Alternating Directions Methods for the Parallel Solution of Large-Scale Block-Structured Optimization Problems

By

Spyridon A. Kontogiorgis

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN – MADISON

1994

Abstract

Prompted by advances in computer technology and the increasing confidence of decision makers in large-scale market models, practitioners of operations research are now tackling problems of increasing detail, complexity and size. This necessitates the development of new solution algorithms that exploit problem structure as well as the properties of the target hardware, in order to minimize turnaround time and maximize model utilization.

Many models in planning and scheduling exhibit a block-angular structure, that can represent spatial or temporal partial decomposability: decision variables can be broken down to largely independent blocks, that correspond to first-level decisions satisfying a subset of the constraints, which may represent a time period, or a geographical region, or a commodity. The blocks interact via coupling constraints related to second-level coordination of block decisions, such as shared resource allocation restrictions.

In this thesis we construct three efficient decomposition algorithms for such block-angular problems. These algorithms belong to the family of alternating directions methods, and can be thought of as block Gauss-Seidel iterative schemes for an augmented Lagrangian, that exploit the block structure. Alternatively, they can be thought of as Douglas–Rachford schemes for calculating a zero of the maximal monotone subgradient operator. Our algorithms are of the "fork–join" type, alternating a local and a global computation phase. In the local phase, decoupled optimization subproblems corresponding to blocks are solved. In the global phase, solution information is combined and a coordination problem is solved, the results of which are used in modifying the objective function of the subproblems. The algorithms are thus similar to price-directed decomposition, with proximal terms added for stability. In contrast with classic Dantzig-Wolfe decomposition, the coordination problem is very simple, and a closed-form solution can be obtained.

We implemented the algorithms on a state-of-the art parallel computer, the Connection Machine 5. We treat the machine in a Multiple Instruction-Multiple Data mode, as a collection of fast

processors, each with local memory, linked by fast networks. A highly-efficient system library for processor communication is used for the fast, distributed computation of the solution of the coordination problem; this reduces drastically the serial bottleneck associated with solving the master problem.

We present numerical results for the parallel solution of linear and quadratic multicommodity flow problems with these three methods; for comparison, we also solve these problems on a DEC 5000 workstation with the widely-used serial optimization package MINOS 5.4 . On large-scale problems, with tens of thousands of variables and constraints, the best of our parallel codes is one to two orders of magnitude faster than serial MINOS 5.4 . We also compare solution times with the Dantzig-Wolfe method on the CM-5, as implemented by Deleone [22]. For the larger problems, the best of our parallel codes is faster by a factor of 2 to 4.

This thesis is organized as follows : in chapter 1 we introduce the convex block-angular problem (**CBA**) and present two of its instances, the multicommodity flow problem, that arises in scheduling and transportation, and the scenario analysis problem, that arises in multi-period financial planning under uncertainty. We briefly present concepts from the fields of parallel computation and convex analysis that we will use in the development of our material. In chapter 2 we study an extended version of the method of Alternating Directions (ADI) as applied to a general optimization problem involving the minimization of the sum of two convex functions subject to linear constraints. The method consists of solving consecutively in each iteration two optimization problems, the objective function of which contains both Lagrangian and proximal terms. This is followed by a simple update of the Lagrangian terms. We provide a main convergence theorem, and also present two variations of the method. In chapter 3 we discuss some additional aspects of the structure of the block-angular problem (**CBA**) and present a first algorithm that specializes the ADI method to it. In chapters 4, 5 and 6 we derive and characterize three highly-parallel ADI algorithms for (**CBA**). These algorithms are such that the first subproblem for each ADI iteration decomposes into independent block-subproblems that can be solved in parallel, while the second ADI subproblem has a simple closed-form solution the computation of which can also be parallelized. Computational experience on the CM-5 is reported in chapter 7. We present our conclusions and directions for future research in chapter 8.

This thesis makes contributions to both the theory and the practice of the ADI method. On the theoretical front, we extend existing results, by relaxing assumptions on the properties of both the objective function and the constraint matrices. We prove convergence of a version of the ADI method, in which the penalty matrices are allowed to vary for finitely many iterations. This can be quite important in practical applications, since it allows for the tuning of the method to the problem, by incorporating run-time heuristics that can speed up the method considerably. We develop such a family of heuristics for the multicommodity flow problem in chapter 7. We also present a unified framework for the two main ADI schemes, the Douglas–Rachford and the Peaceman–Rachford methods, when applied to problem (**CBA**), by showing that the latter is an over-relaxation of the former.

On the programming front, we present an efficient implementation of the method on the CM-5 that takes advantage of the underlying hardware: the global coordination consists of the aggregation of vectors, performed efficiently by the system message-passing library. The largest linear problem we generated has 26,154 rows and 53,952 columns, in the LP representation, and was solved in 336 CPU seconds on a CM-5 with 64 processors.

Acknowledgements

I am grateful to Bob Meyer, my research advisor, for his expert advice, his solid support, and for his skillful guidance of my first steps in parallel computing. I am thankful to prof. Renato De Leone for an extended collaboration in a relaxed Mediterranean atmosphere. I also wish to thank profs. Olvi Mangasarian, Stephen Robinson and Michael Ferris, for having offered me a first-rate education in Mathematical Programming through their courses, their notes and many informal discussions, and also for serving on my doctoral committee.

Thanks are also due to profs. Carl de Boor, Seymour Parter, John Strikwerda and Amos Ron for an excellent series of courses in Numerical Analysis. I would like to thank especially Carl de Boor, for making it possible for me to participate in the graduate program at the University of Wisconsin, and John Strikwerda, for serving on my doctoral committee.

Many fellow students have provided me with good technical advice and have shared with me their knowledge; I would like to thank Jonathan Yackel and Armand Zakarian (my officemates), Misha Solodov, Steve Dirkse, Thanos Tsiolis, Yannis Schoinas and Stefanos Kaxiras. Manolis Tsangaris, Dionisis Pnevmatikatos and Sze-Ping Wong deserve special mention, for the help they provided, their friendship and their support.

I also want to acknowledge the friendship and support of those currently in other countries: Erwin Glattauer, Walter Blaschke, George Bathas, Leo Boutsikaris, João Meidânis, Liliane Maia, Jan Keppler, Maria Rangousi and Isabelle Pieri.

Special thanks to Athan Petridis, for being a life-long friend of adamant integrity.

My social life, coffee-drinking in particular, would definitely be less colorful if it weren't for my friends from the Hellenic Society: Yiannakis, Theo, DT, Kostas, Markos, "Handsome" Antonis, Sifis, Thanos and "Dirty" Harry. Many a scientific, personal, social, historical and sports issue was debated, with varying degrees of seriousness, over cups of steaming cappuccino at varying stages of perfection.

I dedicate this work to the memory of my mother and my grandmother, who both died during my graduate years in Wisconsin. "The dead live among us as on the other side of a veil" say the Naxi people of Yunnan – a wise metaphysics.

I gratefully acknowledge the Air Force Office of Scientific Research and the National Science Foundation for partially funding this research, through grants AFOSR-89-0410, F49620-94-1-0036, and CCR-8907671, CDA-9024618 and CCR-9306807, respectively.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

In this chapter we describe the setting for both the theoretical and computational work of this thesis. We begin by describing, in section 1.2, the mathematical notation we will employ. Section 1.3 presents a brief overview of the basic parallel modes of computation in use today, and discusses performance measures for parallel systems. It also discusses the *fork–join* methodology for coordinating parallel processes, that we employ in our design of algorithms. In section 1.4 we introduce the optimization problem we are solving and discuss in brief some of its realizations in practice. In section 1.5 we survey previous work on the problem and in section 1.6 we present concepts and definitions from convex analysis that we will use in subsequent chapters.

1.2 Mathematical notation

We describe here some of the general notation we will be using. Some notation particular to convex analysis is described in section 1.6. We are concerned exclusively with finite dimensional Hilbert spaces over the real numbers. All scalars, vectors and matrices in this thesis are real.

We let $\mathbb{R}^n$ denote the real n-dimensional space, and let $\mathbb{R}^n_+$ denote its non-negative orthant. The real line is denoted by $\mathbb{R}$. The linear space of $m \times n$ real matrices will be denoted by $\mathbb{R}^{m \times n}$. The identity matrix in the space $\mathbb{R}^{n \times n}$ will be denoted by $I_{n \times n}$, with indices omitted, if they can be easily inferred from the context.

We use lower-case latin and greek letters to represent vectors and scalars. Matrices and sets are represented by capital letters.

