient directions is considerably more computationally expensive. Actually, even in the affine case, computing those directions exactly (or very accurately) may turn to be rather wasteful, especially when far from the solution of the original problem. Therefore, improvements are necessary.

In this paper, we propose two new versions of PVD for the nonlinearly constrained case. The first one applies to problems with block-separable *nonconvex* feasible sets. It is based on the use of *sequential quadratic programming* techniques (SQP). Our second proposal is for *inseparable* convex feasible sets. We introduce a *computable* approximation criterion which allows to employ *inexact* projected gradient directions. This criterion is based on an error bound result, which is of independent interest. We emphasize that its is readily implementable and preserves global convergence of PVD methods based on exact directions.

Our notation is fairly standard. The usual inner product of two vectors $x, y \in \Re^n$ is denoted by $\langle x, y \rangle$, and the associated norm is given by $|x|^2 = \langle x, x \rangle$. When using other norms, we shall specify them explicitly. Analogous notation will be used for subspaces of any other dimensions; for example, the reduced subspaces $\Re^{n_l}$. For the sake of simplicity, we sometimes use a compact (transposed) notation when referring to composite vectors. For instance, $(x_l, \mu_{\bar{l}})$ stands for the column vector $\begin{pmatrix} x_l \\ \mu_{\bar{l}} \end{pmatrix}$. For a differentiable function $f : \Re^n \to \Re$, $\nabla f(x)$ will denote the n-dimensional column vector of partial derivatives of f at the point $x \in \Re^n$. For a differentiable vector-function $c : \Re^n \to \Re^m$, $c'(x)$ will denote the $m \times n$ Jacobian matrix whose rows are the transposed gradients $\nabla c_j(x)^\top, j = 1, \ldots, m$ of the components of c. For a function h we write $h \in C_L^{1,1}(X)$ if its partial derivatives are Lipschitz-continuous on the set X with modulus $L > 0$.

2 Nonconvex Separable Constraints

Suppose the feasible set $\mathcal{C}$ in (1.1) is described by a system of inequality constraints:

$$\mathcal{C} := \{x \in \Re^n \mid c(x) \le 0\}, \tag{2.1}$$

where $c : \Re^n \to \Re^m$. In this section, we assume that $\mathcal{C}$ has block-separable structure. Specifically,

$$c(x) = (c_1(x_1), \ldots, c_p(x_p)), \text{ where } c_l : \Re^{n_l} \to \Re^{m_l}\,,\ l = 1, \ldots, p\,,\ \sum_{l=1}^{p} m_l = m\,. \tag{2.2}$$

Our proposal is to solve parallel subproblems (P_l) of Algorithm 1.1 inexactly, by making one step of the sequential quadratic programming method (SQP) [3, Ch. 13]. Since SQP methods are local by nature, we modify the synchronization step in Algorithm 1.1 by introducing a suitable line-search based on the following exact penalty function (first employed for this purpose in [9]):

$$\theta_\sigma(x) := f(x) + \sigma \sum_{l=1}^{p} |c_l^+(x_l)|_1\,, \tag{2.3}$$

where σ is a positive parameter, and $y^+ := \max\{0, y\}$ with maximum taken componentwise. Given x^i, the l-th block of $\nabla f(x^i)$ will be denoted by $g_l^i := (I_{n_l \times n_l} \quad 0_{n_{\bar{l}} \times n_l}) \nabla f(x^i)$. Our algorithm is the following.

Algorithm 2.1 *Start with any $x^0 \in \mathcal{C}$. Choose parameters $\underline{\sigma} > 0$ and $\tau\,, \beta \in (0,1)$, and positive definite $n_l \times n_l$ matrices $M_l^0\,, l = 1, \ldots, p$. Set $i := 0$.*
Having x^i, check a stopping criterion. If it is not satisfied, compute x^{i+1} as follows.
Parallelization. *For each processor $l \in \{1, \ldots, p\}$ compute $(\delta_l^i, \lambda_l^i) \in \Re^{n_l} \times \Re^{m_l}$ as a KKT point of*

$$(QP_l) \qquad \left\{ \begin{array}{l} \min_{\delta_l \in \Re^{n_l}} \langle g_l^i, \delta_l \rangle + \frac{1}{2}\langle \delta_l, M_l^i \delta_l \rangle \\ c_l(x_l^i) + c_l'(x_l^i)\delta_l \le 0\,. \end{array} \right.$$

Synchronization. *Define*

$$\delta^i := (\delta_1^i, \ldots, \delta_p^i) \in \Re^n \quad \textit{and} \quad \lambda^i := (\lambda_1^i, \ldots, \lambda_p^i) \in \Re^m\,.$$

Line-search. *Choose $\sigma_i \ge |\lambda^i|_\infty + \underline{\sigma}$ and compute*

$$\Delta^i := \langle \nabla f(x^i), \delta^i \rangle - \sigma_i \sum_{l=1}^{p} |c_l^+(x_l^i)|_1\,.$$

Using the merit function (2.3), find m_i, the smallest nonnegative integer m, such that

$$\theta_{\sigma_i}(x^i + \beta^m \delta^i) \le \theta_{\sigma_i}(x^i) + \tau \beta^m \Delta^i\,. \tag{2.4}$$

Set $t_i := \beta^{m_i}$, $x^{i+1} := x^i + t_i\delta^i$, $i := i+1$. Choose new matrices M_l^i, $l = 1, \ldots, p$, and repeat.

■

The above proposal has to be compared to the original PVD algorithm [6]. Two remarks are in order. First, Algorithm 2.1 can be viewed as a PVD method with inexact solution of parallel subproblems. Indeed, "hard" nonlinear PVD subproblems (P_l) of Algorithm 1.1

are approximated here by "easy" quadratic programming subproblems (QP_l). And secondly, the PVD approach is extended to the case of nonconvex constraints. Note that the inclusion of forget-me-not terms does not seem to be crucial in the block-separable case (for example, such terms were not specified in the analysis of [6]). Thus, we drop here secondary variables in (P_l), by taking null PVD-directions. However, the use of secondary variables deserves further study for feasible sets with inseparable constraints. We also note that Algorithm 2.1 can be thought of as a distributed parallel implementation of SQP, where a block-diagonal matrix is chosen to generate quadratic approximations of the objective function. We remark that the contribution of Algorithm 2.1 is meant primarily to PVD framework rather than general SQP methods.

