COST APPROXIMATION ALGORITHMS

The CA algorithm.

The notion of Cost Approximation (CA) was created in the thesis [39], to describe the construction of the subproblem of a class of iterative methods in mathematical programming. In order to explain the notion of CA, we will consider the following conceptual problem (the full generality of the algorithm is explained in detail in the book [45] and in the papers [37, 38, 40, 42, 43, 44, 46]):

$$\text{minimize} \quad T(x) := f(x) + u(x), \tag{1a}$$
$$\text{subject to} \quad x \in X, \tag{1b}$$

where $X \subseteq \Re^n$ is nonempty, closed and convex, $u : \Re^n \mapsto \Re \cup \{+\infty\}$ is lower semi-continuous (l.s.c.), proper and convex, and $f : \Re^n \mapsto \Re \cup \{+\infty\}$ is continuously differentiable (for short, in C^1) on $\mathtt{dom}\, u \cap X$, where $\mathtt{dom}$ denotes effective domain. This problem is general enough to cover convex optimization ($f = 0$), unconstrained optimization ($f = 0$ and $X = \Re^n$), and differentiable constrained optimization ($u = 0$). We note that if $\mathtt{int}\,(\mathtt{dom}\, u) \cap X$ is nonempty, then any locally optimal solution x^* satisfies the inclusion

$$-\nabla f(x^*) \in \partial u(x^*) + N_X(x^*), \tag{2}$$

where N_X is the normal cone operator for X and ∂u is the subdifferential mapping for u. Equivalently, by the definitions of these two operators,

$$\nabla f(x^*)^{\mathrm{T}}(x - x^*) + u(x) - u(x^*) \geq 0, \ x \in X.$$

The CA algorithm was devised in order to place a number of existing algorithms for (1) in a common framework, thereby facilitating comparisons, for example, between their convergence properties. In short, the method works iteratively as follows. Note that, from (2), we seek a zero of the mapping $[\nabla f + \partial u + N_X]$. Given an iterate, $x^t \in \mathtt{dom}\, u \cap X$, this mapping is approximated by a monotone mapping, constructed so that a zero of which is easier to find. Such a point, y^t, is then utilized in the search for a new iterate, x^{t+1}, having the property that the value of some merit function for (1) is reduced sufficiently, for example through a line search in T along the direction of $d^t := y^t - x^t$.

Instances of the CA algorithm.

To obtain a monotone approximating mapping, we introduce a monotone mapping $\Phi^t : \mathtt{dom}\, u \cap X \mapsto \Re^n$, which replaces the (possibly nonmonotone) mapping ∇f; by subtracting off the error at x^t, $[\Phi^t - \nabla f](x^t)$, from Φ^t, so that the resulting mapping becomes $[\Phi^t + \partial u + N_X] + [\nabla f - \Phi^t](x^t)$, the CA subproblem becomes the inclusion

$$[\Phi^t + \partial u + N_X](y^t) + [\nabla f - \Phi^t](x^t) \ni 0^n. \tag{3}$$

We immediately reach an interesting fixed-point characterization of the solutions to (2):

Theorem 1 (Fixed-point [45]) *The point x^t solves (2) if and only if $y^t = x^t$ solves (3).*

This result is a natural starting point for devising stopping criteria for an algorithm.

Assume now that $\Phi^t \equiv \nabla \varphi^t$ for a convex function φ^t. We may then derive the inclusion equivalently as follows. At x^t, we replace f with the function φ^t, and subtract off the linearization of the error at x^t; the subproblem objective function then becomes

$$T_{\varphi^t}(y) := \varphi^t(y) + u(y) + [\nabla f(x^t) - \nabla \varphi^t(x^t)]^{\mathrm{T}} y.$$

It is straightforward to establish that (3) is the optimality conditions for the convex problem of minimizing T_{φ^t} over X.

Linearization methods.

Our first example instances utilize Taylor expansions of f to construct the approximations.

Let $u = 0$ and $X = \Re^n$. Let $\Phi^t(y) := (1/\gamma_t)Q^t y$, where $\gamma_t > 0$ and Q^t is a symmetric and positive definite mapping in $\Re^{n \times n}$. The inclusion (3) reduces to

$$\nabla f(x^t) + (1/\gamma_t)Q^t(y^t - x^t) = 0^n,$$

that is, $y^t = x^t - \gamma_t(Q^t)^{-1}\nabla f(x^t)$. The direction of $y^t - x^t$, $d^t := -\gamma_t(Q^t)^{-1}\nabla f(x^t)$, is the search direction of the class of *deflected gradient* methods, which includes the *steepest descent* method ($Q^t := I^n$, the identity matrix) and

deflected gradient
steepest descent

quasi-Newton methods (Q^t equals [an approximation of] $\nabla^2 f(x^t)$, if positive definite). (See further [47, 35, 50, 5].)

In the presence of constraints, this choice of Φ^t leads to $y^t = P_X^{Q^t}[x^t - \gamma_t (Q^t)^{-1}\nabla f(x^t)]$, where $P_X^{Q^t}[\cdot]$ denotes the projection onto X with respect to the norm $\|z\|_{Q^t} := \sqrt{z^{\mathrm{T}} Q^t z}$. Among the algorithms in this class we find the *gradient projection* algorithm ($Q^t := I^n$) and *Newton's* method ($Q^t := \nabla^2 f(x^t)$, $\gamma_t := 1$). (See [19, 27, 50, 5].)

A first-order Taylor expansion of f is obtained from choosing $\varphi^t(y) := 0$; this results in $T_{\varphi^t}(y) = \nabla f(x^t)^{\mathrm{T}} y$ (if $u = 0$ is still assumed), which is the subproblem objective in the *Frank–Wolfe* algorithm ([17, 5]).

We next provide the first example of the very useful fact that the result of the cost approximation (in the above examples a linearization), leads to different approximations of the original problem, and ultimately to different algorithms, depending on which *representation* of the problem to one applies the cost approximation.