For a vector $x \in \mathbb{R}^n$ and a matrix $A \in \mathbb{R}^{m \times n}$ we denote the transposes by x^T and A^T, respectively. All untransposed vectors are assumed to be column vectors. We denote by $x(i)$ the i-th component of vector x, and by $A(i,j)$ the i-th component of the j-th column vector of matrix A.

The notation $\operatorname{diag}(A_1, \ldots, A_n)$ will denote a block-diagonal matrix with blocks $A_1, \ldots, A_n$ placed along the diagonal. The blocks can either be submatrices or scalars.

We let $x^T y$ denote the Euclidean inner product of x and y in $\mathbb{R}^n$, and we let, for any x in $\mathbb{R}^n$, $\|x\|_2$ stand for $(x^T x)^{\frac{1}{2}}$. The symbol $\|\cdot\|$ will represent an arbitrary norm.

Let now y be an arbitrary but fixed vector in $\mathbb{R}^n$. We will represent by $y^T(\bullet)$ the real-valued function that maps any x in $\mathbb{R}^n$ to $y^T x$.

A sequence of vectors $x^1, x^2, \ldots$ will be denoted by $\{x^i\}$, with similar notation for sequences of matrices and scalars. We will reserve superscript t to denote vectors and matrices generated at step $t = 0, 1, 2, \ldots$ of an iterative process. We will use the notation $x_{[1]}, x_{[2]}, \ldots, x_{[K]}$ to denote K disjoint sub-vectors that a vector x can be partitioned to. In case it is clear from the context, we sometimes let x represent the concatenation of the vectors $x_{[1]}, x_{[2]}, \ldots, x_{[K]}$.

For scalars α and β, the operators $\min\{\alpha, \beta\}$, $\max\{\alpha, \beta\}$ have the standard meaning. We define $\alpha_+ := \max\{\alpha, 0\}$ and $\alpha_- := \max\{-\alpha, 0\}$. For vectors x and y, $\min\{x, y\}$ and $\max\{x, y\}$ are to be taken component-wise. Similarly, for a vector x, x_+ and x_- are defined component-wise.

The expression $A := B$, or $B =: A$ means "A is defined to be equal to B".

Further notation will be defined where it is needed.

1.3 Parallel Scientific Computing

1.3.1 A classification

The field of parallel computing is currently undergoing rapid growth; new architectures are being developed and implemented, and sophisticated performance models are being studied, either analytically or with the help of simulation. There is extensive experimentation going on in the design and configuration of both the processors and the interconnection network.

For a general introduction to parallel computation one may consult, among others, the books by Hwang [51] and Lawson [60], on the design of hardware, and by Akl [3] and Bertsekas and Tsitsiklis [11], on the design of parallel scientific algorithms.

Although parallel programming is still far from being standardized, the basic desirable parallel computing modes have been characterized. One can attempt a broad classification of parallel systems on the basis of the implementation of the overall control of the computing process. Four major categories emerge:

- *Wavefront* systems are self-controlled. There is no direct timing mechanism; processes are event-driven and communicate asynchronously with each other.
- *Systolic* systems use time-ordered control; computing is subdivided into equal units of time, and proceeds among processing elements (=: PEs) synchronously in discrete time stages. Time multiplexing can be used to create a "virtual" system with more PEs.
- Single Instruction Multiple Data (*SIMD*) systems use a central control mechanism to broadcast instructions to PEs at regular lock-step (synchronous) intervals.
- Multiple Instruction Multiple Data (*MIMD*) systems use PEs with individual control over their part of the computation and their own execution rate; communication between PEs can be either asynchronous or by synchronization barriers.

The first two classes can be thought of as "extended" pipelines for scalar processors; they are more suited for specific numerical tasks, such as sorting, matrix operations and Fast Fourier Transform calculations, usually encountered in digital signal processing applications. They are usually implemented as massive arrays of simple processors, sometimes bit-serial. Their architectural layout is usually tailored to the computational task they will be performing; these are applications-specific systems.

In contrast, the last two classes are general-purpose and have more flexibility in their implementation. In particular, MIMD architectures have received the largest attention in the research community, and a number of quite diverse MIMD architectures are available on the computer market, in the high-performance bracket.

An advantage of MIMD systems is that the user is in control of the degree of "parallelization" of the application: tasks can be allocated specifically to PEs; the communication pattern between PEs as well as the frequency and timing of the message exchange can be explicitly specified. Many large-scale scientific problems have a structure that allows for their decomposition in relatively large pieces, that can be worked on fairly independently, in a *data parallel* fashion; these are natural candidates for a MIMD implementation. One such example are domain decomposition codes for the solution of partial differential equations. Another example is the method of Alternating Directions for convex programming, which is the focus of this thesis. Such methods exhibit *irregular* parallelism: the algorithm may execute different steps per domain. SIMD architectures have inherent difficulties in treating efficiently this kind of parallelism.

Another classification of parallel architectures is based on the size of the unit of parallelism, also known as the level of *granularity*. In *fine–grain* parallelism, micro-elements, such as instructions and data, are processed in parallel. In *coarse–grain* parallel systems, PEs operate as autonomous uniprocessors executing local programs, with limited coordination.

Granularity can also be assigned to a parallel algorithm, based on the size of the data unit it operates upon. As an example, in an algorithm for LU matrix factorization [90] it is possible to have:

- Fine–grain parallelism associated with element-level update operations.
- Medium–grain parallelism associated with row-level update operations.
- Coarse–grain parallelism associated with independent pivots.

In case several levels of granularity are possible, a choice is made by weighing a number of factors, such as the quality of the CPU, the amount of memory available per processor and the cost of synchronization and communication.

Parallel systems can also be classified on the basis of the implementation of the control mechanism (tightly vs. loosely coupled systems), the organization of memory (shared vs. distributed) and the topology of the interconnection network (grid, torus, tree etc.).

1.3.2 *Fork–Join* coordination

A high-level protocol for the global organization of the computation in a MIMD environment is *fork–join*, shown in Figure 1. In this protocol each PE is assigned a part of the current data of the problem. During the *fork*, or *data parallel*, phase each PE performs a round of calculations on these data, such as solving an optimization problem. When all PEs have completed the round, they signal to each other the termination of the phase; this is called *barrier synchronization*.

Then the *join*, or *global coordination*, phase takes place, during which the PEs combine data (such as solution information) and a global calculation is performed, usually on a single PE that gathers the data. The results of the calculation are broadcast to all PEs, and are used in modifying the local data.

Then another *fork* phase begins, to be followed by another *join* and so on, until it is determined

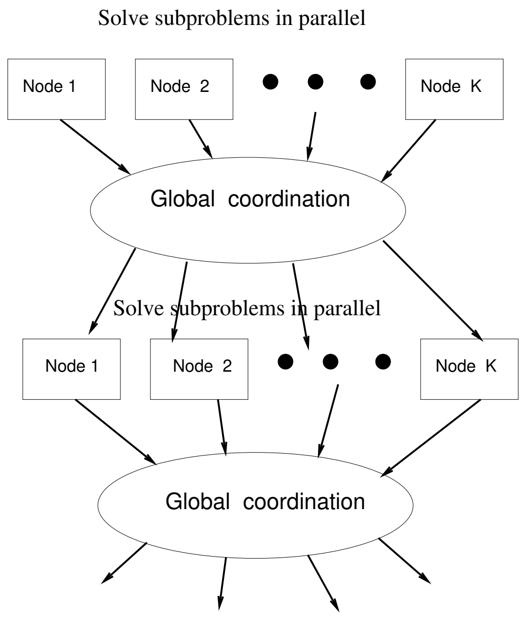

FIGURE 1: *The Fork–Join coordination protocol.*

(usually in the *join* phase) that global termination criteria have been met, and the computation concludes.

Depending on the communications protocol of the system, sometimes it is preferable to replicate, if possible, the *join* phase on all PEs, especially if the cost of broadcasting is high. This can also be determined during run-time, depending on the current load on the network in a time-sharing environment.

Also note that the protocol is general enough to be implementable on any distributed computing environment, such as a cluster of multiprocessors connected via Local Area Networks (LANs); what is required is an implementation of the message-passing routines based on primitives of the LAN protocol (such as TCP/IP).

The protocol is also known as *master/slave*; this reflects the case in which there is a dedicated *master* processor, that synchronizes the *slave* PEs via broadcasts, executes the *join* phase, and combines and redistributes data sent by the slaves.

Fork–Join coordination is quite popular in parallel optimization; it is employed by many decomposition algorithms such as: the Dantzig-Wolfe decomposition [19], the Schultz-Meyer shifted barrier method [92], the Parallel Variable Distribution [33], and Parallel Constraint Distribution [32] algorithms of Ferris and Mangasarian, and the Diagonal Quadratic Approximation method of Mulvey and Ruszczyński [71].