In Algorithm 2.1, we assume that subproblems (QP_l) have nonempty feasible sets for every iteration (this is guaranteed, for example, if for each l the Jacobian c_l' maps $\Re^{n_l}$ onto $\Re^{m_l}$, or if c_l is convex). Alternatively, it is known that feasibility of subproblems can be forced by introducing an extra "slack" variable [1, p. 377]. It is sometimes argued that SQP methods are not convenient for solving large-scale (with n and m large) nonlinear programs when there are inequality constraints present. More precisely, it is argued that the combinatorial aspect introduced by the complementarity condition in the KKT system associated to each QP makes the subproblems resolution very costly. On the other hand, SQP techniques are known to be very efficient for small to medium size problems, and they are often the method of choice in that setting. In relation to Algorithm 2.1, it is important to note that each subproblem (QP_l) is a quadratic programming problem of relatively small dimension (n_l and m_l are presumed to be small compared to n and m). There exist a number of very fast and reliable algorithms for solving such problems (see, e.g., [5]). In Section 4, we report some numerical results for Algorithm 2.1, obtained via simulation on a serial computer.

For further reference, we state the optimality conditions for (QP_l): the pair $(\delta_l^i, \lambda_l^i) \in \Re^{n_l} \times \Re^{m_l}$ solves the KKT system

$$
\begin{aligned}
g_l^i + M_l^i \delta_l^i + (c_l'(x_l^i))^\top \lambda_l^i &= 0, && (2.5a)\\
c_l(x_l^i) + c_l'(x_l^i)\delta_l^i &\le 0, && (2.5b)\\
\lambda_l^i \ge 0 \quad \text{and} \quad \langle \lambda_l^i, c_l(x_l^i) + c_l'(x_l^i)\delta_l^i \rangle &= 0. && (2.5c)
\end{aligned}
$$

Denoting by $\theta_\sigma'(x; d)$ the usual directional derivative of the merit function θ_σ at $x \in \Re^n$ in the direction $d \in \Re^n$, we next state that δ^i is a descent direction for this function at the point x^i. This in turn will imply that the line-search procedure in Algorithm 2.1 is well-defined.

Lemma 2.1 *If $f, c \in C_L^{1,1}(\Re^n)$, then*

$$
\theta_{\sigma_i}'(x^i; \delta^i) \le \Delta^i \le -\sum_{l=1}^{p} \langle \delta_l^i, M_l^i \delta_l^i \rangle - \underline{\sigma} \sum_{l=1}^{p} |c_l^+(x_l^i)|_1 , \tag{2.6}
$$

where x^i, δ^i, Δ^i, M_l^i are defined in Algorithm 2.1.

Proof. Defining the index-sets $I_+(x^i) := \{j \mid c_j(x^i) > 0\}$, $I_=(x^i) := \{j \mid c_j(x^i) = 0\}$, we have that (e.g., see [9, p. 301])

$$\theta'_{\sigma_i}(x^i;\delta^i) = \langle \nabla f(x^i), \delta^i\rangle + \sigma_i \sum_{j\in I_+} \langle \nabla c_j(x^i), \delta^i\rangle + \sigma_i \sum_{j\in I_=} (\langle \nabla c_j(x^i), \delta^i\rangle)^+ .$$

Using (2.5b) componentwise, we have that

$$\langle \nabla c_j(x^i), \delta^i\rangle \le -c_j(x^i) \begin{cases} < 0 & \text{if } j\in I_+(x^i) \\ = 0 & \text{if } j \in I_=(x^i) \end{cases} \Rightarrow \begin{array}{rcl} \langle \nabla c_j(x^i), \delta^i\rangle & \le & -|c_j^+(x^i)|\,,\ j\in I_+(x^i) \\ (\langle \nabla c_j(x^i), \delta^i\rangle)^+ & \le & -|c_j^+(x^i)|\,,\ j\in I_=(x^i)\,. \end{array}$$

Hence,

$$\theta'_{\sigma_i}(x^i;\delta^i) \le \langle \nabla f(x^i), \delta^i\rangle - \sigma_i \sum_{l=1}^p |c_l^+(x_l^i)|_1 = \Delta^i\,. \tag{2.7}$$

To obtain the right-most inequality in (2.6) we proceed as follows:

$$\begin{array}{rcll} \langle \nabla f(x^i), \delta^i\rangle & = & \displaystyle\sum_{l=1}^p \langle g_l^i, \delta_l^i\rangle & \\ & = & -\displaystyle\sum_{l=1}^p \langle M_l^i\delta_l^i + (c_l'(x_l^i))^\top \lambda_l^i, \delta_l^i\rangle & \text{[by (2.5a)]} \\ & = & -\displaystyle\sum_{l=1}^p \langle M_l^i\delta_l^i, \delta_l^i\rangle - \sum_{l=1}^p \langle \lambda_l^i, c_l'(x_l^i)\delta_l^i\rangle & \\ & = & -\displaystyle\sum_{l=1}^p \langle M_l^i\delta_l^i, \delta_l^i\rangle + \sum_{l=1}^p \langle \lambda_l^i, c_l(x_l^i)\rangle & \text{[by (2.5c)]} \\ & \le & -\displaystyle\sum_{l=1}^p \langle M_l^i\delta_l^i, \delta_l^i\rangle + \sum_{l=1}^p \langle \lambda_l^i, c_l^+(x_l^i)\rangle & \text{[by (2.5c) and } c_l^+(x_l^i)\ge c_l(x_l^i)]\,. \end{array}$$