Consider the problem

$$\begin{aligned} &\text{minimize} \quad f(x) && \text{(4a)}\\ &\text{subject to} \quad g_i(x) = 0, \qquad i = 1,\dots,\ell, && \text{(4b)} \end{aligned}$$

where f and g_i, $i = 1,\dots,\ell$, are functions in C^2. We may associate this problem with its first-order optimality conditions, which in this special case is

$$F(x^*,\lambda^*) := \begin{pmatrix} \nabla_x L(x^*,\lambda^*) \\ -\nabla_\lambda L(x^*,\lambda^*) \end{pmatrix} = \begin{pmatrix} 0^n \\ 0^\ell \end{pmatrix}, \qquad (5)$$

where $\lambda \in \Re^\ell$ is the vector of Lagrange multipliers for the constraints (4b), and $L(x,\lambda) := f(x) + \lambda^{\mathrm{T}} g(x)$ is the associated Lagrangean function. We consider using Newton's method for this system, and therefore introduce a (primal–dual) mapping $\Phi : \Re^{2(n+\ell)} \mapsto \Re^{n+\ell}$ of the form

$$\begin{aligned} \Phi((y,p),(x,\lambda)) &:= \nabla F(x,\lambda) \begin{pmatrix} y \\ p \end{pmatrix} \\ &= \begin{pmatrix} \nabla_x^2 L(x,\lambda) & \nabla g(x)^{\mathrm{T}} \\ -\nabla g(x) & 0 \end{pmatrix} \begin{pmatrix} y \\ p \end{pmatrix}. \end{aligned}$$

The resulting CA subproblem in (y,p) can be written as the following linear system:

$$\begin{aligned} \nabla_x^2 L(x,\lambda)(y-x) + \nabla g(x)^{\mathrm{T}} p &= 0^n, \\ \nabla g(x)(y-x) &= -g(x); \end{aligned}$$

this system constitutes the first-order optimality conditions for (e.g., [4, Sec. 10.4])

$$\begin{aligned} \text{minimize} \quad & f(x) + \nabla f(x)^{\mathrm{T}}(y-x) \\ & + (1/2)(y-x)^{\mathrm{T}} \nabla_x^2 L(x,\lambda)(y-x) \\ \text{subject to} \quad & g(x) + \nabla g(x)^{\mathrm{T}}(y-x) = 0^\ell, \end{aligned}$$

where we have added some fixed terms in the objective function for clarity. This is the generic subproblem of *sequential quadratic programming* (*SQP*) methods for the solution of (4); see, for example, [16, 5].

Regularization, splitting and proximal point methods.

We assume now that $f := f_1 + f_2$, where f_1 is convex on $\mathtt{dom}\, u \cap X$, and rewrite the cost mapping as

$$\begin{aligned} [\nabla f + \partial u + N_X] = [\nabla \varphi^t + \nabla f_1 + \partial u + N_X] \\ - [\nabla \varphi^t - \nabla f_2]. \end{aligned}$$

The CA subproblem is, as usual, derived by fixing the second term at x^t; the difference to the original setup is that we have here performed an *operator splitting* in the mapping ∇f to keep an additive part from being approximated. (Note that such a splitting can always be found by first choosing f_1 as a convex function, and then define $f_2 := f - f_1$. Note also that we can derive this subproblem from the original derivation by simply redefining $\varphi^t := \varphi^t + f_1$.) We shall proceed to derive a few algorithms from the literature.

quasi-Newton
gradient projection
Newton's → Newton's method
Frank–Wolfe
representation → problem representation
sequential quadratic programming
operator splitting

Consider choosing $\varphi^t(y) = 1/(2\gamma_t)\|y - x^t\|^2$, $\gamma_t > 0$. If $f_2 = 0$, then we obtain the subproblem objective $T_{\varphi^t}(y) = T(y) + 1/(2\gamma_t)\|y - x^t\|^2$, which is the subproblem in the *proximal point* algorithm (e.g., [33, 34, 32, 51, 52]). This is the most classical algorithm among the regularization methods. More general choices of strictly convex functions φ^t are of course possible, leading for example to the class of regularization methods based on Bregman functions ([9, 14, 22]) and ψ-divergence functions ([54, 23]). If, on the other hand, $f_1 = 0$, then we obtain the gradient projection algorithm if also $u = 0$.

We can also construct algorithms in between these two extremes, yielding a true operator splitting. If both f_1 and f_2 are nonzero, choosing $\varphi^t = 0$ defines a *partial linearization* ([25]) of the original objective, wherein only f_2 is linearized. Letting $x = (x_1^{\mathrm{T}}, x_2^{\mathrm{T}})^{\mathrm{T}}$, the choice $\varphi^t(y) = 1/(2\gamma_t)\|y_1 - x_1^t\|^2$ leads to the *partial proximal point* algorithm ([20, 7]); choosing $\varphi^t(y) = f(y_1, x_2^t)$ leads to a linearization of f in the variables x_2.

Several well-known methods can be derived either directly as CA algorithms, or as *inexact proximal point* algorithms. For example, the *Levenberg–Marquardt* algorithm ([49, 5]), which is a Newton-like algorithm wherein a scaled diagonal matrix is added to the Hessian matrix in order to make the resulting matrix positive definite, is the result of solving the proximal point subproblem with one iteration of a Newton algorithm. Further, the *extra-gradient* algorithm of Korpelevich [24] is the result of instead applying one iteration of the gradient projection algorithm to the proximal point subproblem.

The perhaps most well-known splitting algorithm is otherwise the class of *matrix splitting* methods in quadratic programming (e.g., [35, 36, 29, 28]). In a quadratic programming problem, we have $f(x) = \frac{1}{2}x^{\mathrm{T}}Ax + q^{\mathrm{T}}x$, where $A \in \Re^{n\times n}$. A splitting (A_1^t, A_2^t) of this matrix is one for which $A = A_1^t + A_2^t$, and it is further termed regular if $A_1^t - A_2^t$ is positive definite. Matrix splitting methods correspond to choosing $f_1(x) = \frac{1}{2}x^{\mathrm{T}}A_1^t x$, and results in the CA subproblem mapping $y \mapsto A_1^t y + [A_2^t x^t + q]$, which obviously is monotone whenever A_1^t was chosen positive semi-definite.

Due to the fact that proximal point and splitting methods have dual interpretations as augmented Lagrangean algorithms ([51]), a large class of multiplier methods is included among the CA algorithms. See [45, Chapter 3.2–3.3] for more details.

Perturbation methods.

All the above algorithms assume that (i) the mappings ∂u and N_X are left intact and (ii) that the CA subproblem has the fixed-point property of Theorem 1. We here relax these assumptions, and are then able to derive subgradient algorithms as well as perturbed CA algorithms which include both regularization algorithms and exterior penalty algorithms.

Let $[\Phi^t + N_X] + [\nabla f + \partial u + \Phi^t]$ represent the original mapping, having moved ∂u to the second term. Then by letting any element $\xi_u(x^t) \in \partial u(x^t)$ represent this point-to-set mapping at x^t, we reach the subproblem mapping of the *auxiliary problem principle* of Cohen [12]. Further letting $\Phi^t(y) = (1/\gamma_t)[y - x^t]$ yields the subproblem in the classical *subgradient optimization* scheme ([48, 53]), where, assuming further that $f = 0$, $y^t := P_X[x^t - \gamma_t\xi_u(x^t)]$. (Typically, $\ell_t := 1$ is taken.)