Methods that employ *fork–join* coordination are popular in other areas of scientific computing, as well. For instance, Kowalik and Kumar present in [58] such a method for solving tridiagonal linear systems; it consists of a careful organization of forward and backward steps of Gaussian elimination, the majority of which are carried out in parallel, in *fork* phases, and the rest are carried out in *join* phases.

1.3.3 Metrics for Parallel Performance

1.3.3.1 Execution time

As argued by Hennessy and Patterson [48], the most reliable performance metric for any computer system is *program execution time*; it incorporates several other measures of goodness, that, when taken independently, do not portray accurately the overall performance. These measures are: the quality of the instruction set; the capability of the compiler to produce code that keeps the ALU busy and minimizes stalling; the latency and bandwidth of the memory subsystem; the clock rate of the chip, and the rate of execution of floating-point operations.

When the object of benchmarking is a parallel system, program execution time can also mirror: the communication and synchronization costs, the latency and bandwidth of the interconnection network, manifest in, for example, the rate of servicing of requests to global memory, and the quality of the routing scheme for inter-processor message passing. The collection [23] presents an overview of parallel benchmarking, including a discussion of performance metrics and descriptions of benchmark suites in use today.

1.3.3.2 Speedup and Amdahl's Law

Let now T_p stand for the execution time of a program on a system with p processors. Let also T_1 be the execution time for the same program, on the same input data, on a single processor, and let T^* be the time it takes to execute the "best" known serial algorithm for the same problem on the same data set, on a single processor. We define the (absolute) *speedup* [79], [104], [11, chapter 1] as the ratio

$$s_p^* := \frac{T^*}{T_p} \tag{1}$$

the *relative speedup* s_p as the ratio

$$s_p := \frac{T_1}{T_p} \tag{2}$$

and the *efficiency* μ_p as

$$\mu_p := \frac{s_p^*}{p} \tag{3}$$

The value of μ_p is usually less that one, indicating efficiency loss due to a number of factors, such as: overhead introduced by communication (message packing and transfer), PE initialization (getting identification parameters, setting the communication topology and opening the channels), selection statements (switching PEs on and off to selectively operate on parts of data), duplication of operations, and PE suspension (forced idleness) due to load imbalance or message dependencies.

In certain cases the above definition (1) can be problematic: for problems such as Linear Programming there is no uncontested "best" serial algorithm; even though bounds in operations count are available, these are usually worst-case estimates and the actual performance may depend strongly on the particular data set and on the quality of the implementation. Moreover, definition (1) does not reflect well the "degree of parallelization" of the algorithm, as measured again its serial implementation; this degree is reflected better in the relative speedup. Thus, in the recent mathematical programming literature, one also finds measurements of parallel efficiency based on the relative speedup (Mangasarian with Deleone [21] and Ferris [33].) In our efficiency estimates for the ADI algorithm in section 7.5.1.3, we will also adopt this approach.

Observe now that $s_p \leq p$, since we can simulate an algorithm running on p processors as p serial copies running on a single processor, in roughly p-fold time. (Super-linear speedup can be observed when the memory available on a single PE is either less than the cumulative memory needed by p processors to solve the problem, or the result is too-frequent paging ("thrashing"). We exclude such "exotic" cases from the discussion.) If $s_p \simeq p$, then the parallel algorithm has *linear speedup*; this is clearly the best one can hope for. However, this is difficult to achieve, because of the limit on the amount of parallelism that can be extracted; this is described quantitatively by Amdahl's Law [5], which states that: "parallel speedup cannot exceed the inverse of the fraction of the residual sequential code." The relevant formula, stated in terms of efficiency, is

$$\mu_p = ([p-1]s+1)^{-1} \tag{4}$$

where s is the fraction of the residual sequential (:= non-parallelizable) code.

Thus, if $s = 0$, the speedup is almost linear, if we allow a small margin for communications overhead. Otherwise, if, say, 10% of the operations must be performed sequentially, then the maximum speedup is 10.

Over the years there have been re-evaluations [46] and extensions [103] of Amdahl's Law, in an effort to incorporate parallel environment factors, such as the process granularity, the synchronization overhead etc. Gustafson [8] also showed that for certain numerical problems the fraction of residual sequential code can be reduced as the problem size increases, and thus efficiency may improve with problem size.

However, the consensus is that "to exploit more parallelism, programmers should design new parallel algorithms, instead of parallelizing sequential algorithms" [104], especially since many large-scale sequential code libraries do not parallelize/vectorize well: a recent survey [98] reports that a flow simulation model, typical of numerical models in geophysical sciences, runs only three times faster on a vector Cray Y-MP that on an IBM RS-6000 workstation! The non-vectorizable part of the code is a library routine for solving a finite-difference discretization of a PDE, which accounts for about 30% of the total CPU operations. By Amdahl's Law, the speedup in this case cannot exceed 3, and this was corroborated by observations in practice.

Amdahl himself (as quoted in [79]) has recently remarked that the great majority of problems in technical computing exhibit $50 - 70\%$ concurrency available for exploitation by either a vector or a parallel architecture; only few exhibit concurrency of $85 - 94\%$, and this results from an intrinsic physical or logical decomposability of their structure. This suggests that the success of parallel decomposition algorithms can be largely attributed to the efficient way they exploit the underlying problem structure.

The design of efficient genuinely-parallel algorithms for the general case is a difficult task, especially if we consider the fact that efficiency may depend strongly on the architecture of the target parallel system; an efficient algorithm for a shared memory system may perform poorly on a distributed memory system, and vice versa.

There is ongoing theoretical work on deriving models for parallel computation, and on deriving efficient algorithms on these models. Success has been partial so far. We present here some of the basics of parallel complexity models, following the survey of Mayr [67], which also has a sizeable bibliography.

1.3.3.3 Complexity-Theoretic Parallel Computation

The basic computing model is the *Parallel Random Access Machine*, or *PRAM*, which consists of an unbounded number of computing cells, which are *Random Access Machines* [1, chapter 1], and an unbounded number of global, shared memory cells. A *RAM* consists of a read-only input tape, a write-only output tape, a stored program and registers. Any RAM can access any global memory cell in a single step. Each RAM executes its own program (which can be a copy of a global one) and synchronization is provided by a global clock.

Problems for which efficient parallel algorithms can be constructed belong to the complexity class $\mathcal{NC}$: for any such problem, there are positive constants c and k and a *PRAM* algorithm that requires $O(n^c)$ processors and runs in polylogarithmic ($:= O(\log^k n)$) time steps to solve a problem instance of size n.

Efficiency is defined in terms of the *relative* speedup, and is considered as a function of the input size of the problem. A parallel algorithm is called *optimal* if it runs in polylogarithmic time with efficiency $\Omega(1)$, and *efficient* if it runs in polylogarithmic time with efficiency $\Omega(\log^{-k} n)$, for some constant k.

Efficient parallel algorithms have been devised for some fundamental problems such as sorting and finding connected components and spanning forests in graphs. However, for depth-first search in a general graph, no $\mathcal{NC}$ algorithms are currently known, and it might be that the problem is inherently sequential. (For a detailed discussion see Mayr [67]).

1.4 The block-angular problem

1.4.1 Description

Our object of study is a convex minimization problem with linear constraints having a specific block structure in both the objective function and the constraint matrix. The convex block-angular problem, denoted as (**CBA**), is

$$\begin{array}{llllllll} \displaystyle\min_{x_{[1]}, x_{[2]}, \ldots, x_{[K]}} & f_{[1]}(x_{[1]}) & + & f_{[2]}(x_{[2]}) & + & \ldots & + & f_{[K]}(x_{[K]}) \\ & & & & & & & \\ \text{subject to} & A_{[1]}x_{[1]} & & & & & & = b_{[1]} \\ & & & A_{[2]}x_{[2]} & & & & = b_{[2]} \\ & & & & & \ddots & & \\ & & & & & & A_{[K]}x_{[K]} & = b_{[K]} \\ & D_{[1]}x_{[1]} & + & D_{[2]}x_{[2]} & + & \ldots & + & D_{[K]}x_{[K]} \le \mathtt{d} \\ & & & 0 \le x_{[i]} \le u_{[i]}, & i = 1, \ldots, K. & & & \end{array} \qquad (5)$$

In the above $x_{[i]}$ are the vectors of variables and $\sum_{i=1}^{K} f_{[i]}(\cdot)$ is the objective function. We take each component function in the objective $f_{[i]}$ to be finite-valued, convex and continuous. In many important applications each function $f_{[i]}$ is block quadratic (**LQ**) in the vector $x_{[i]}$, i.e.,

$$f_{[i]}(x_{[i]}) = c_{[i]}{}^T x_{[i]} + x_{[i]}{}^T Q_{[i]} x_{[i]}$$

in which $c_{[i]}$ is a real vector and $Q_{[i]}$ is a real symmetric positive semi-definite matrix. This includes the case where the objective is linear ($Q_{[i]} = 0$).