Combining the latter relation with (2.7), we have that

$$\begin{array}{rcll} \Delta^i & \le & -\displaystyle\sum_{l=1}^p \langle M_l^i\delta_l^i, \delta_l^i\rangle + \sum_{l=1}^p \langle \lambda_l^i, c_l^+(x_l^i)\rangle - \sigma_i\sum_{l=1}^p |c_l^+(x_l^i)|_1 & \\ & \le & -\displaystyle\sum_{l=1}^p \langle M_l^i\delta_l^i, \delta_l^i\rangle + \sum_{l=1}^p (|\lambda_l^i|_\infty - \sigma_i)|c_l^+(x_l^i)|_1 & \text{[by Cauchy-Schwarz inequality]} \\ & \le & -\displaystyle\sum_{l=1}^p \langle M_l^i\delta_l^i, \delta_l^i\rangle - \underline{\sigma}\sum_{l=1}^p |c_l^+(x_l^i)|_1\,, & \text{[by the choice of } \sigma_i] \end{array}$$

which completes the proof. ∎

For Algorithm 2.1 to be globally convergent, we assume that the penalization parameter σ_i is kept bounded above, which essentially means that the multipliers λ_l^i stay bounded. From the practical point of view, the latter assumption is natural. In what follows, we show convergence of Algorithm 2.1 to Karush-Kuhn-Tucker (KKT) points of (1.1), i.e., pairs $(\bar{x}, \bar{\lambda}) \in \Re^n \times \Re^m$ satisfying

$$\begin{array}{rcl} \nabla f(\bar{x}) + c'(\bar{x})^\top \bar{\lambda} & = & 0\,, \\ c(\bar{x}) & \le & 0\,, \\ \bar{\lambda}\ge 0 \quad \text{and} \quad \langle \bar{\lambda}, c(\bar{x})\rangle & = & 0\,. \end{array} \tag{2.8}$$

Theorem 2.1 *Suppose that $f, c \in C_L^{1,1}(\Re^n)$, and let the feasible set $\mathcal{C}$ have block-separable structure as in* (2.2). *Let $\{(x^i, \lambda^i)\}$ be a sequence generated by Algorithm 2.1. Assume there exists an iteration index i_0 such that $\sigma_i = \bar{\sigma}$ for all $i \geq i_0$ and the matrices M_l^i are uniformly positive definite and bounded for all $l = 1, \dots, p$ and all iteration indices i. Then either $\theta_{\sigma_i}(x^i) \to -\infty$, or every accumulation point $(\bar{x}, \bar{\lambda})$ of the sequence $\{(x^i, \lambda^i)\}$ is a KKT point of the problem, i.e., it satisfies* (2.8).

Proof. By (2.6) in Lemma 2.1, δ^i is a direction of descent for the merit function θ_{σ_i} at x^i, unless $\delta_l^i = 0$ and $c_l^+(x_l^i) = 0$ for all $l = 1, \dots, p$. In the former case, by standard argument the line-search procedure is well-defined and terminates finitely with some stepsize $t_i > 0$. In the latter case, (2.5a)-(2.5c) reduce to

$$\begin{aligned} g_l^i + M_l^i \delta_l^i + (c_l'(x_l^i))^\top \lambda_l^i &= 0\,, \\ c_l(x_l^i) &\leq 0\,, \\ \lambda_l^i \geq 0 \quad \text{and} \quad \langle \lambda_l^i, c_l(x_l^i) + c_l'(x_l^i)\delta_l^i \rangle &= 0\,. \end{aligned}$$

Now taking into account separability of constraints, it follows that (x^i, λ^i) satisfies the KKT system (2.8).

Suppose that the method does not terminate and generates an infinite sequence of iterates. We prove first that the sequence of stepsizes $\{t_i\}$ is bounded away from 0. Take any $t \in (0, 1]$. Since $f \in C_L^{1,1}(\Re^n)$, by [1, Proposition A.24] we have that

$$f(x^i + t\delta^i) \leq f(x^i) + t\langle \nabla f(x^i), \delta^i \rangle + \frac{Lt^2}{2}|\delta^i|^2\,. \tag{2.9}$$

Similarly, since $c \in C_L^{1,1}(\Re^n)$, using the equivalence of the norms in the finite-dimensional setting, for some $R_1 > 0$ we have that

$$\begin{aligned} R_1 t^2 |\delta_l^i|^2 &\geq |c_l(x_l^i + t\delta_l^i) - c_l(x_l^i) - t c_l'(x_l^i)\delta_l^i|_1 \\ &= |c_l(x_l^i + t\delta_l^i) - t(c_l(x_l^i) + c_l'(x_l^i)\delta_l^i) - (1-t)c_l(x_l^i)|_1 && [\text{add } \pm t c_l(x_l^i)] \\ &\geq |\Big(c_l(x_l^i + t\delta_l^i) - t(c_l(x_l^i) + c_l'(x_l^i)\delta_l^i) - (1-t)c_l(x_l^i)\Big)^+|_1 && [\text{use } |a|_1 \geq |a^+|_1] \\ &\geq |c_l^+(x_l^i + t\delta_l^i)|_1 - |\Big(t(c_l(x_l^i) + c_l'(x_l^i)\delta_l^i) + (1-t)c_l(x_l^i)\Big)^+|_1\,. && [\text{use } |(a-b)^+|_1 \geq |a^+|_1 - |b^+|_1] \end{aligned}$$

Re-writing the above relation, we obtain

$$\begin{aligned} |c_l^+(x_l^i + t\delta_l^i)|_1 &\leq |\Big(t(c_l(x_l^i) + c_l'(x_l^i)\delta_l^i) + (1-t)c_l(x_l^i)\Big)^+|_1 + R_1 t^2 |\delta_l^i|^2 \\ &\leq t|\Big(c_l(x_l^i) + c_l'(x_l^i)\delta_l^i\Big)^+|_1 + (1-t)|c_l^+(x_l^i)|_1 + R_1 t^2 |\delta_l^i|^2 && [\text{by convexity of } |(\cdot)^+|_1] \\ &= (1-t)|c_l^+(x_l^i)|_1 + R_1 t^2 |\delta_l^i|^2\,. && [\text{by (2.5b)}] \end{aligned}$$

Together with (2.9), the last inequality yields

$$\begin{aligned} \theta_{\sigma_i}(x^i + t\delta^i) &= f(x^i + t\delta^i) + \sigma_i \sum_{l=1}^p |c_l^+(x_l^i + t\delta_l^i)|_1 \\ &\leq f(x^i) + t\langle \nabla f(x^i), \delta^i \rangle + \sigma_i(1-t) \sum_{l=1}^p |c_l^+(x_l^i)|_1 + R_2 t^2 |\delta^i|^2 \\ &= \theta_{\sigma_i}(x^i) + t\Delta^i + R_2 t^2 |\delta^i|^2\,, \end{aligned}$$

where $R_2 > 0$ is a fixed constant depending on $R_1, L, \bar{\sigma}$. By a direct comparison of the latter relation with (2.4), we conclude that (2.4) is guaranteed to be satisfied once m is large enough so that $t = \beta^m$ falls within the set of t satisfying $t\Delta^i + R_2 t^2 |\delta^i|^2 \leq \tau t \Delta^i$. In particular, since the line-search procedure did not accept the stepsize $t = \beta^{m_i - 1}$, it follows that

$$\text{either } m_i = 0 \text{ or } t = \beta^{m_i - 1} > \frac{(\tau - 1)\Delta^i}{R_2 |\delta^i|^2} . \tag{2.10}$$