---

proximal point
partial linearization
partial proximal point
inexact proximal point
Levenberg–Marquardt
extra-gradient
matrix splitting
auxiliary problem principle
subgradient optimization
epi-convergent

Let again $[\Phi^t + \partial u + N_X] + [\nabla f + \Phi^t]$ represent the original problem mapping, but further let u be replaced by an *epi-convergent* sequence $\{u^t\}$ of l.s.c., proper and convex functions. An example of an epi-convergent sequence of convex functions is provided by convex exterior penalty functions. In this way, we can construct CA algorithm that approximate the objective function and simultaneously replace some of the constraints of the problem with exterior penalties. See [3, 13] for example methods of this type.

One important class of regularization methods takes as the subproblem mapping $[\Phi^t + \nabla f + \partial u + N_X]$, where Φ^t is usually taken to be strongly monotone [cf. (10) below]. This subproblem mapping evidently does not have the fixed-point property, as it is not identical to the original one at x^t unless $\Phi^t(x^t) = 0^n$ holds. In order to ensure convergence, we must therefore force the sequence $\{\Phi^t\}$ of mappings to tend to zero; this is typically done by constructing the sequence as $\Phi^t := (1/\gamma_t)\Phi$ for a fixed mapping Φ and for a sequence of $\gamma_t > 0$ constructed so that $\{\gamma_t\} \to \infty$ holds. For this class of algorithms, Browder [10] has established convergence to a unique limit point x^* which satisfies $-\Phi(x^*) \in N_{X^*}(x^*)$, where X^* is the solution set of (2). The origin of this class of methods is the work of Tikhonov [55] for *ill-posed* problems, that is, problems with multiple solutions. The classical regularization mapping is the scaled identity mapping, $\Phi^t(y) := (1/\gamma_t)[y]$, which leads to least-squares (least-norm) solutions. See further the references [56, 49].

Variational inequality problems.

Consider the following extension of (2):

$$-F(x^*) \in \partial u(x^*) + N_X(x^*), \tag{6}$$

where $F : X \mapsto \Re^n$ is a continuous mapping on X. When $F = \nabla f$ we have the situation in (2), and also in the case when $F(x,y) = (\nabla_x \Pi(x,y)^{\mathrm{T}}, -\nabla_y \Pi(x,y)^{\mathrm{T}})^{\mathrm{T}}$ holds for some saddle function Π on some convex product set $X \times Y$ [cf. (5)], the *variational inequality problem* (6) has a direct interpretation as the necessary optimality conditions for an optimization problem. In other cases, however, a merit function (or, objective function), for the problem (6) is not immediately available. We will derive a class of suitable merit functions below.

Given the convex function $\varphi : \mathtt{dom}\, u \cap X \mapsto \Re$ in C^1 on $\mathtt{dom}\, u \cap X$, we introduce the function

$$\psi(x) := \operatorname*{supremum}_{y \in X} L(y,x), \quad x \in \mathtt{dom}\, u \cap X, \tag{7a}$$

where

$$L(y,x) := u(x) - u(y) + \varphi(x) - \varphi(y) \tag{7b}$$

$$+ [F(x) - \nabla\varphi(x)]^{\mathrm{T}}(x-y). \tag{7c}$$

We introduce the optimization problem

$$\operatorname*{minimize}_{x \in X} \psi(x). \tag{7d}$$

Theorem 2 (Gap function [45]) *For any $x \in X$, $\psi(x) \geq 0$ holds. Further, $\psi(x) = 0$ if and only if x solves (6). Hence, the solution set of (6) (if nonempty) is identical to that of the optimization problem (7d), and the optimal value is zero.*

The theorem shows that the CA subproblem defines an auxiliary function ψ which measures the violation of (6), and which can be used (directly or indirectly) as a merit function in an algorithm.

To immediately illustrate the possible use of this result, let us consider the extension of Newton's method to the solution of (6). Let $x \in \mathtt{dom}\, u \cap X$, and consider the following cost approximating mapping: $y \mapsto \Phi(y,x) := \nabla F(x)(y-x)$. The CA subproblem then is the linearized variational inequality problem of finding $y \in \mathtt{dom}\, u \cap X$ such that

$$[F(x) + \nabla F(x)^{\mathrm{T}}(y-x)]^{\mathrm{T}}(z-y) + u(z) \tag{8a}$$

$$-u(y) \geq 0, \qquad \forall z \in X. \tag{8b}$$

Assuming that x is not a solution to (6), we are interested in utilizing the direction $d := y - x$ in a line search based on a merit function. We will utilize the *primal gap function* ([62, 2]) for this purpose, which corresponds to the choice $\varphi := 0$ in the definition of ψ. We denote the

ill-posed
variational inequality problem
primal gap function

primal gap function by ψ_{p}. Let w be an arbitrary solution to its inner problem, that is, $\psi_{\mathrm{p}}(x) = u(x) - u(w) + F(x)^{\mathrm{T}}(x-w)$. The step length is chosen such that the value of ψ_{p} decreases sufficiently; to show that this is possible, we use Danskin's Theorem and the variational inequality (8) with $z = w$ to obtain [the max is taken over all w defining $\psi_{\mathrm{p}}(x)$]

$$\begin{aligned}\psi_{\mathrm{p}}'(x;d) := \max_{w}\{ & [F(x)+\nabla F(x)^{\mathrm{T}}(x-w)]^{\mathrm{T}} d \\ & + u'(x;d)\} \\ & \leq -\psi_{\mathrm{p}}(x) - d^{\mathrm{T}}\nabla F(x)^{\mathrm{T}} d,\end{aligned}$$

which shows that d defines a direction of descent with respect to the merit function ψ_{p} at all points outside the solution set, whenever F is monotone and in C^1 on $\mathtt{dom}\, u \cap X$. (See also [30] for convergence rate results.) So, if Newton's method is supplied with a line search with respect to the primal gap function, it is globally convergent for the solution of variational inequality problems.

The merit function ψ and the optimization problem (7d) cover several examples previously considered for the solution of (6).

The primal gap function, as typically all other gap functions, is nonconvex, and further also nondifferentiable in general. In order to utilize methods from differentiable optimization, we consider letting φ be *strictly* convex, whence the solution y^t to the inner problem (7a) is unique. Under the additional assumption that $\mathtt{dom}\, u \cap X$ is bounded and that u is in C^1 on this set, ψ is in C^1 on $\mathtt{dom}\, u \cap X$. Among the known differentiable gap functions that are covered by this class of merit functions we find those of Auchmuty [1], Fukushima [18], Larsson and Patriksson [26, 40], and Marcotte et al. [59, 60, 61, 31].

Descent properties.

Optimization.