For $i = 1, \ldots, K$, $A_{[i]}$ is a $m_i \times n_i$ real matrix describing equality constraints on the corresponding block of variables $x_{[i]}$, with the vector $b_{[i]} \in \mathbb{R}^{m_i}$ serving as the right hand side. The vector $u_{[i]} \in \mathbb{R}^{n_i}$ is a component-wise upper bound on the levels of the activity $x_{[i]}$. If some components of the activity are allowed to grow without restriction, we take the corresponding components of the upper bound vector to be $+\infty$. The requirement $0 \leq x_{[i]}$ is not restrictive, since any lower bound on $x_{[i]}$ can be translated to zero by a suitable translation of variables. We may also assume that no upper bound on $x_{[i]}$ is zero; otherwise, we would set those variables to zero, delete the corresponding columns in the matrices, and consider the reduced problem.

The matrix $A_{[i]}$ and the vectors $b_{[i]}$ and $u_{[i]}$ define the *block* constraints for block i.

Each real matrix $D_{[i]}$ is of size $m_0 \times n_i$; the matrix $\left[D_{[1]} \ldots D_{[K]}\right]$ and the shared resource vector $\mathtt{d} \in \mathbb{R}^{m_0}$ define the *coupling* constraints that the block variables $x_{[i]}$, $i = 1, \ldots, K$, must jointly meet. We can think of $D_{[i]} x_{[i]}$ as the amount of $\mathtt{d}$ that activity $x_{[i]}$ is consuming.

The constraint matrix thus has a special structure: block variables interact only in the coupling constraints; if relatively few of these constraints are active at optimality, then the blocks are loosely coupled. If there were no coupling constraints, (**CBA**) would be fully decomposable by block, since the objective function is block-separable and each block of the constraints involves only one block of variables. Then any solution made up of the concatenation of optimal vectors for the block problems would be optimal for (**CBA**) and vice versa. It is the presence of the coupling constraints that makes the problem hard.

When convenient, we can assume without loss of generality that the coupling constraints are equalities: this can be accomplished by the introduction of a block of non-negative slack variables $x_{[K+1]}$. We would take $f_{[K+1]} \equiv 0$, $u_{[K+1]} = +\infty$, and the matrix $D_{[K+1]}$ to be the identity on $\mathbb{R}^{m_0 \times m_0}$. Then

$$D_{[1]} x_{[1]} + D_{[2]} x_{[2]} + \ldots + D_{[K]} x_{[K]} + x_{[K+1]} = \mathtt{d}, \quad 0 \leq x_{[i]} \leq u_{[i]}, \; i = 1, \ldots, K+1$$

We now describe two real-world optimization problems that give rise to (**CBA**) problems. The first one, multicommodity network flow, occurs in scheduling and transportation problems. The second one, scenario analysis, arises in stochastic optimization for multi-period planning under uncertainty.

1.4.2 Multicommodity Network Flow (MC)

In modeling urban traffic, railroads, communications, military logistics and multiproduct production and distribution, the following problem arises: K commodities flow over the same network, or over networks that share arcs. The objective is to find a flow of minimal cost such that, for each commodity:

(i) Supply and demand is met.

(ii) Flow is conserved at transshipment nodes.

(iii) Arc capacities for that commodity are not exceeded.

These restrictions correspond to the block constraints. The coupling constraints correspond to the restriction:

(iv) The aggregate flow on a set of arcs shared by the commodities does not exceed some joint capacity.

This problem can be modeled as (**CBA**) in the following way: the matrices $A_{[i]}$ and the vectors $b_{[i]}$ are chosen to reflect (i) and (ii), the vectors $u_{[i]}$ are the upper bounds as dictated by (iii), and the matrices $D_{[i]}$ and vector $\mathtt{d}$ are chosen to reflect (iv).

Each $A_{[i]}$ is thus the node-arc incidence matrix of a topological network; rows correspond to nodes and columns to arcs; each column has exactly two non-zero entries, one of which is $+1$ (for an outgoing arc), and the other is -1 (for an incoming arc). The vector $b_{[i]}$ represents node divergences: it has a positive entry for a supply node, a negative entry for a demand node, and a zero entry for a transshipment node. Each matrix $D_{[i]}$ consists of columns with one $+1$ entry and of zero columns; nonzero columns correspond to arcs that appear in the linking constraints: thus, $D_{[i]}\ (j,k)$ is 1 if the k-th arc for the i-th commodity participates in linking constraint j, and is 0 otherwise. All $D_{[i]}$ matrices have the same number of rows, which is the number of linking constraints. From the above description we see that the constraint matrices $A_{[i]}$ and $D_{[i]}$ for (**MC**) can be very sparse.

The multicommodity network flow problem has been studied extensively, and specialized algorithms have been developed for it over the years. A good coverage of these is provided in the survey articles by Kennington [55] and Assad [6], and in the books by Kennigton and Helgason [56] and by

Ahuja, Magnanti and Orlin [2].

1.4.3 Scenario Analysis

In some cases of modeling there is uncertainty about the value of some of the parameters of the system. In certain situations one can assign "reasonable" values to such parameters; otherwise, one has to include the uncertainty into the model, and solve a stochastic optimization problem. A way of modeling uncertainty is to treat the parameters as random variables, and assume that they come from a probability space (Ω, P), where Ω is the set of possible realizations and P is the associated probability density. Then one is interested in solving the problem

$$\begin{array}{rl} \text{minimize} & \mathbf{E}\{f(x,\xi)\} := \int\limits_{\Omega} f(x,\xi)\; dP(\xi) \\ \text{subject to} & x \in C(\xi) \end{array}$$

i.e. minimizing the expected value of the utility function f over all realizations of the parameters x that belong to the feasible set $C(\xi)$.

In many cases the complete structure of (Ω, P) is not known, because only limited information is available. Then one can resort to *scenario analysis*, i.e. model uncertainty by a few discrete scenarios $\mathbf{S} := \{\mathbf{s_1}, \ldots, \mathbf{s_L}\}$ and then solve the problem

$$\begin{array}{rl} \text{minimize} & \sum\limits_{l=1}^{L} p_l\; f(x_l, s_l) \\ \text{subject to} & x_l \in C_l, \quad s_l \in \mathbf{S} \end{array}$$

in which p_l is the probability that scenario s_l occurs, and C_l is the associated feasible set.

In the case when scenarios correspond to decisions taken for multi-stage planning, the problem can have the following specialized form: each x_l represents a sequence of decisions at stages $t = 1, \ldots, T$, $x_l = \left(x_l^1, x_l^2, \ldots, x_l^T\right)$, and the constraint sets C_l link decisions over two consecutive stages

$$D_l^{t-1} x_l^{t-1} + H_l^t x_l^t = b_l^t, \quad x_l^t \geq 0, \quad t = 1, \ldots, T \tag{6}$$

This is a fully decomposable problem, since both the objective and the block constraints (6) decompose per scenario. However, the modeling of limited information introduces additional constraints that link the scenario subproblems and give the problem the structure of (**CBA**), except that, in the most direct formulation, the coupling constraints are equalities.

For any scenario s_l, at any stage t, information is available for data only up to that stage, i.e. only the scenario data $d_l^\tau := (D_l^\tau, H_l^\tau, b_l^\tau)$, $\tau = 1, \ldots, t$, are known. Then, if scenarios s_i and s_j

have common past and present stages, i.e. if $d_i^\tau = d_j^\tau$, $\tau = 1, \ldots, t$, then they must make the same decision at stage t, i.e. $x_i^t = x_j^t$.