By (2.6) in Lemma 2.1 and the uniform positive definiteness of matrices M_l^i, there exists $R_3 > 0$ such that

$$-\Delta^i \geq \sum_{l=1}^{p} \langle M_l^i \delta_l^i, \delta_l^i \rangle \geq \sum_{l=1}^{p} R_3 |\delta_l^i|^2 = R_3 |\delta^i|^2 ,$$

and using (2.10), we conclude that

$$t_i = \beta^{m_i} \geq \bar{t} := \min\{1 ; \beta(1 - \tau) R_3 / R_2\} > 0 .$$

By the assumption that $\sigma_i = \bar{\sigma}$ for all $i \geq i_0$, from (2.4) it follows that the sequence $\{\theta_{\bar{\sigma}}(x^i)\}$ is nonincreasing. Hence, it is either unbounded below, or it converges. In the latter case, (2.4) implies that $\tau t_i \Delta^i \to 0$, and since $t_i \geq \bar{t} > 0$, it holds that

$$\Delta^i \to 0 .$$

By the definition of Δ^i and the uniform positive definiteness of matrices M_l^i, we conclude that

$$\delta_l^i \to 0 , \quad c_l^+(x_l^i) \to 0 , \quad l = 1, \ldots, p .$$

Now, passing onto the limit in (2.5a)-(2.5c) as $i \to \infty$, and taking into account the boundedness of the matrices M_l^i, we obtain the assertions of the theorem. ■

3 Convex Inseparable Constraints

Suppose now that the feasible set $\mathcal{C}$ is defined by a system of convex inequalities, i.e., $c : \Re^n \to \Re^m$ in (2.1) has convex components $c_j(\cdot), j = 1, \ldots, m$. We emphasize that in this section we do not assume separability of constraints. In this setting, it appears that the only currently known way to ensure convergence of the PVD algorithm is to use the projected gradient directions for the change of secondary variables [20]. Omitting the iteration indices i, if x is the current iterate, to compute these directions one has to solve a subproblem of the following structure:

$$\min_{c(z) \leq 0} \frac{1}{2} |z - (x - \nabla f(x))|^2 . \tag{3.1}$$

This is done by some iterative algorithm. As already discussed in Section 1, solving this problem in the general nonlinear case can be quite costly. Moreover, when far from a solution of the original problem, exact (or even very accurate) projection directions are perhaps unnecessary. This suggests developing a stopping rule for solving (3.1) or, equivalently, an approximation criterion for projection directions to be used in the PVD scheme. For algorithmic purposes, it is important to make this criterion constructive and implementable.

Assuming some constraint qualification condition [13], we have that $\bar{z}$ solves (3.1), i.e.,

$$\bar{z} = P_{\mathcal{C}}[x - \nabla f(x)]$$

if, and only if, the pair $(\bar{z}, \bar{u}) \in \Re^n \times \Re^m$ satisfies the KKT system

$$\begin{aligned} \nabla_z L(\bar{z}, \bar{u}) = \bar{z} - x + \nabla f(x) + c'(\bar{z})^\top \bar{u} &= 0\,, \\ c(\bar{z}) &\leq 0\,, \\ \bar{u} \geq 0 \quad \text{and} \quad \langle \bar{u}, c(\bar{z}) \rangle &= 0\,, \end{aligned} \tag{3.2}$$

where

$$L(z, u) = \frac{1}{2}|z - (x - \nabla f(x))|^2 + \langle u, c(z) \rangle \tag{3.3}$$

is the standard Lagrangian for problem (3.1).

Suppose $z \in \mathcal{C}$ and $u \in \Re^m_+$ is some current approximation to a primal-dual optimal solution of (3.1), generated by an iterative algorithm applied to solve this problem. Lemma 3.1 below establishes an *error bound* for the distance from z to $P_{\mathcal{C}}[x - \nabla f(x)]$ in terms of violations of the KKT conditions (3.2) by the pair (z, u). A nice feature of this estimate is that unlike some (perhaps, most) error bounds results in the literature (see [18] for a survey), it does not involve any expressions which are not readily computable or any other quantities which are not *observable*. Furthermore, this error bound holds globally, i.e., not just in some neighbourhood of the solution point. Thus it can be easily employed for algorithmic purposes.

Lemma 3.1 is related to error bounds for strongly convex programs obtained in [15]. However, Theorem 2.2 in [15] involves certain constants which are in general not computable, while Corollary 2.4 in [15] assumes not only that z is primal feasible, but also that (z, u) is *dual* feasible, which here means that $u \geq 0\,, \nabla_z L(z, u) = 0$. In Lemma 3.1 we only assume that z is primal feasible and that the approximate multiplier u is nonnegative. Our assumptions are not only weaker but they also appear to be more suitable in an algorithmic framework, as they will be satisfied at each iteration of many standard optimization methods. Dual feasibility, in contrast, is unlikely to be satisfied along iterations of typical algorithms, except in the limit.

Lemma 3.1 *Let $c : \Re^n \to \Re^m$ be convex and differentiable, and suppose that the set $\mathcal{C}$ given by (2.1) satisfies some constraint qualification.*

Then for any $z \in \mathcal{C}$ and $u \in \Re^m_+$, it holds that

$$|z - P_{\mathcal{C}}[x - \nabla f(x)]| \leq \varepsilon(z, u),$$

where

$$\varepsilon(z, u) := \frac{1}{2}|\nabla_z L(z, u)| + \frac{1}{2}\sqrt{|\nabla_z L(z, u)|^2 - 4\langle u, c(z) \rangle}. \tag{3.4}$$