Assume that x^t is not a solution to (2). We are interested in the conditions under which the direction of $d^t := y^t - x^t$ provides a descent direction for the merit function T. Let $d^t := \bar{y}^t - x^t$, where $\bar{y}^t$ is a possibly inexact solution to (3). Then, if $\Phi^t = \nabla\varphi^t$, the requirement is that

$$T_{\varphi^t}(\bar{y}^t) < T_{\varphi^t}(x^t), \tag{9}$$

that is, any improvement in the value of the subproblem objective over that at the current iterate is enough to provide a descent direction. To establish this result, one simply utilizes the convexity of φ^t and u and the formula for the directional derivative of T in the direction of d^t (see [45, Proposition 2.14.b]). We further note that (9) is possible to satisfy if and only if x^t is not a solution to (2); this result is in fact a special case of Theorem 1.

If Φ^t has stronger monotonicity properties, descent is also obtained when Φ^t is not necessarily a gradient mapping, and, further, if it is Lipschitz continuous then we can establish measures of the steepness of the search directions, extending the gradient relatedness conditions of unconstrained optimization. Let Φ^t be *strongly monotone* on $\mathtt{dom}\, u \cap X$, that is, for $x, y \in \mathtt{dom}\, u \cap X$,

$$[\Phi^t(x) - \Phi^t(y)]^{\mathrm{T}}(x-y) \geq m_{\Phi^t}\|x-y\|^2, \tag{10}$$

for some $m_{\Phi^t} > 0$. This can be used to establish that

$$T'(x^t; d^t) \leq -m_{\Phi^t}\|d^t\|^2.$$

If y^t is not an exact solution to (3), in the sense that for a vector $\bar{y}^t$, we satisfy a perturbation of (3) where its right-hand side 0^n is replaced by $r^t \neq 0^n$, then $d^t := \bar{y}^t - x^t$ is a descent direction for T at x^t if $\|r^t\| < m_{\Phi^t}\|d^t\|$.

Variational inequality problems.

The requirements for obtaining a descent direction in the problem (6) are necessarily much stronger than in the problem (2), the reason being the much more complex form that the merit functions for (6) takes. (For example, the directional derivative of T at x in any direction d depends only on those quantities, while the directional derivative of ψ depends also on the argument y which defines its value at x.) Typically, monotonicity of the mapping F is required, as is evidenced in the above example of the Newton method. If further a differentiable merit function is used, the requirements are slightly strengthened, as the following example result shows.

Theorem 3 (Descent properties [60, 45]) *Assume that X is bounded, u is finite on X and F is monotone and in C^1 on X. Let*

---

strongly monotone

$\varphi : X \times X \mapsto \Re$ be a continuously differentiable function on $X \times X$ of the form $\varphi(y,x)$, strictly convex in y for each $x \in X$. Let $\alpha > 0$. Let $x \in X$, y be the unique vector in X satisfying

$$\psi_\alpha(x) := \underset{y \in X}{\operatorname{maximum}}\ L_\alpha(y,x),$$

where

$$L_\alpha(y,x) := u(x) - u(y) + (1/\alpha)[\varphi(x,x) - \varphi(y,x)] + [F(x) - (1/\alpha)\nabla_y \varphi(x,x)]^{\mathrm{T}}(x-y).$$

Then, with $d := y - x$, either d satisfies

$$\psi'_\alpha(x;d) \le -\gamma\psi_\alpha(x), \qquad \gamma \in (0,1),$$

or

$$\psi_\alpha(x) \le -1/[\alpha(1-\gamma)](\varphi(y,x) + \nabla_x \varphi(y,x)^{\mathrm{T}} d).$$

A descent algorithm is devised from this result as follows. For a given $x \in X$ and choice of $\alpha > 0$, the CA subproblem is solved with the scaled cost approximating, continuous and iteration-dependent function φ. If the resulting direction does not have the descent property, then the value of α is increased and the CA subproblem rescaled and resolved. Theorem 3 shows that a sufficient increase in the value of α will produce a descent direction unless x solves (6).

Step length rules.

In order to establish convergence of the algorithm, the step length taken in the direction of d^t must be such that the value of the merit function decreases sufficiently. An exact line search obviously works, but we will introduce simpler step length rules that do not require a one-dimensional minimization to be performed.

The first is the *Armijo rule*. We assume temporarily that $u = 0$. Let $\alpha, \beta \in (0,1)$, and $\ell := \beta^{\bar{\imath}}$, where $\bar{\imath}$ is the smallest nonnegative integer i such that

$$f(x^t + \beta^i d^t) - f(x^t) \le \alpha\beta^i \nabla f(x^t)^{\mathrm{T}} d^t. \qquad (11)$$

There exists a finite integer such that (11) is satisfied for any search direction $\bar{d}^t := \bar{y}^t - x^t$ satisfying (9), by the descent property and Taylor's formula (see [45, Lemma 2.24.b]).

Armijo rule
relaxation step
divergent series step length rule

In the case where $u \neq 0$, however, the situation becomes quite different, since $T := f + u$ is nondifferentiable. Simply replacing $\nabla f(x^t)^{\mathrm{T}} d^t$ with $T'(x^t; d^t)$ does not work. We can however use an overestimate of the predicted decrease $T'(x^t; d^t)$. Let $\alpha, \beta \in (0,1)$, and $\ell := \beta^{\bar{\imath}}$, where $\bar{\imath}$ is the smallest nonnegative integer i such that

$$T(x^t + \beta^i d^t) - T(x^t) \le \alpha\beta^i [\nabla\varphi^t(x^t) - \nabla\varphi^t(y^t)]^{\mathrm{T}} d^t,$$

where now y^t necessarily is an exact solution to (3), and φ^t must further be strictly convex. We note that $T'(x^t; d^t) \le [\nabla\varphi^t(x^t) - \nabla\varphi^t(y^t)]^{\mathrm{T}} d^t$ indeed holds, with equality in the case where $u = 0$ and $X = \Re^n$ (see [45, Remark 2.28]).

To develop still simpler step length rules, we further assume that ∇f is Lipschitz continuous, that is, that for $x, y \in \operatorname{\mathtt{dom}} u \cap X$,

$$\|\nabla f(x) - \nabla f(y)\| \le M_{\nabla f}\|x - y\|,$$

for some $M_{\nabla f} > 0$. The Lipschitz continuity assumption implies that for every $\ell \in [0,1]$,

$$T(x^t + \ell d^t) - T(x^t) \le \ell[\nabla\varphi^t(x^t) - \nabla\varphi^t(y^t)]^{\mathrm{T}} d^t + (M_{\nabla f}/2)\ell^2 \|d^t\|^2;$$

adding a strong convexity assumption on φ^t yields that

$$T(x^t + \ell d^t) - T(x^t) \le \ell(-m_{\varphi^t} + M_{\nabla f}\ell/2)\|d^t\|^2.$$

This inequality can be used to validate the *relaxation step*, which takes

$$\ell_t \in (0, 2m_{\varphi^t}/M_{\nabla f}) \cap [0,1], \qquad (12)$$

and the *divergent series step length rule*,

$$[0,1] \supset \{\ell_t\} \to 0, \qquad \sum_{t=0}^{\infty} \ell_t = \infty. \qquad (13)$$

In the case of (12), descent is guaranteed in each step, while in the case of (13), descent is guaranteed after a finite number of iterations.