This is the *non-anticipativity* requirement. If we let $\mathcal{A}(\rangle, \sqcup)$ denote the set of all scenarios that make identical decisions with scenario i up to stage t, we can formulate the scenario analysis problem as

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{l=1}^{L} p_l \ f(x_l, s_l) \\ \text{subject to} & x_l \in C_l, \quad l = 1, \ldots, L, \\ & x_i^\tau = x_j^\tau \quad \text{for all } j \in \mathcal{A}(\rangle, \sqcup), \quad | = \infty, \ldots, \mathcal{L}, \quad \sqcup = \infty, \ldots, \mathcal{T} \end{array} \tag{7}$$

The non-anticipativity constraints can also be modeled as follows [70]: suppose that at stage t scenarios $s_1, s_2, \ldots, s_k$ have common past and present data. Then non-anticipativity is enforced by coupling constraints of the type

$$\begin{array}{rcl} x_1^t & = & x_2^t \\ x_2^t & = & x_3^t \\ & \vdots & \\ x_{k-1}^t & = & x_k^t \end{array}$$

so that each constraint links only two blocks. This formulation results in coupling matrices that are quite sparse: for a three-stage problem, Figure 2 shows the decision tree, and Figure 3 shows the structure of the constraint matrix.

Our presentation of the scenario analysis problem is based on material in Wets [102] and Ruszczyński [89]. For a theoretical treatment of Stochastic Linear Programming one may consult the book by Kall [53].

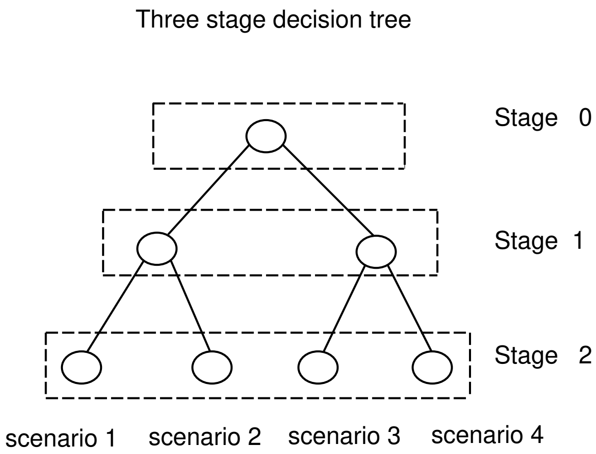

FIGURE 2: *A three-stage decision problem.*

$$
\begin{bmatrix}
x & & & & & & & & & & & \\
x & x & & & & & & & & & & \\
 & x & x & & & & & & & & & \\
 & & & x & & & & & & & & \\
 & & & x & x & & & & & & & \\
 & & & & x & x & & & & & & \\
 & & & & & & x & & & & & \\
 & & & & & & x & x & & & & \\
 & & & & & & & x & x & & & \\
 & & & & & & & & & x & & \\
 & & & & & & & & & x & x & \\
 & & & & & & & & & & x & x \\
x & & & x & & & & & & & & \\
 & & & x & & & x & & & & & \\
 & & & & & & x & & & x & & \\
 & x & & & x & & & & & & & \\
 & & & & & & & x & & & x &
\end{bmatrix}
$$

FIGURE 3: *The constraint matrix for the three-stage decision problem: x marks a possibly non-zero block. The last five rows are the non-anticipativity constraints.*

1.5 Previous work

Following Lasdon [59], we classify methods for solving large, structured mathematical problems, such as (**CBA**), into direct and indirect. Direct methods specialize an existing algorithm to take advantage of problem structure. For instance, when applying the simplex method to the linear (**CBA**) problem, procedures such as generalized upper bounding and compact basis triangularization allow for efficient manipulation of the basis inverse. Indirect methods essentially follow the *fork-join* protocol, by decomposing the original problem into decoupled first-level subproblems and then solving a master coordination problem. Eckstein [27] surveys parallel implementations to date.

The Dantzig-Wolfe [19] and Benders [7] decomposition are two classic indirect methods. When applied to (**CBA**), both can be implemented in parallel, since each block can be assigned to a PE. In the *fork* phase each PE solves the associated subproblem; then, in the *join*, a PE collects subproblem solution information, partially redefines the master problem, by adding a column or an objective cut, and solves it. This solution is then used to create a new set of resource prices or allocations which are passed to the subproblems, and the cycle repeats.

Another family of indirect methods for (**CBA**) employs penalty functions. The idea is to eliminate the coupling constraints by placing them in the objective function via a suitable penalty term. The resulting nonseparable and nonlinear penalty problem is solved iteratively, by an algorithm that induces separability in the penalty objective. Then the problem decomposes into independent block subproblems, the solution of which is used to redefine the penalty in the next iteration. In this general framework one can place the logarithmic shifted barrier algorithm of Schultz and Meyer [91], [92], the smooth linear-quadratic penalty method of Zenios, Pinar and Dembo [107], [81], and the Diagonal Quadratic Approximation method of Mulvey and Ruszczyński [71], [89].

Several other decomposition methods take advantage of the matrix structure of the problem. Zenios [106] specializes the row-action method of Censor and Lent [16] to quadratic (**MC**) ; the algorithm iterates on each constraint in an almost cycling fashion, adjusting primal and dual variables to preserve complementarity and constraint satisfaction. Resource allocation methods maintain a feasible allocation of the shared resource among the blocks; for a given allocation, a subproblem is solved for each block, and then a master problem is solved, in order to determine an improved allocation. The master problem is non-smooth, and can be solved by a subgradient method (Geoffrion [39], Kennington and Helgason [56, chapter 6]), or a bundle method (Medhi [68], Ferris and Horn [31], De Leone et al. [20].)

In coarse-grain implementations of subproblem/master algorithms, the solution of the master problem on a single processor is a potential bottleneck that can reduce parallel efficiency. We are thus interested in methods that generate master problems which are of small size and/or have an explicit solution, that can be computed in parallel.

The Parallel Constraint Distribution method of Ferris and Mangasarian [30], [32] displays such attractive features; it distributes the constraints among the PEs and modifies each subproblem objective function with augmented Lagrangian terms from other PEs. New Lagrangian information, arising from the solution of the subproblems, is aggregated on a master PE, and the cycle is repeated. The Parallel Variable Distribution method by the same authors [33], distributes the variables among the PEs, so that each PE has primary responsibility for updating its own block of variables while allowing the remaining ones to change in a restricted fashion. This process is carried out in parallel, and is followed by a master problem of controlled size, in which the affine hull of (some of) the points computed in the *fork* phase is searched for an optimal point.

The Alternating Directions method, which we begin to study in the next chapter, is also such that the *join* phase is of a very simple computational nature, and can be calculated fast on the CM-5.

The trade-off between simple and complex coordination is that an algorithm with a complex coordination phase may be able to utilize better the subproblem information, and thus converge in fewer iterations than a scheme with simple coordination. (For further discussion see De Leone et al. [22]. See also Ho et al. [50], [43] for an improvement of the parallel performance of Dantzig-Wolfe decomposition via load-balancing schemes).

1.6 Facts from convex analysis

In this section we group some basic definitions and properties of convex functions and sets. We follow Robinson [83] in our presentation. The reader may also consult Rockafellar [85]. We are concerned exclusively with sets and functions defined on real, finite-dimensional Hilbert spaces.

Definition 1.6.1 *A set $C \subset \mathbb{R}^n$ is* CONVEX *if for each x and y in C and each $\lambda \in (0,1)$, one has that $(1-\lambda)x + \lambda y \in C$.*

Definition 1.6.2 *The* INDICATOR *function of a convex set C, $\psi(\cdot \mid C)$, is defined by*

$$\psi(x \mid C) := \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{otherwise} \end{cases}$$

Definition 1.6.3 *A set $H \subset \mathbb{R}^n$ of the form $H = \{x \in \mathbb{R}^n \mid c^T x \leq \alpha\}$ for some vector $c \in \mathbb{R}^n$ and a real scalar α is called a* CLOSED HALF-SPACE.

Definition 1.6.4 *A set is called* POLYHEDRAL *if it is the intersection of a finite number of closed half-spaces.*

Definition 1.6.5 *A real function is called* EXTENDED-VALUED *if it can explicitly assume the values $+\infty$ and $-\infty$.*

Definition 1.6.6 *The* EPIGRAPH *of an extended-valued function f is the set in $\mathbb{R}^{n+1}$*

$$\text{epi } f := \{(x, \mu) \mid x \in X, \mu \in \mathbb{R}, f(x) \leq \mu\}$$

Definition 1.6.7 *An extended-valued function is called* CONVEX *if its epigraph is a convex set.*

Definition 1.6.8 *A function is called* CLOSED *if its epigraph is a closed set.*

Definition 1.6.9 *The* EFFECTIVE DOMAIN *of an extended-valued function f is the set*

$$\operatorname{dom} f := \{x \in X \mid f(x) < +\infty\}$$

The effective domain of a convex function is a convex set.