Proof. Let $\bar{z} = P_{\mathcal{C}}[x - \nabla f(x)]$ and $\bar{u}$ be the associated multiplier, so that the pair $(\bar{z}, \bar{u})$ satisfies (3.2). Denote

$$\gamma := \langle \nabla_z L(z, u) - \nabla_z L(\bar{z}, \bar{u}), z - \bar{z} \rangle - \langle c(z) - c(\bar{z}), u - \bar{u} \rangle\,,$$

where $L(\cdot,\cdot)$ is defined by (3.3). Take any $z \in \mathcal{C}$ and $u \in \Re^m_+$. Denoting further $y = x - \nabla f(x)$, we have that

$$\begin{array}{rcl} \gamma & = & \langle z - y + c'(z)^\top u - (\bar{z} - y + c'(\bar{z})^\top \bar{u}), z - \bar{z}\rangle - \langle c(z) - c(\bar{z}), u - \bar{u}\rangle \\ & = & |z - \bar{z}|^2 + \langle u, c(\bar{z}) - c(z) - c'(z)(\bar{z} - z)\rangle + \langle \bar{u}, c(z) - c(\bar{z}) - c'(\bar{z})(z - \bar{z})\rangle \\ & \geq & |z - \bar{z}|^2 , \end{array} \tag{3.5}$$

where the inequality follows from $u \geq 0$, $\bar{u} \geq 0$ and the facts that, by the convexity of $c(\cdot)$, $c(\bar{z}) - c(z) - c'(z)(\bar{z} - z) \geq 0$ and $c(z) - c(\bar{z}) - c'(\bar{z})(z - \bar{z}) \geq 0$.

On the other hand,

$$\begin{array}{rcl} \gamma & = & \langle \nabla_z L(z,u), z - \bar{z}\rangle - \langle c(z), u\rangle + \langle \bar{u}, c(z)\rangle + \langle u, c(\bar{z})\rangle \\ & \leq & |\nabla_z L(z,u)||z - \bar{z}| - \langle u, c(z)\rangle , \end{array} \tag{3.6}$$

where the equality follows from the KKT conditions (3.2), and the inequality follows from the facts that $\bar{u} \geq 0$, $u \geq 0$ and $c(z) \leq 0$, $c(\bar{z}) \leq 0$, and the Cauchy-Schwarz inequality. Denoting $t := |z - \bar{z}|\,, \alpha := |\nabla_z L(z,u)| \geq 0$ and $\beta := -\langle u, c(z)\rangle \geq 0$, and combining (3.5) with (3.6), we obtain the following quadratic inequality in t:

$$t^2 - \alpha t - \beta \leq 0\,.$$

Resolving this inequality, we obtain that

$$t \leq \frac{1}{2}(\alpha + \sqrt{\alpha^2 + 4\beta})\,.$$

Recalling definitions of the quantities involved, we conclude that

$$|z - \bar{z}| \leq \frac{1}{2}|\nabla_z L(z,u)| + \frac{1}{2}\sqrt{|\nabla_z L(z,u)|^2 - 4\langle u, c(z)\rangle} = \varepsilon(z,u)\,.$$

■

We propose the following PVD algorithm based approximations of the projected gradient directions.

Algorithm 3.1 *Choose parameters* $\sigma_1 \in (0,1)$ *and* $\sigma_2 \in (0, (1 - \sigma_1)^2)$.
Start with any $x^0 \in \mathcal{C}$. *Set* $i := 0$.
Having x^i, *check a stopping criterion. If it is not satisfied, proceed as follows.*
PVD-direction choice. *Compute* $z^i \approx P_{\mathcal{C}}[x^i - \nabla f(x^i)]$ *such that* $z^i \in \mathcal{C}$ *and the associated approximate Lagrange multiplier* $u^i \in \Re^m_+$ *for problem (3.1) satisfy*

$$\varepsilon(z^i, u^i) \leq \min\left\{\sigma_1\,, \frac{\sigma_2|z^i - x^i|}{|\nabla f(x^i)|}\right\}|z^i - x^i|, \tag{3.7}$$

where $\varepsilon(z^i, u^i)$ *is given by (3.4). Set* $d^i := x^i - z^i$.
Compute x^{i+1} *as follows.*

Parallelization. *For each processor* $l \in \{1, \dots, p\}$ *compute a solution* $(y_l^i, \mu_{\bar{l}}^i) \in \Re^{n_l} \times \Re^{p-1}$ *of* (P_l) *as defined in Algorithm 1.1.*
Synchronization. *Compute* $x^{i+1} \in \mathcal{C}$ *such that*

$$f(x^{i+1}) \le \min_{l \in \{1, \dots, p\}} \psi_l^i(y_l^i, \mu_{\bar{l}}^i).$$

Set $i := i + 1$*; and repeat.*

■

Note that Algorithm 3.1 is a general-purpose method for problems with no special structure. The example presented in the Introduction shows that for such problems computing a meaningful direction for secondary variables is indispensable. Compared to *any* alternative, computing an approximate projected gradient direction seems to be quite favorable in terms of cost, and perhaps even the best one can do. Regarding the tolerance criterion (3.7), we note that if z^i is the exact projection point then $\varepsilon(z^i, u^i) = 0$, while $z^i - x^i \ne 0$ (if $z^i = x^i$ then x^i satisfies the first-order necessary optimality condition $x^i = P_{\mathcal{C}}[x^i - \nabla f(x^i)]$). From this, it is easy to see that the projection problem (3.1) always has inexact solutions satisfying (3.7). Hence, the method is well-defined.

As for the convergence properties of Algorithm 3.1, our result is the following.

Theorem 3.1 *Let* $c : \Re^n \to \Re^m$ *be convex and differentiable, and* $f \in C_L^{1,1}(\mathcal{C})$*. Suppose* $\{x^i\}$ *is a sequence generated by Algorithm 3.1. Then either* f *is unbounded from below on* $\mathcal{C}$*, or the sequence* $\{f(x^i)\}$ *converges, and every accumulation point of the sequence* $\{x^i\}$ *satisfies the first-order necessary optimality condition.*