Convergence properties.

Convergence of the CA algorithm can be established under many combinations of (i) the properties of the original problem mappings, (ii) the choice of forms and convexity properties of the cost approximating mappings, (iii) the choice of accuracy in the computations of the CA subproblem solutions, (iv) the choice of merit function, and (v) the choice of step length rule. A subset of the possible results is found in [45, Chapters 5–9]. Evident from these results is that convergence relies on reaching a critical mass in the properties of the problem and algorithm, and that, given that this critical mass is reached, there is a very large freedom-of-choice how this mass is distributed. So, for example, weaker properties in the monotonicity of the subproblem must be compensated both by stronger coercivity conditions on the merit function and by the use of more accurate subproblem solutions and step length rules.

Decomposition CA algorithms.

Assume that $\mathtt{dom}\, u \cap X$ is a *Cartesian product* set, that is, for some finite index set $\mathcal{C}$ and positive integers n_i with $\sum_{i\in\mathcal{C}} n_i = n$,

$$X = \prod_{i\in\mathcal{C}} X_i, \qquad X_i \subseteq \Re^{n_i};$$
$$u(x) = \sum_{i\in\mathcal{C}} u_i(x_i), \qquad u_i : \Re^{n_i} \mapsto \Re \cup \{+\infty\}.$$

Such problems arise in applications of equilibrium programming, for example in traffic ([41]) and Nash ([21]) equilibrium problems; of course, box constrained and unconstrained problems fit into this framework as well.

The main advantage of this problem structure is that one can devise several decomposition versions of the CA algorithm, wherein components of the original problem are updated upon in parallel or sequentially, independently of each other. With the right computer environment at hand, this can mean a dramatic increase in computing efficiency. We will look at three computing models for decomposition CA algorithm, and compare their convergence characteristics. In all three cases, decomposition is achieved by choosing the cost approximating mapping separable with respect to the partition of $\Re^n$ defined by $\mathcal{C}$:

$$\Phi(x)^{\mathrm{T}} = [\Phi_1(x_1)^{\mathrm{T}}, \ldots, \Phi_{|\mathcal{C}|}(x_{|\mathcal{C}|})^{\mathrm{T}}]. \qquad (14)$$

Cartesian product
sequential → sequential CA algorithm
Gauss–Seidel

The individual subproblems, given x, then are to find y_i, $i \in \mathcal{C}$, such that

$$\Phi_i(y_i) + \partial u_i(y_i) + N_{X_i}(y_i) + F_i(x) - \Phi_i(x_i) \ni 0^{n_i};$$

if $\Phi_i \equiv \nabla\varphi_i$ for some convex function $\varphi_i : \mathtt{dom}\, u_i \cap X_i \mapsto \Re$ in C^1 on $\mathtt{dom}\, u_i \cap X_i$, then this is the optimality conditions for

$$\operatorname*{minimize}_{y_i\in X_i}\ T_{\varphi_i}(y_i) := \varphi_i(y_i) + u_i(y_i) + [F_i(x) - \nabla\varphi_i(x_i)]^{\mathrm{T}} y_i.$$

Sequential decomposition.

The *sequential* CA algorithm proceeds as follows. Given an iterate $x^t \in \mathtt{dom}\, u \cap X$ at iteration t, choose an index $i_t \in \mathcal{C}$ and a cost approximating mapping $\Phi^t_{i_t}$, and solve the problem of finding $y^t_{i_t} \in \Re^{n_{i_t}}$ such that $(i = i_t)$

$$\Phi^t_i(y^t_i) + \partial u_i(y^t_i) + N_{X_i}(y^t_i) + F_i(x^t) - \Phi^t_i(x^t_i) \ni 0^{n_i}.$$

Let $y^t_j := x^t_j$ for all $j \in \mathcal{C} \setminus \{i_t\}$ and $d^t := y^t - x^t$. The next iterate, x^{t+1}, is then defined by $x^{t+1} := x^t + \ell_t d^t$, that is,

$$x^{t+1}_j := \begin{cases} x^t_j + \ell_t(y^t_j - x^t_j), & j = i_t, \\ x^t_j, & j \neq i_t, \end{cases}$$

for some value of ℓ_t such that $x^t_{i_t} + \ell_t(y^t_{i_t} - x^t_{i_t}) \in \mathtt{dom}\, u_{i_t} \cap X_{i_t}$ and the value of a merit function ψ is reduced sufficiently.

Assume that F is the gradient of a function $f : \mathtt{dom}\, u \cap X \mapsto \Re$. Let the sequence $\{i_t\}$ be chosen according to the *cyclic rule*, that is, in iteration t,

$$i_t := t \pmod{|\mathcal{C}|} + 1.$$

Choose the cost approximating mapping $(i = i_t)$

$$y_i \mapsto \Phi^t_i(y_i) := \nabla_i f(x^t_{\neq i}, y_i), \quad y_i \in \mathtt{dom}\, u_i \cap X_i.$$

Note that this mapping is monotone whenever f is convex in x_i. Since $\Phi^t_i(x^t_i) = \nabla_i f(x^t)$, the CA subproblem is equivalent (under this convexity assumption) to finding

$$y^t_i \in \arg\operatorname*{minimum}_{y_i\in X_i} \{f(x^t_{\neq i}, y_i) + u_i(y_i)\}.$$

An exact line search would produce $\ell_t := 1$, since y^t_i minimizes $f(x_{\neq i}, \cdot) + u_i$ over $\mathtt{dom}\, u_i \cap X_i$ (the remaining components of x kept fixed), and so $x^{t+1}_i := y^t_i$. The iteration described is that of the classic *Gauss–Seidel* algorithm ([35]) (also known as the *relaxation algorithm*, the *coordinate descent method*, and the *method of successive displacements*), originally proposed for the solution of unconstrained problems. The Gauss–Seidel algorithm is hence a special case of the sequential CA algorithm.

In order to compare the three decomposition approaches, we last provide the step length requirement in the relaxation step length rule [cf. (12)]. The following interval is valid under the assumptions that for each $i \in \mathcal{C}$, $\nabla_i f$ is Lipschitz continuous on $\mathtt{dom}\, u_i \cap X_i$ and each mapping Φ_i^t is strongly monotone:

$$\ell_{i,t} \in \left(0, 2m_{\Phi_i^t}/M_{\nabla_i f}\right) \cap [0,1]. \qquad (15)$$

Synchronized parallel decomposition.

The synchronized parallel CA algorithm is identical to the original scheme, where the CA subproblems are constructed to match the separability structure in the constraints.