Definition 1.6.10 *A convex function f is called* PROPER *if it is not identically $+\infty$ and it never assumes the value $-\infty$.*

Definition 1.6.11 *Let f be a proper, convex function $\mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$, and let a be a real number. The* LEVEL SET *Λ_a is defined as*

$$\Lambda_a := \{x \in X \mid f(x) \leq a\}$$

For f a proper, convex function, a nonempty level set Λ_a is convex.

For a proper function f we also have an alternative characterization of convexity.

Proposition 1.6.12 *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$. If, for each x and $y \in \mathbb{R}^n$, $x \neq y$, and each $\lambda \in (0, 1)$*

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$$

then f is convex, while in case

$$f((1-\lambda)x + \lambda y) < (1-\lambda)f(x) + \lambda f(y)$$

then f is called STRICTLY CONVEX*, and finally if*

$$f\left((1-\lambda)x+\lambda y\right) \le (1-\lambda)f(x)+\lambda f(y) - \delta\lambda(1-\lambda)\left\|x-y\right\|_2^2$$

for some $\delta > 0$, then f is called STRONGLY CONVEX *with modulus δ.*

Definition 1.6.13 *A function f is* LOWER SEMICONTINUOUS *at x if*

$$f(x) \le \liminf_{y\to x} f(y) = \lim_{\epsilon\to 0}\left[\inf\{f(y) \mid 0 < \|y-x\| < \epsilon\}\right]$$

We then have the following important characterization theorem.

Theorem 1.6.14 *Let f be an extended-valued real convex function. Then, the following are equivalent :*

(i) f is lower semicontinuous on its domain.

(ii) the level sets Λ_a are closed, for all real a.

(iii) f is closed.

The following theorem, due to Bolzano and Weierstass, is very useful in applications.

Theorem 1.6.15 *Let $f : X \longrightarrow \mathbb{R}$ be lower semicontinuous on a compact set $K \subset X$. Then, there exists $x_0 \in K$ such that $f(x_0) = \inf_{y\in K} f(y)$.*

We now define the conjugate function and the subdifferential set.

Definition 1.6.16 *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{\pm\infty\}$. The* CONJUGATE *of f is the function f^*, defined for each $y \in \mathbb{R}^n$ by*

$$f^*(y) = \sup_{x\,\in\,\mathbb{R}^n} \left\{\ y^T x - f(x)\ \right\}$$

The conjugate f^* is a closed, convex function.

Definition 1.6.17 *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{\pm\infty\}$ be a convex function, and let $x_0 \in \mathbb{R}^n$. A vector $y \in \mathbb{R}^n$ is a* SUBGRADIENT *of f at x_0 if*

$$f(x) \ge f(x_0) + y^T(x - x_0), \quad \forall x \in \mathbb{R}^n$$

The SUBDIFFERENTIAL *set of f at x_0, ∂f_{x_0}, is the set of all subgradients of f at x_0.*

For all $x_0 \in \mathbb{R}^n$, ∂f_{x_0} is a closed, convex set.

We conclude by defining the linear complementarity problem.

Definition 1.6.18 (LCP) *Let LCP (q, M) denote the following problem: given a vector $q \in \mathbb{R}^n$ and an $n \times n$ real matrix M, find a non-negative vector $x \in \mathbb{R}^n$ such that $Mx + q \ge 0$ and $x^T(Mx + q) = 0$.*

Chapter 2

The Alternating Directions Method for Optimization

2.1 Overview

In this chapter we study the Alternating Directions method (ADI) as applied to the minimization of the sum of two convex functions subject to linear equations

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n,\, z \in \mathbb{R}^m} & G_1(x) + G_2(z) \\ \text{subject to} & Ax + b = Bz \end{array}$$

In subsequent chapters we show how to embed (**CBA**) in this general problem in three ways, with each one producing a distinct iterative scheme.

We introduce the method in section 2.2. Section 2.3 contains the main result, the convergence theorem 2.3.1. This theorem extends existing results, by relaxing the assumptions on the functions and the constraints and also by treating an extended version of the method.

A characteristic of the method is convergence to the optimal value of the objective function even if the primal iterates do not converge. (This is a common occurrence in optimization algorithms; e.g. the subgradient method for unconstrained nondifferentiable optimization in Polyak [82, section 5.3.2].) We present an example illustrating this mode of convergence for the ADI method in section 2.4.

Then we provide, in section 2.5, a sequence of corollaries that deal with special cases of interest and with an implementable stopping criterion. Finally, in section 2.6 we present two variations of the ADI method, one that employs relaxation of the primal iterates and the *Peaceman–Rachford* variant. As we will demonstrate in subsequent chapters, these two variants are closely related when applied to the ADI methods we derive for the block-angular problem (**CBA**).

The ADI method for optimization has been studied extensively and sufficient conditions for convergence have been obtained by using either the theory of maximal monotone operators (Lions and Mercier [62], Gabay [37], Eckstein and Bertsekas [25], [28]) or the theory of saddle-points of Lagrangian functions (Glowinski with Marrocco, Fortin and Le Tallec [42], [34], [41]).

The Progressive Hedging algorithm of Rockafellar and Wets [88, 102] is a version of the ADI algorithm applied to scenario analysis problems. When used to solve partial differential equations, the ADI method is usually called the *Douglas–Rachford* method, because it corresponds to the method in [24] by these two authors. This viewpoint is discussed in detail in [37], [41], [62], [25].

Brézis [14] provides a good textbook level introduction to the general theory of maximal monotone operators. Robinson [84] presents the theory from an optimization viewpoint; the connection is that fixed-point iteration on the resolvent of the subdifferential operator is equivalent to the proximal point algorithm. (See also Rockafellar [86], [87] on this.) For a general theoretical discussion of ADI and other splitting methods for the solution of partial differential equations, see Marchuk [66]; Johnsson et al. [52] discuss parallel implementations. In [29] Eckstein and Ferris construct Douglas–Rachford and Peaceman–Rachford operator splitting methods for the monotone linear complementarity problem. In his Ph.D. thesis [25] Eckstein demonstrates that, for the minimization problem shown above, some other decomposition algorithms, among them the algorithm of Han and Lou [47], Spingarn's Method of Partial Inverses [93], [94] and Golshtein's Block Method of Convex Programming [44], [45], are instances of the Douglas–Rachford algorithm. He then proceeds to derive ADI methods for monotropic programming; in [26] he discusses their *fine-grain* implementation on the Connection Machine CM–2 parallel computer. Fukushima [36] constructs an ADI method for the dual, rather than the primal problem. Cheng and Teboulle [17] present a modified ADI method, in which quadratic proximal terms replace the augmented Lagrangian penalty terms.

In the existing literature convergence of the ADI algorithm for optimization is proved under rather strong assumptions on the problem: strict convexity and differentiability of G_1 and growth of G_1 faster than linear at infinity [34], [41]. For the finite-dimensional case, A is assumed to have full column rank [25], [28] or G_1 is assumed to have compact level sets [11, chapter 3]; for the dual algorithm in [36], both primal and dual problems are assumed to be feasible and the solution set of the primal is assumed to be bounded. For the finite-dimensional setting, we have been able to relax these assumptions: for instance, we require only existence, not uniqueness, of minimizers for the original problem and the two subproblems.

ADI methods can be constructed also for the Linear Complementarity Problem LCP(M, q) where the matrix M is symmetric positive semidefinite. Then the LCP is equivalent to the quadratic problem [64]

$$\min_{x \geq 0} \quad \frac{1}{2} x^T M x + q^T x \tag{8}$$

Let M be split as $M = B + C$. After introducing an additional vector of variables z, we have the equivalent problem with block-separable objective

$$\begin{array}{ll} \displaystyle\min_{x \geq 0,\ z \geq 0} & \displaystyle\frac{1}{2} x^T B x + \frac{1}{2} q^T x + \frac{1}{2} z^T C z + \frac{1}{2} q^T z \\ \text{subject to} & x = z \end{array}$$

which is in the form of the general problem for ADI. The convergence theory for ADI, presented in the following sections, requires that both quadratic terms in the objective be convex, i.e. that both B and C matrices be positive semidefinite. In comparison, the "regular" splittings for the LCP in (8) (see [65], [18, chapter 5]) require that the matrix $B - C$ be positive definite.

2.2 Description of the algorithm

We want to solve the following linearly-constrained convex problem

$$\begin{array}{ll} \displaystyle\min_{x \in \mathbb{R}^n,\, z \in \mathbb{R}^m} & G_1(x) + G_2(z) \\ \text{subject to} & Ax + b = Bz \end{array} \tag{9}$$

where $G_1 : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, $G_2 : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are extended-real valued, proper, closed, convex functions, $b \in \mathbb{R}^l$, $A : \mathbb{R}^n \to \mathbb{R}^l$ is a nonzero $l \times n$ matrix, and $B : \mathbb{R}^m \to \mathbb{R}^l$ is a nonzero $l \times m$ matrix.