Proof. Consider any iteration $i = 0, 1, \dots$, any processor $l \in \{1, \dots, p\}$, and the point

$$(x_l^i - \alpha d_l^i, -\alpha e_{\bar{l}}) \in \Re^{n_l + p - 1}, \quad 0 < \alpha < \min\left\{1, \frac{2}{L}((1 - \sigma_1)^2 - \sigma_2)\right\},$$

where $e_{\bar{l}} = (1, \dots, 1) \in \Re^{p-1}$. We first show that this point is feasible for the corresponding subproblem (P_l) of minimizing the function $\psi_l^i(x_l, \mu_{\bar{l}}) := f(x_l, x_{\bar{l}}^i + D_{\bar{l}}^i \mu_{\bar{l}})$. Indeed, using the notation (1.2),

$$\begin{aligned} (x_l^i - \alpha d_l^i, x_{\bar{l}}^i - \alpha D_{\bar{l}}^i e_{\bar{l}}) &= (x_l^i - \alpha d_l^i, x_{\bar{l}}^i - \alpha d_{\bar{l}}^i) \\ &= x^i - \alpha d^i \\ &= (1 - \alpha) x^i + \alpha z^i \in C, \end{aligned}$$

where the first equality follows from the structure of $D_{\bar{l}}^i$, and the inclusion is by the facts that $x^i, z^i \in \mathcal{C}$, $\alpha \in (0, 1)$, and the convexity of the set $\mathcal{C}$. As a result, we have that

$$\begin{aligned} f(x^i) - f(y_l^i, x_{\bar{l}}^i + D_{\bar{l}}^i \mu_{\bar{l}}^i) &= \psi_l^i(x_l^i, 0) - \psi_l^i(y_l^i, \mu_{\bar{l}}^i) \\ &\ge \psi_l^i(x_l^i, 0) - \psi_l^i(x_l^i - \alpha d_l^i, -\alpha e_{\bar{l}}) \\ &= f(x^i) - f(x_l^i - \alpha d_l^i, x_{\bar{l}}^i - \alpha D_{\bar{l}}^i e_{\bar{l}}) \\ &= f(x^i) - f(x^i - \alpha d^i) \\ &\ge \alpha \langle \nabla f(x^i), d^i \rangle - \frac{L}{2} \alpha^2 |d^i|^2, \end{aligned} \tag{3.8}$$

where the last inequality follows from $f \in C_L^{1,1}(\mathcal{C})$ (e.g, [1, Proposition A.24]).

By the construction of the algorithm and Lemma 3.1,

$$|z^i - P_C[x^i - \nabla f(x^i)]| \le \varepsilon(z^i, u^i)\,.$$

Hence, there exists some $\xi^i \in \Re^n$ such that

$$z^i + \xi^i = P_C[x^i - \nabla f(x^i)]\,, \quad |\xi^i| \le \varepsilon(z^i, u^i)\,. \tag{3.9}$$

Define the (continuous) residual function

$$R(x) := x - P_C[x - \nabla f(x)]\,.$$

With this notation,

$$x^i - z^i - \xi^i = d^i - \xi^i = R(x^i)\,.$$

By properties of the projection operator (e.g., see [1, Proposition B.11]), since $x^i \in \mathcal{C}$ we have that

$$\begin{aligned} 0 &\ge \langle x^i - \nabla f(x^i) - P_C[x^i - \nabla f(x^i)], x^i - P_C[x^i - \nabla f(x^i)]\rangle \\ &= \langle x^i - \nabla f(x^i) - z^i - \xi^i, x^i - z^i - \xi^i\rangle \\ &= -\langle \nabla f(x^i), d^i\rangle + \langle \nabla f(x^i), \xi^i\rangle + |d^i - \xi^i|^2\,. \end{aligned}$$

Hence,

$$\langle \nabla f(x^i), d^i\rangle \ge |R(x^i)|^2 - |\nabla f(x^i)||\xi^i|\,.$$

Combining the latter relation with (3.8) and (3.9), and taking into account the synchronization step of Algorithm 3.1, we further obtain

$$f(x^i) - f(x^{i+1}) \ge \alpha(|R(x^i)|^2 - \varepsilon(z^i, u^i)|\nabla f(x^i)| - \frac{L}{2}\alpha|d^i|^2)\,. \tag{3.10}$$

Observe that

$$|d^i| \le |R(x^i)| + |\xi^i| \le |R(x^i)| + \varepsilon(z^i, u^i) \le |R(x^i)| + \sigma_1|d^i|\,,$$

where the first relation is by the Cauchy-Schwarz inequality, the second follows from (3.9), and the last from (3.7). Hence,

$$|R(x^i)| \ge (1 - \sigma_1)|d^i|\,.$$

Using (3.10), the latter inequality, and again (3.7), we have that

$$f(x^i) - f(x^{i+1}) \ge \alpha\left((1 - \sigma_1)^2 - \sigma_2 - \frac{L\alpha}{2}\right)|d^i|^2\,. \tag{3.11}$$

By the choice of α and assumptions on parameters σ_1 and σ_2, the sequence $\{f(x^i)\}$ is nonincreasing. If $f(\cdot)$ is bounded below on $\mathcal{C}$, then $\{f(x^i)\}$ is bounded below, and hence it converges. In the latter case, $\{f(x^i) - f(x^{i+1})\} \to 0$ and therefore $\{d^i\} \to 0$, by (3.11). On the other hand,

$$|d^i| \geq |R(x^i)| - |\xi^i| \geq |R(x^i)| - \varepsilon(z^i, u^i) \geq |R(x^i)| - \sigma_1 |d^i|,$$

so that

$$|R(x^i)| \leq (1 + \sigma_1)|d^i|.$$

We conclude that $\{R(x^i)\}$ also tends to zero. By the continuity of $R(\cdot)$, it then follows that for every accumulation point $\bar{x}$ of the sequence $\{x^i\}$, $R(\bar{x}) = 0$. It is well known that the latter is equivalent to the the minimum principle necessary optimality condition $\bar{x} \in \mathcal{C}$, $\langle \nabla f(\bar{x}), x - \bar{x} \rangle \geq 0$ for all $x \in \mathcal{C}$. ■

4 Preliminary Numerical Experience

To get some insight into computational properties of our approach in § 2, we considered test problems taken from the study of Sphere Packing Problems [4]. In particular, the problems chosen are the same as the ones used in [16]. We note that Problems 2 and 3 below are considered quite challenging for nonlinear programming algorithms, especially if one is seeking a relatively accurate solution. Not having access to a parallel computer, we have carried out a simulation, i.e., the subproblems are solved serially on a serial machine. Even though this is admittedly a rather crude experiment, it nevertheless gives some idea on what one might expect in actual parallel implementation.

Given p spheres in $\Re^\nu$ with $\nu, p \in \{1, 2, \ldots\}$, so that $x \in \Re^{\nu p}$, the problems are as follows.