We presume the existence of a multiprocessor powerful enough to solve the $|\mathcal{C}|$ CA subproblems in parallel. (If fewer than $|\mathcal{C}|$ processors are available, then either some of the subproblems are solved in sequence or, if possible, the number of components is decreased; in either case, the convergence analysis will be the same, with the exception that the value of $|\mathcal{C}|$ may change.)

In the sequential decomposition CA algorithm, the step lengths are chosen individually for the different variable components, whereas the original CA algorithm uses a uniform step length, ℓ_t. If the relative scaling of the variable components is poor, in the sense that F or u changes disproportionally to unit changes in the different variables x_i, $i \in \mathcal{C}$, then this ill-conditioning may result in a poor performance of the parallel algorithm. Being forced to use the same step length in all the components can also have an unwanted effect due to the fact that the values of some variable components are close to their optimal ones while others may be far from optimal, in which case one might for example wish to use longer steps for the latter components. These two factors lead us to introduce the possibility to scale the component directions in the synchronized parallel CA algorithm. We stress that such effects cannot in general be accommodated into the original algorithm through a scaling of the mappings Φ_i^t. The scaling factors $s_{i,t}$ introduced are assumed to satisfy

$$0 < \underline{s}_i \leq s_{i,t} \leq 1, \qquad i \in \mathcal{C};$$

Note that the upper bound of one is without any loss of generality.

Assume that F is the gradient of a function $f : \mathtt{dom}\, u \cap X \mapsto \Re$. In the parallel algorithm, choose the cost approximating mapping of the form (14), where for each $i \in \mathcal{C}$,

$$y_i \mapsto \Phi_i^t(y_i) := \nabla_i f(x_{\neq i}^t, y_i), \quad y_i \in \mathtt{dom}\, u_i \cap X_i.$$

This mapping is monotone on $\mathtt{dom}\, u \cap X$ whenever f is convex in each component x_i. Since $\Phi_i^t(x_i^t) = \nabla_i f(x^t)$, $i \in \mathcal{C}$, it follows that the CA subproblem is equivalent (under the above convexity assumption on f) to finding

$$y_i^t \in \arg \operatorname*{minimum}_{y_i \in X_i} \{f(x_{\neq i}^t, y_i) + u_i(y_i)\}.$$

Choosing $\ell_t := 1$ and $s_{i,t} := 1$, $i \in \mathcal{C}$, yields $x^{t+1} := y^t$, and the resulting iteration is that of the *Jacobi* algorithm [35, 8] (also known as the *method of simultaneous displacements*). The Jacobi algorithm, which was originally proposed for the solution of systems of equations, is therefore a parallel CA algorithm where the cost approximating mapping is (16) and unit steps are taken.

The admissible step in component i is $\ell s_{i,t} \in [0,1]$, where

$$\ell \in \left(0, \min_{i \in \mathcal{C}} \left\{2m_{\Phi_i^t}/(s_{i,t} M_{\nabla f})\right\}\right). \qquad (16)$$

The maximal step is clearly smaller than in the sequential approach. To this conclusion contributes both the min-operation and that $M_{\nabla_i f} \leq M_{\nabla f}$; both of these requirements are introduced here because the update is made over

relaxation algorithm
coordinate descent method
Jacobi

all variable components simultaneously. (An intuitive explanation is that the sequential algorithm utilizes more recent information when it constructs the subproblems.) One may therefore expect the sequential algorithm to converge to a solution with a given accuracy in less iterations, although the parallel algorithm may be more efficient in terms of solution time; the scaling introduced by $s_{i,t}$ may also improve the performance of the parallel algorithm to some extent.

Although the parallel version of the algorithm may speed-up the practical convergence rate compared to the sequential one, the need for synchronization in carrying out the updating step will generally deteriorate performance, since faster processors must wait for slower ones. In the next section, we therefore introduce an asynchronous version of the parallel algorithm, in which processors do not wait to receive the latest information available.

Asynchronous parallel decomposition.

In the algorithms considered in this section, the synchronization step among the processors is removed. Because the speed of computations and communications can vary among the processors, and communication delays can be substantial, processors will perform the calculations out of phase with each other. Thus, the advantage of reduced synchronization is paid for by an increase in interprocessor communications, the use of outdated information, and a more difficult convergence detection (see [8]). (Certainly, the convergence analysis also becomes more complicated.) Recent numerical experiments indicate, however, that the introduction of such *asynchronous* computations can substantially enhance the efficiency of parallel iterative methods (e.g., [15, 6, 11]).

The model of partial asynchronism that we use is as follows. For each processor (or, variable component) $i \in \mathcal{C}$, we introduce

(a) initial conditions, $x_i(t) := x_i^0 \in X_i$, for all $t \leq 0$,
(b) a set $\mathcal{T}^i$ of times at which x_i is updated, and
(c) a variable $\tau_j^i(t)$ for each $j \in \mathcal{C}$ and $t \in \mathcal{T}^i$, denoting the time at which the value of x_j used by processor i at time t is generated by processor j, satisfying $0 \leq \tau_j^i(t) \leq t$ for all $j \in \mathcal{C}$ and $t \geq 0$.

We note that the sequential CA algorithm and the synchronized parallel CA algorithm can both be expressed as asynchronous algorithms: the cyclic sequential algorithm model is obtained from the choices $\mathcal{T}^i := \cup_{k\geq 0}\{|\mathcal{C}|k+i-1\}$ and $\tau_j^i(t) := t$, while the synchronous parallel model is obtained by choosing $\mathcal{T}^i := \{1, 2, \dots\}$ and $\tau_j^i(t) := t$, for all i, j and t.

The communication delay from processor j to processor i at time t is $t - \tau_j^i(t)$. The convergence of the *partially* asynchronous parallel decomposition CA algorithm is based on the assumption that this delay is upper bounded: there exists a positive integer P such that

(i) for every $i \in \mathcal{C}$ and $t \geq 0$, at least one element of $\{t, \dots, t+P-1\}$ belongs to $\mathcal{T}^i$, and
(ii) $0 \leq t - \tau_j^i(t) \leq P - 1$ holds for all $i, j \in \mathcal{C}$ and all $t \geq 0$.
(iii) $\tau_i^i(t) = t$ holds for all $i \in \mathcal{C}$ and all $t \geq 0$.

In short, parts (i) and (ii) of the assumption state that no processor waits for an arbitrarily long time to compute a subproblem solution or to receive a message from another processor. (Note that a synchronized model satisfies $P = 1$.) Part (iii) of the assumption states that processor i always uses the most recent value of its own component x_i of x, and is in [58] referred to as a computational *nonredundancy* condition. This condition holds in general when no variable component is updated simultaneously by more than one processor, as, for example, in message passing systems. For further discussions on the assumptions, we refer the reader to [8, 57]; we only remark that they are easily enforced in practical implementations.