The Lagrangian function associated with problem (9) is

$$L_0(x, z, p) := G_1(x) + G_2(z) + p^T (Ax + b - Bz) \tag{10}$$

in which $p \in \mathbb{R}^l$ is a tentative Lagrange multiplier vector. From standard duality theory, if the Lagrangian admits a saddle-point at (x^*, z^*, p^*), i.e., if

$$L_0(x^*, z^*, p^*) = \min_{x,\, z} \ \max_{p} \quad L_0(x, z, p)$$

then (x^*, z^*) is a primal solution of problem (9) and p^* is an associated dual multiplier. We make the following basic assumption.

Assumption 2.2.1 *Problem (9) admits a lagrangean saddlepoint.*

Thus, a way of solving (9) is by finding a saddle-point of the Lagrangian [101], or, for reasons of computational stability [9], of the augmented Lagrangian

$$L_\lambda\ (x, z, p) := G_1(x) + G_2(z) + p^T(Ax + b - Bz) + \frac{\lambda}{2}\|Ax + b - Bz\|_2^2 \tag{11}$$

in which λ, the penalty parameter, is any positive scalar. Both Lagrangians have the same set of saddle-points.

An iteration of the method of multipliers consists of taking a minimization step in the primal space, followed by a steepest-ascent-type update of the multipliers p

$$\begin{aligned} (x^{t+1}, z^{t+1}) &\in \underset{x,\, z}{\operatorname{argmin}}\ \ L_\lambda\ (x, z, p^t) \\ p^{t+1} &= p^t + \lambda\ \nabla_p\ L_\lambda\ (x^{t+1}, z^{t+1}, p^t) \end{aligned}$$

From the viewpoint of *fork–join* computation, if the original objective $G_1(x) + G_2(z)$ and the linear constraints $Ax+b-Bz$ are block-separable, one is interested in having the primal minimization problem decompose in smaller subproblems that can be solved independently in parallel. However, in the method of multipliers the quadratic penalty term $\|Ax + b - Bz\|_2^2$ is generally not separable, because it contains cross-terms. In some cases this drawback is not present in the Alternating Directions method, in which the primal minimization step is done in a block Gauss-Seidel fashion, first over the x and then over the z variables

$$\begin{aligned} x^{t+1} &\in \underset{x}{\operatorname{argmin}}\ \ L_\lambda\ (x, z^t, p^t) \\ z^{t+1} &\in \underset{z}{\operatorname{argmin}}\ \ L_\lambda\ (x^{t+1}, z, p^t) \\ p^{t+1} &= p^t + \lambda\ \nabla_p\ L_\lambda\ (x^{t+1}, z^{t+1}, p^t) \end{aligned}$$

Arbitrary vectors p^0 and z^0 are chosen as starting point, and, in the simplest case, the scalar penalty $\lambda > 0$ is fixed throughout the iterations. The method works towards achieving optimality in both the primal and the dual space, by taking alternating steps in each. For strongly convex problems satisfying a Lipschitz condition it can be proven that the rate of convergence is linear [62]. In order to speed convergence, we vary the penalty λ per iteration, by a heuristic procedure, that may be tuned to the application at hand. An even more general approach is to use a separate value of λ for each linear constraint, also varying per iteration. In the most general case, we allow linear transformations of the constraints, by employing a square symmetric positive definite matrix H^t in the constraint term, which is ultimately fixed.

This results in the following extension of the ADI algorithm for problem (9).

$$\begin{aligned}
x^{t+1} &\in \underset{x}{\operatorname{argmin}} \quad G_1(x) + {p^t}^T Ax + \frac{1}{2}(Ax + b - Bz^t)^T H^t (Ax + b - Bz^t) && (12)\\
z^{t+1} &\in \underset{z}{\operatorname{argmin}} \quad G_2(z) - {p^t}^T Bz + \frac{1}{2}(Ax^{t+1} + b - Bz)^T H^t (Ax^{t+1} + b - Bz) && (13)\\
p^{t+1} &= p^t + H^t \left(Ax^{t+1} + b - Bz^{t+1}\right) && (14)
\end{aligned}$$

The minimization in the subproblems (12) and (13) takes place in the effective domains of G_1 and G_2. These are nonempty (since both functions are proper) convex sets. Thus the subproblems are feasible. In order to have the algorithm well-defined, we need to guarantee the existence of minimizers, i.e. we need to make the following assumption.

Assumption 2.2.2 *Subproblems (12) and (13) are solvable.*

Thus, the values $G_1(x^{t+1})$ and $G_2(z^{t+1})$ are finite, for all $t \geq 0$. Notice that we do not require uniqueness of the minimizers in the subproblems. This would require stronger assumptions on the problem, such as G_1 and G_2 being strictly convex, or the matrices A and B having full column rank.

2.3 The convergence theorem

We now state and prove the convergence theorem for the extended ADI method. Our proof is modeled after [11, chapter 3] and [41, chapter 3].

Theorem 2.3.1 (Convergence of ADI) *Let assumptions 2.2.1 and 2.2.2 hold, and let $\{(x^t, z^t, p^t)\}$ be a sequence of iterates produced by the ADI algorithm (12)–(14). Let the symmetric positive definite matrices $\{H^t\}$ be ultimately fixed. Then*

(i) The sequence $\{(Ax^t, Bz^t)\}$ converges, and the limit satisfies the constraints of problem (9).

(ii) The sequence $\{G_1(x^t) + G_2(z^t)\}$ converges to the optimal value of the objective function for problem (9).

(iii) The sequence $\{p^t\}$ converges to an optimal dual multiplier for problem (9).

(iv) Any minimizers of subproblems of the form (12) and (13) in which p^t, Ax^{t+1} and Bz^t are fixed at their limit values and H^t is an arbitrary symmetric positive definite matrix, are optimal for problem (9).

A point that perhaps needs clarification is the statement in part (iii) regarding "an optimal dual multiplier for problem (9)", i.e., we are considering a dual multiplier independently of a primal solution. This is a property of saddle-points: dual multipliers are associated with the minimization problem, not with specific primal solutions, as we will show in the following lemma.

Lemma 2.3.2 *Let X^0 be a nonempty set in $\mathbb{R}^n$, and define on it the functions $\theta : X^0 \to \mathbb{R}$, $g : X^0 \to \mathbb{R}^m$. Suppose that (x_1, u_1) satisfies saddle-point conditions with respect to θ and g*

$$\theta(x_1) + u^T g(x_1) \leq \theta(x_1) + {u_1}^T g(x_1) \leq \theta(x) + {u_1}^T g(x) \quad \forall u \in \mathbb{R}^m,\ \forall\ x \in X^0\ . \tag{15}$$

If (x_2, u_2) also satisfies these conditions, then so do (x_1, u_2) and (x_2, u_1).

Proof: By arguments similar to those in [63, section 5.3.1], both x_1 and x_2 solve

$$\begin{array}{cc} \min\limits_{x \in X^0} & \theta(x) \\ \text{subject to} & g(x) = 0 \end{array}$$

and therefore $\theta(x_1) = \theta(x_2)$ and $g(x_1) = g(x_2) = 0$. Thus

$$\begin{aligned} \theta(x_2) + {u_1}^T g(x_2) = \theta(x_2) = \theta(x_1) &= \\ \theta(x_1) + {u_1}^T g(x_1) &\leq \theta(x) + {u_1}^T g(x) \ \ \forall\ x \in X^0 \end{aligned} \tag{16}$$

On the other hand, since $g(x_2) = 0$, for any $u \in \mathbb{R}^m$ we have $u^T g(x_2) = 0$, and thus

$$\theta(x_2) + u^T g(x_2) \leq \theta(x_2) = \theta(x_2) + {u_1}^T g(x_2) \ \ \forall u \in \mathbb{R}^m \tag{17}$$

Combining (17) and (16) we get

$$\theta(x_2) + u^T g(x_2) \leq \theta(x_2) + {u_1}^T g(x_2) \leq \theta(x) + {u_1}^T g(x) \quad \forall u \in \mathbb{R}^m,\ \forall\ x \in X^0$$

which shows that (x_2, u_1) is a saddle-point. The proof for (x_1, u_2) follows by symmetry. ∎

In the case of problem (9), we take the set X^0 in the lemma above to be the effective domain of $G_1 + G_2$ (product of the respective effective domains). By assumption 2.2.1 this set is nonempty. Thus, for the application of the lemma, we take the function θ to be the restriction of $G_1 + G_2$ to that set. Also, since the constraints for problem (9) are linear, each optimal point corresponds to a saddle-point.