Problem 1.

$$\min \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} \sum_{t=1}^{\nu} x_{(i-1)\nu+t} x_{(j-1)\nu+t}$$

$$\text{s.t.} \ \sum_{t=1}^{\nu} x_{(i-1)\nu+t}^2 - 1 = 0, \ i = 1, \ldots, p.$$

Problem 2. Given an integer $\mu \geq 1$,

$$\min \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} \frac{1}{(\sum_{t=1}^{\nu} (x_{(i-1)\nu+t} - x_{(j-1)\nu+t})^2 + 1)^\mu}$$

$$\text{s.t.} \ \sum_{t=1}^{\nu} x_{(i-1)\nu+t}^2 - 1 = 0, \ i = 1, \ldots, p.$$

Problem 3.

$$\min \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} \left(\left(\sum_{t=1}^{\nu} (x_{(i-1)\nu+t} - x_{(j-1)\nu+t})^2 \right)^{-6} - 2 \left(\sum_{t=1}^{\nu} (x_{(i-1)\nu+t} - x_{(j-1)\nu+t})^2 \right)^{-3} \right)$$

$$\text{s.t.} \ \sum_{t=1}^{\nu} x_{(i-1)\nu+t}^2 - 1 = 0, \ i = 1, \ldots, p.$$

We have implemented in Matlab our Algorithm 2.1, the serial SQP method, and the general PVD Algorithm 1.1. All codes were run under Matlab version 6.0.0.88 (Release 12) on a Sun UltraSPARCstation. Details of the implementation are as follows. Algorithm 1.1 is implemented using the function `constr.m` from the Matlab Optimization Toolbox for solving the subproblems (P_l). Algorithm 2.1 and the serial SQP method are quite sophisticated quasi-Newton implementations with line-search based on the merit function (2.3). Each (QP_l) is solved by the Null-Space Method [8]. The penalty parameter σ_i is updated using a modification of the Mayne and Polak rule [17] that allows σ_i to decrease, if warranted. The stepsize t_i is computed with an Armijo rule that uses safeguarded quadratic interpolation. In order to prevent the Maratos effect (i.e., to ensure that the unit stepsize is asymptotically accepted), a second-order correction is added to the search direction δ^i when necessary [3, Ch. 13.4]. Finally, the quasi-Newton matrices are updated by the BFGS formula, using also the Powell correction and the Oren-Luenberger scaling. The methods stop when the relative error, measured by the sum of the norm of the reduced gradient and the norm of the constraints, is less than 10^{-5}.

We first compare our Algorithm 2.1 with the general PVD Algorithm 1.1. Since this comparison turned out rather obvious and one-sided (which is certainly not surprising), we do not report exhaustive testing for this part. In Table 2 we report the number of iterations and the running times for Algorithms 1.1 and 2.1 on Problem 1 (results for other problems follow the same trend and do not yield any further insight). Note that the running times reported are serial, without any regard to the parallel nature of the algorithms. Because the iterative structure of the two algorithms is the same, this seems to be a meaningful and fair comparison. It is already clear from intuition that solving exactly the general nonlinear subproblems (P_l) in Algorithm 1.1 is very costly, and is unlikely to yield a competitive algorithm. This was easily confirmed by our experience. Of course, one could use heuristic considerations to (somehow) adjust dynamically the tolerance in solving the subproblems (in our experiment, all subproblems are solved to within the tolerance of 10^{-5}, same as the original problem). However, there is no convergence analysis to support such a strategy within Algorithm 1.1 in the constrained case (strictly speaking, there is no even a convergence proof for Algorithm 1.1 in the case under consideration, since the feasible set is nonconvex!). As discussed above, one of the motivations for our Algorithm 2.1 is precisely to provide a constructive implementable way of solving subproblems (P_l) inexactly, by solving their quadratic approximations. The results in Table 2 confirm that the proposed approach certainly makes sense.

Our next set of experiments concerns with the comparison between Algorithm 2.1 and serial SQP. It is not easy to come up with a meaningful comparison of a serial method and a parallel method on a serial machine. For example, the overall running time of a "parallel" method obtained on a serial machine is considered notoriously unreliable to predict time that would be required by its parallel implementation (i.e., just dividing the time by the number of "processors" is not a good measure). We therefore focus on indicators which we feel should be still meaningful for the actual parallel implementation, as discussed below. To evaluate the gain in computing the search directions, we report the total time spent solving quadratic programming subproblems within the SQP and within the serial implementation of Algorithm 2.1. Since no communication between the processors would be needed within this phase, the "expected" time for solving QPs by the parallel implementation of Algorithm

		SQP			Algorithm 2.1			
Problem ν, μ	p	QP time	iter #	calls to oracle	QP time (serial)	"speedup" %	iter #	calls to oracle
Prob.1	2	.43	11	12	.31	143	14	30
$\nu = 4$	4	8.81	10	13	5	175	31	36
	8	59.4	9	12	4.48	1323	13	20
	16	179	16	38	2.69	6656	16	23
	32	766	9	12	3.07	24951	9	10
$\nu = 10$	2	7.35	11	13	3.03	242	11	13
	4	42.4	9	12	5.5	771	10	13
	8	314	9	13	12.1	2603	11	15
	16	2167	8	13	17.6	12251	8	13
Prob.2	2	.45	12	13	.55	81	26	28
$\mu = 1, \nu = 4$	4	13.8	15	40	2.52	549	15	22
	8	23	15	19	2.16	1056	26	27
	16	3070	65	1030	21.4	14293	32	740
	32	29101	341	594	115.2	25263	341	471
$\mu = 2, \nu = 4$	2	2.57	18	19	1.4	183	17	30
	4	17.2	20	33	3.04	570	19	34
	8	91.5	15	21	6.96	1319	21	22
	16	1385	31	59	18.4	7511	29	68
	32	21865	65	76	186.2	11736	147	172
$\mu = 4, \nu = 4$	2	2.06	50	137	.63	316	28	59
	4	95	104	265	12.8	748	76	162
	8	224	34	35	15.2	1488	44	64
	16	1807	159	215	48.96	3687	288	319
	32	8077	95	155	47.7	16827	139	203
Prob.3	2	.91	10	11	.95	96	16	17
$\nu = 3$	4	28.6	54	89	6.95	411	58	132
	8	209	59	123	12.7	1651	53	109
	16	1522	141	223	112	1359	181	257
	32	35679	187	445	374	9551	201	511
$\nu = 4$	2	1.38	35	49	.725	194	34	66
	4	7.29	33	56	1.23	592	29	55
	8	134	88	132	12.9	1044	154	235
	16	2222	198	407	46.3	4802	274	599
	32	23289	267	630	100.5	23173	300	752

Table 1: **Results for Algorithm 2.1 and the SQP method on sphere packing problems.** Starting points are generated randomly, with coordinates in $[-10, 10]$. Times of solving QPs are measured (in seconds) using the intrinsic Matlab function cputime.