The iterate $x(t)$ is defined by the vector of $x_i(t)$, $i \in \mathcal{C}$. At a given time t, processor i has knowledge of a possibly outdated version of $x(t)$;

asynchronous
partially $\rightarrow$ *partially asynchronous computation*

we let

$$x^i(t)^{\mathrm{T}} := \left[x_1(\tau_1^i(t))^{\mathrm{T}}, \dots, x_{|\mathcal{C}|}(\tau_{|\mathcal{C}|}^i(t))^{\mathrm{T}}\right]$$

denote this vector. [Note that (iii) above implies the relation $x_i^i(t) := x_i(\tau_i^i(t)) = x_i(t)$.]

To describe the (partially) asynchronous parallel CA algorithm, processor i updates $x_i(t)$ according to

$$x_i(t+1) := x_i(t) + \ell s_i(y_i(t) - x_i^i(t)), \quad t \in \mathcal{T}^i,$$

where $y_i(t)$ solves the CA subproblem defined at $x^i(t)$, and $s_i \in (0,1]$ is a scaling parameter. (We define $d_i(t) := y_i(t) - x_i^i(t)$ to be zero at each $t \notin \mathcal{T}^i$.)

The admissible step length for $i \in \mathcal{C}$ is $\ell s_i \in [0,1]$, where

$$\ell \in \left(0, \frac{2 \min_{i\in\mathcal{C}}\{m_{\Phi_i}/s_i\}}{M_{\nabla f}[1 + (|\mathcal{C}|+1)P]}\right). \tag{17}$$

If further for some $M \geq 0$ and every $i \in \mathcal{C}$, all vectors x, y in $\mathtt{dom}\, u \cap X$ with $x_i = y_i$ satisfy

$$\|\nabla_i f(x) - \nabla_i f(y)\| \leq M\|x - y\|, \tag{18}$$

then, in the above result, the step length restrictions are adjusted to

$$\ell \in \left(0, \frac{2 \min_{i\in\mathcal{C}}\{m_{\Phi_i}/s_i\}}{M_{\nabla f} + (|\mathcal{C}|+1)MP}\right).$$

(We interpret the property (18) as a quantitative measure of the coupling between the variables.)

Most important to note is that the upper bound on ℓ is (essentially) inversely proportional to the maximal allowed asynchronism P; this is very intuitive, since if processors take longer steps then they should exchange information more often. Conversely, the more outdated the information is, the less reliable it is, hence the shorter step.

The relations among the step lengths in the three approaches [cf. (15), (16), and (17)] quantify the intuitive result that utilizing an increasing degree of parallelism and asynchronism results in a decreasing quality of the step directions, due to the usage of more outdated information; subsequently, smaller step lengths must be used. More detailed discussions about this topic is found in [45, Section 8.7.2].

References

[1] Auchmuty, G.: 'Variational principles for variational inequalities', *Numerical Functional Analysis and Optimization* **10** (1989), 863–874.

[2] Auslender, A.: *Optimisation: Méthodes Numériques*, Masson, Paris, 1976.

[3] Auslender, A., Crouzeix, J. P., and Fedit, P.: 'Penalty-proximal methods in convex programming', *Journal of Optimization Theory and Applications* **55** (1987), 1–21.

[4] Bazaraa, M. S., Sherali, H. D., and Shetty, C. M.: *Nonlinear Programming: Theory and Algorithms*, second ed., John Wiley & Sons, New York, NY, 1993.

[5] Bertsekas, D. P.: *Nonlinear Programming*, second ed., Athena Scientific, Belmont, MA, 1999.

[6] Bertsekas, D. P., and Castañon, D. A.: 'Parallel synchronous and asynchronous implementations of the auction algorithm', *Parallel Computing* **17** (1991), 707–732.

[7] Bertsekas, D. P., and Tseng, P.: 'Partial proximal minimization algorithms for convex programming', *SIAM Journal on Optimization* **4** (1994), 551–572.

[8] Bertsekas, D. P., and Tsitsiklis, J. N.: *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, London, 1989.

[9] Bregman, L. M.: 'A relaxation method of finding a common point of convex sets and its application to problems of optimization', *Soviet Mathematics Doklady* **7** (1966), 1578–1581.

[10] Browder, F. E.: 'Existence and approximation of solutions of nonlinear variational inequalities', *Proceedings of the National Academy of Sciences, U.S.A.* **56** (1966), 1080–1086.

[11] Chajakis, E. D., and Zenios, S. A.: 'Synchronous and asynchronous implementations of relaxation algorithms for nonlinear network optimization', *Parallel Computing* **17** (1991), 873–894.

[12] Cohen, G.: 'Optimization by decomposition and coordination: A unified approach', *IEEE Transactions on Automatic Control* **ac-23** (1978), 222–232.

[13] Cominetti, R.: 'Coupling the proximal point algorithm with approximation methods', *Journal of Optimization Theory and Applications* **95** (1997), 581–600.

[14] Eckstein, J.: 'Nonlinear proximal point algorithms using Bregman functions, with applications to convex programming', *Mathematics of Operations Research* **18** (1993), 202–226.

[15] El Baz, D.: 'A computational experience with distributed asynchronous iterative methods for convex network flow problems': *Proceedings of the 28th IEEE Conference on Decision and Control*, 1989, pp. 590–591.

[16] Fletcher, R.: *Practical Methods of Optimization*, second ed., John Wiley & Sons, Chichester, U.K., 1987.

[17] Frank, M., and Wolfe, P.: 'An algorithm for quadratic programming', *Naval Research Logistics Quarterly* **3** (1956), 95–110.

[18] Fukushima, M.: 'Equivalent differentiable optimization problems and descent methods for asymmetric variational inequality problems', *Mathematical Programming* **53** (1992), 99–110.

[19] Goldstein, A. A.: 'Convex programming in Hilbert space', *Bulletin of the American Mathematical Society* **70** (1964), 709–710.

[20] Ha, C. D.: 'A generalization of the proximal point algorithm', *SIAM Journal on Control and Optimization* **28** (1990), 503–512.

[21] Harker, P. T., and Pang, J.-S.: 'Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms and applications', *Mathematical Programming* **48** (1990), 161–220.

[22] Kiwiel, K. C.: 'Generalized Bregman projections in convex feasibility problems', *Journal of Optimization Theory and Applications* **96** (1998), 139–157.

[23] Kiwiel, K. C.: 'Subgradient method with entropic projections for convex nondifferentiable minimization', *Journal of Optimization Theory and Applications* **96** (1998), 159–173.

[24] Korpelevich, G. M.: 'The extragradient method for finding saddle points and other problems', *Matekon* **13** (1977), 35–49.

[25] Larsson, T., and Migdalas, A.: 'An algorithm for nonlinear programs over Cartesian product sets', *Optimization* **21** (1990), 535–542.

[26] Larsson, T., and Patriksson, M.: 'A class of gap functions for variational inequalities', *Mathematical Programming* **64** (1994), 53–79.

[27] Levitin, E. S., and Polyak, B. T.: 'Constrained minimization methods', *USSR Computational Mathematics and Mathematical Physics* **6** (1966), 1–50.

[28] Luo, Z.-Q., and Tseng, P.: 'On the convergence of a matrix splitting algorithm for the symmetric monotone linear complementarity problem', *SIAM Journal on Control and Optimization* **29** (1991), 1037–1060.

[29] Mangasarian, O. L.: 'Convergence of iterates of an inexact matrix splitting algorithm for the symmetric monotone linear complementarity problem', *SIAM Journal on Optimization* **1** (1991), 114–122.

[30] Marcotte, P., and Dussault, J.-P.: 'A note on a globally convergent Newton method for solving monotone variational inequalities', *Operations Research Letters* **6** (1987), 35–42.

[31] Marcotte, P., and Zhu, D.: 'Global convergence of descent processes for solving non strictly monotone variational inequalities', *Computational Optimization and Applications* **4** (1995), 127–138.

[32] Martinet, B.: 'Détermination approchée d'un point fixe d'une application pseudo-contractante', *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences (Paris), Série A* **274** (1972), 163–165.

[33] Minty, G. J.: 'Monotone (nonlinear) operators in Hilbert space', *Duke Mathematical Journal* **29** (1962), 341–346.

[34] Moreau, J.-J.: 'Proximité et dualité dans un espace Hilbertien', *Bulletin de la Société Mathématique de France* **93** (1965), 273–299.

[35] Ortega, J. M., and Rheinboldt, W. C.: *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, NY, 1970.

[36] Pang, J.-S.: 'On the convergence of a basic iterative method for the implicit complementarity problem', *Journal of Optimization Theory and Applications* **37** (1982), 149–162.

[37] Patriksson, M.: 'Partial linearization methods in nonlinear programming', *Journal of Optimization Theory and Applications* **78** (1993), 227–246.

[38] Patriksson, M.: 'A unified description of iterative algorithms for traffic equilibria', *European Journal of Operational Research* **71** (1993), 154–176.

[39] Patriksson, M.: *A unified framework of descent algorithms for nonlinear programs and variational inequalities*, PhD thesis, Department of Mathematics, Linköping Institute of Technology, Linköping, Sweden, 1993.

[40] Patriksson, M.: 'On the convergence of descent methods for monotone variational inequalities', *Operations Research Letters* **16** (1994), 265–269.

[41] Patriksson, M.: *The Traffic Assignment Problem—Models and Methods*, Topics in Transportation. VSP BV, Utrecht, The Netherlands, 1994.

[42] Patriksson, M.: 'Merit functions and descent algorithms for a class of variational inequality problems', *Optimization* **41** (1997), 37–55.

[43] Patriksson, M.: 'Cost approximation: A unified framework of descent algorithms for nonlinear programs', *SIAM Journal on Optimization* **8** (1998), 561–582.

[44] Patriksson, M.: 'Decomposition methods for differentiable optimization problems over Cartesian product sets', *Computational Optimization and Applications* **9** (1998), 5–42.

[45] Patriksson, M.: *Nonlinear Programming and Variational Inequality Problems: A Unified Approach*, Vol. 23 of *Applied Optimization*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[46] PATRIKSSON, M.: 'Cost approximation algorithms with nonmonotone line searches for a general class of nonlinear programs', *Optimization* **44** (1999), 199–217.

[47] POLYAK, B. T.: 'Gradient methods for the minimisation of functionals', *USSR Computational Mathematics and Mathematical Physics* **3** (1963), 864–878.

[48] POLYAK, B. T.: 'A general method of solving extremum problems', *Soviet Mathematics Doklady* **8** (1967), 593–597.

[49] POLYAK, B. T.: *Introduction to Optimization*, Optimization Software, New York, NY, 1987.

[50] PSHENICHNY, B. N., AND DANILIN, YU. M.: *Numerical Methods in Extremal Problems*, MIR Publishers, Moscow, 1978.

[51] ROCKAFELLAR, R. T.: 'Augmented Lagrangians and applications of the proximal point algorithm in convex programming', *Mathematics of Operations Research* **1** (1976), 97–116.

[52] ROCKAFELLAR, R. T.: 'Monotone operators and the proximal point algorithm', *SIAM Journal on Control and Optimization* **14** (1976), 877–898.

[53] SHOR, N. Z.: *Minimization Methods for Non-Differentiable Functions*, Springer-Verlag, Berlin, 1985.

[54] TEBOULLE, M.: 'Entropic proximal mappings with applications to nonlinear programming', *Mathematics of Operations Research* **17** (1992), 670–690.

[55] TIKHONOV, A. N.: 'Solution of incorrectly formulated problems and the regularization method', *Soviet Mathematics Doklady* **4** (1963), 1035–1038.

[56] TIKHONOV, A. N., AND ARSENIN, V. YA.: *Solutions of Ill-Posed Problems*, John Wiley & Sons, New York, NY, 1977.

[57] TSENG, P., BERTSEKAS, D. P., AND TSITSIKLIS, J. N.: 'Partially asynchronous, parallel algorithms for network flow and other problems', *SIAM Journal on Control and Optimization* **28** (1990), 678–710.

[58] ÜRESIN, A., AND DUBOIS, M.: 'Asynchronous iterative algorithms: Models and convergence', in L. KRONSJÖ AND D. SHUMSHERUDDIN (eds.): *Advances in Parallel Algorithms*, 1992, pp. 302–342.

[59] WU, J. H., FLORIAN, M., AND MARCOTTE, P.: 'A general descent framework for the monotone variational inequality problem', *Mathematical Programming* **61** (1993), 281–300.

[60] ZHU, D. L., AND MARCOTTE, P.: 'Modified descent methods for solving the monotone variational inequality problem', *Operations Research Letters* **14** (1993), 111–120.

[61] ZHU, D. L., AND MARCOTTE, P.: 'An extended descent framework for variational inequalities', *Journal of Optimization Theory and Applications* **80** (1994), 349–366.

[62] ZUHOVICKIĬ, S. I., POLYAK, R. A., AND PRIMAK, M. E.: 'Two methods of search for equilibrium points of n-person concave games', *Soviet Mathematics Doklady* **10** (1969), 279–282.

Michael Patriksson
Department of Mathematics
Chalmers University of Technology
SE-412 96 Göteborg (Gothenburg)
Sweden
E-mail address: mipat@math.chalmers.se

Key words and phrases: linearization methods, gradient projection, quasi-Newton, Frank–Wolfe, sequential quadratic programming, operator splitting, proximal point, Levenberg–Marquardt, auxiliary problem principle, subgradient optimization, variational inequality problem, merit function, Cartesian product, Gauss–Seidel, Jacobi, asynchronous computations.