		Algorithm 1.1		Algorithm 2.1	
Name	p	iter	time	iter	time
Prob.1	$p = 2$	1	.81	23	1.25
$\nu = 3$	$p = 4$	9	19.1	13	2.35
	$p = 8$	5	153.8	10	10.8
	$p = 16$	5	287.6	9	66.5

Table 2: **Results for Algorithms 1.1 and 2.1 on Problem 1.** Starting points are feasible, generated randomly. Times are measured (in seconds) using the intrinsic Matlab function cputime.

2.1 can indeed be obtained dividing by the number of processors. It is somewhat surprising that even for smaller problems, see Table 1, the *serial* time for solving QPs in Algorithm 2.1 is already considerably smaller than in standard SQP, with the difference becoming drastic for larger problems. If we approximate the "speedup" efficiency for computing directions in parallel by

$$\frac{\text{time spent solving QPs in SQP}}{\text{time spent solving QPs in serial Algorithm 2.1}} * 100\,,$$

then these efficiencies vary from acceptable (around 80%) to very high (over 10000%), and grow fast with the size of the problem. This confirms our motivation as discussed in § 2, i.e., smaller QPs are significantly easier to solve. Obviously, the possible price to pay in Algorithm 2.1 is "deterioration" of directions compared to a good implementation of the full SQP algorithm. This issue is addressed by reporting the numbers of iterations and calls to the oracle evaluating the function and derivatives values, see Table 1. We note that the numbers of iterations and function/derivatives evaluations are usually slightly higher for Algorithm 2.1, but not always. Overall, those numbers for the two methods are quite similar. Given the impressive gain that Algorithm 2.1 exhibits in solving QP subproblems, this indicates that the parallel implementation of this algorithm should indeed be efficient, at least when computing the function and derivatives is cheap relative to solving QPs. In particular, this should be the case for large-scale problems where the functions and their derivatives are given explicitly.

5 Concluding remarks

Two new parallel constrained optimization algorithms based on the variables distribution (PVD) have been presented. The first one consists of a parallel sequential quadratic programming approach for the case of block-separable constraints. This is the first PVD-type method whose convergence has been established for nonconvex feasible sets. The second proposed algorithm employs approximate projected gradient directions for the case of general (inseparable) convex constraints. The use of inexact directions is of particular relevance when the projection operation is computationally costly.

References

[1] D.P. Bertsekas. *Nonlinear programming.* Athena Scientific, Belmont, Massachusetts, 1995.

[2] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation.* Prentice–Hall, Inc, Englewood Cliffs, New Jersey, 1989.

[3] J.F. Bonnans, J.Ch. Gilbert, C. Lemaréchal, and C. Sagastizábal. *Optimisation Numérique: aspects théoriques et pratiques.* Springer–Verlag, Berlin, 1997.

[4] J.H. Conway and N.J.C. Sloane. *Sphere Packings, Lattices and Groups.* Springer–Verlag, New York, 1988.

[5] R. W. Cottle, J.-S. Pang, and R. E. Stone. *The Linear Complementarity Problem.* Academic Press, New York, 1992.

[6] M.C. Ferris and O.L. Mangasarian. Parallel variable distribution. *SIAM Journal on Optimization*, 4:815–832, 1994.

[7] M. Fukushima. Parallel variable transformation in unconstrained optimization. *SIAM Journal on Optimization*, 8:658–672, 1998.

[8] N.I.M. Gould. On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem. *Mathematical Programming*, 32:90–99, 1985.

[9] S.-P. Han. A globally convergent merthod for nonlinear programming. *Journal of Optimization Theory and Applications*, 22:297–309, 1977.

[10] S.-P. Han. Optimization by updated conjugate subspaces. In *Numerical Analysis, No. 140, Pitman Research Notes in Mathematics*, pages 82–97. Longman Scientific and Technical, Burnt Mill, England, 1986.

[11] Y. Y. Lin and J.-S. Pang. Iterative methods for large convex quadratic programs: A survey. *SIAM Journal on Control and Optimization*, 25:383–411, 1987.

[12] C.-S. Liu and C.-H. Tseng. Parallel synchronous and asynchronous space-decomposition algorithms for large-scale minimization problems. *Computational Optimization and Applications*, 17:85–107, 2000.

[13] O.L. Mangasarian. *Nonlinear Programming.* McGraw–Hill, New York, 1969.

[14] O.L. Mangasarian. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Control and Optimization*, 33:1916–1925, 1995.

[15] O.L. Mangasarian and R. De Leone. Error bounds for strongly convex programs and (super)linearly convergent iterative schemes for the least 2-norm solution of linear programs. *Applied Mathematics and Optimization*, 17:1–14, 1988.

[16] J. M. Martínez. Two-phase model algorithm with global convergence for nonlinear programming. *Journal of Optimization Theory and Applications*, 96:397–436, 1998.

[17] D.Q. Mayne and E. Polak. A superlinearly convergent algorithm for constrained optimization problems. *Mathematical Programming Study*, 16:45–61, 1982.

[18] J.-S. Pang. Error bounds in mathematical programming. *Mathematical Programming*, 79:299–332, 1997.

[19] M.V. Solodov. New inexact parallel variable distribution algorithms. *Computational Optimization and Applications*, 7:165–182, 1997.

[20] M.V. Solodov. On the convergence of constrained parallel variable distribution algorithms. *SIAM Journal on Optimization*, 8:187–196, 1998.

[21] P. Tseng. Dual coordinate ascent methods for non-strictly convex minimization. *Mathematical Programming*, 59:231–248, 1993.

[22] E. Yamakawa and M. Fukushima. Testing parallel variable transformation. *Computational Optimization and Applications*, 13:253–274, 1999.