In the proof of theorem 2.3.1 we will also use the following linearization lemma, which specializes [15, theorem 2.3].

Lemma 2.3.3 *Let X^0 be a nonempty, convex subset of $\mathbb{R}^n$. Suppose $J : X^0 \to \mathbb{R}$ is a function of the form $J = J_1 + J_2$, where J_1 and J_2 are convex functions and J_2 is differentiable on X^0. Then, $\bar{x}$ is a minimum for J if and only if*

$$J_1(x) - J_1(\bar{x}) + \nabla J_2(\bar{x})^T (x - \bar{x}) \geq 0, \quad \forall x \in X^0$$

In order to prove the convergence theorem 2.3.1, we state and prove a collection of lemmas. We begin by proving, in lemma 2.3.4, that the ADI iterates $\{(Ax^t, Bz^t, p^t)\}$ are bounded. In lemma 2.3.5 we show that the sequence $\{G_1(x^t) + G_2(z^t)\}$ converges to the optimal value of the objective function for problem (9) and that, in the limit, $\{(Ax^t, Bz^t)\}$ satisfies the constraints of problem (9). In lemma 2.3.6 we show that the sequences $\{Ax^t\}$, $\{Bz^t\}$ and $\{p^t\}$ converge, and that the limit point of $\{p^t\}$ is an optimal dual multiplier for problem (9). Finally, lemma 2.3.7 shows how to employ the limit values of $\{Ax^t\}$, $\{Bz^t\}$ and $\{p^t\}$ as proximal terms in order to obtain a primal optimal vector for problem (9) by solving two minimization subproblems.

Lemma 2.3.4 *Let $\{(x^t, z^t, p^t)\}$ be a sequence of iterates produced by the ADI algorithm for problem (9). Let the symmetric positive definite matrices $\{H^t\}$ be ultimately fixed. Then the sequences $\{Ax^t\}$, $\{Bz^t\}$ and $\{p^t\}$ are bounded.*

Proof: Applying the linearization lemma 2.3.3 to subproblem (12) with the correspondences

$$\begin{aligned} J_1(x) &= G_1(x), \\ J_2(x) &= {p^t}^T Ax + \frac{1}{2}\,(Ax + b - Bz^t)^T H^t (Ax + b - Bz^t) \end{aligned}$$

we get, after using the symmetry of H^t

$$[p^t + H^t(Ax^{t+1} + b - Bz^t)]^T A(x^{t+1} - x) \leq G_1(x) - G_1(x^{t+1}) \;\; \forall x \in \mathbb{R}^n \tag{18}$$

This expression can be simplified, by using the multiplier update (14). We obtain

$$[p^{t+1} + H^t B(z^{t+1} - z^t)]^T A(x^{t+1} - x) \leq G_1(x) - G_1(x^{t+1}) \;\; \forall x \in \mathbb{R}^n \tag{19}$$

Applying the linearization lemma to subproblem (13) with the correspondences

$$\begin{aligned} J_1(z) &= G_2(z), \\ J_2(z) &= -{p^t}^T Bz + \frac{1}{2}\,(Ax^{t+1} + b - Bz)^T H^t (Ax^{t+1} + b - Bz) \end{aligned}$$

we get in a similar fashion

$$-\left[p^t + H^t(Ax^{t+1} + b - Bz^{t+1})\right]^T B(z^{t+1} - z) \le G_2(z) - G_2(z^{t+1}) \quad \forall z \in \mathbb{R}^m \tag{20}$$

which can be simplified to

$$-{p^{t+1}}^T B(z^{t+1} - z) \le G_2(z) - G_2(z^{t+1}) \quad \forall z \in \mathbb{R}^m \tag{21}$$

Let now $(x^*,\ z^*)$ be any primal optimal vectors and let p^* be any optimal dual vector for problem (9). Then

$$Ax^* + b = Bz^* \tag{22}$$

and also

$$G_1(x^*) + G_2(z^*) = L_0(x^*, z^*, p^*) = \min_{x,\,z} \quad L_0(x, z, p^*)$$

that is

$$G_1(x^*) + G_2(z^*) \le G_1(x) + G_2(z) + {p^*}^T(Ax + b - Bz) \quad \forall x \in \mathbb{R}^n,\ \forall z \in \mathbb{R}^m$$

For the primal ADI iterates $\{x^t\}$ and $\{z^t\}$ this implies

$$G_1(x^*) + G_2(z^*) \le G_1(x^{t+1}) + G_2(z^{t+1}) + {p^*}^T(Ax^{t+1} + b - Bz^{t+1}) \tag{23}$$

or, after taking (14) into account,

$$G_1(x^*) + G_2(z^*) \le G_1(x^{t+1}) + G_2(z^{t+1}) + {p^*}^T(H^t)^{-1}(p^{t+1} - p^t) \tag{24}$$

On the other hand, substituting x^* for x in (19) we obtain

$$\left[p^{t+1} + H^t B(z^{t+1} - z^t)\right]^T A(x^{t+1} - x^*) \le G_1(x^*) - G_1(x^{t+1}) \tag{25}$$

We now use (22) and (14) in order to simplify (25). We have that

$$\begin{aligned} A(x^{t+1} - x^*) &= (Ax^{t+1} + b - Bz^{t+1}) + B(z^{t+1} - z^*) \\ &= (H^t)^{-1}(p^{t+1} - p^t) + B(z^{t+1} - z^*) \end{aligned} \tag{26}$$

Using (26) and the symmetry of H^t we can rewrite (25) as

$$\begin{aligned} &\left[p^{t+1} + H^t B(z^{t+1} - z^t)\right]^T B(z^{t+1} - z^*) + (p^{t+1} - p^t)^T B(z^{t+1} - z^t) \\ &+ {p^{t+1}}^T (H^t)^{-1}(p^{t+1} - p^t) \qquad \le \quad G_1(x^*) - G_1(x^{t+1}) \end{aligned} \tag{27}$$

Substituting z^* for z in (21) we obtain

$$-{p^{t+1}}^T B(z^{t+1} - z^*) \leq G_2(z^*) - G_2(z^{t+1}) \tag{28}$$

We now add (24), (27) and (28). We obtain

$$\begin{aligned} (p^{t+1} - p^*)^T (H^t)^{-1}(p^{t+1} - p^t) \quad &+ \quad [H^t B(z^{t+1} - z^t)]^T B(z^{t+1} - z^*) \\ + (p^{t+1} - p^t)^T B(z^{t+1} - z^t) \quad & \qquad\qquad\qquad\qquad \leq 0 \end{aligned} \tag{29}$$

We concentrate firstly on the last term in the above three-term sum. In order to find its sign, we substitute z^t for z in (21). This yields

$$-{p^{t+1}}^T B(z^{t+1} - z^t) \leq G_2(z^t) - G_2(z^{t+1}) \tag{30}$$

Now, to eliminate the left hand side term in this last equation, we consider (21) once more, this time for iteration t, and we substitute z^{t+1} for z. We get

$${p^t}^T B(z^{t+1} - z^t) \leq G_2(z^{t+1}) - G_2(z^t) \tag{31}$$

Adding (30) and (31) yields

$$0 \leq (p^{t+1} - p^t)^T B(z^{t+1} - z^t) \tag{32}$$

i.e. the third term in (29) is always non-negative. Then, *a fortiori*, the sum of the other two terms is always non-positive, i.e.

$$(p^{t+1} - p^*)^T (H^t)^{-1}(p^{t+1} - p^t) + [H^t B(z^{t+1} - z^t)]^T B(z^{t+1} - z^*) \leq 0 \tag{33}$$

We now introduce the notation

$$\bar{z}^t := z^t - z^*, \qquad \bar{p}^t := p^t - p^*$$

Then (33) can be rewritten as

$$(\bar{p}^{t+1} - \bar{p}^t)^T (H^t)^{-1}\bar{p}^{t+1} + [B(\bar{z}^{t+1} - \bar{z}^t)]^T H^t \; B\bar{z}^{t+1} \leq 0 \tag{34}$$

in which we have used the symmetry of H^t. We will now provide estimates for each of the two inner product terms. We use the fact that, for any two vectors $a, c \in \mathbb{R}^l$, and any $l \times l$ real symmetric matrix M, one has

$$(a - c)^T M a = \frac{1}{2}\left[(a - c)^T M(a - c) + a^T M a - c^T M c\right]$$

In case M is symmetric positive definite, it induces a vector norm in $\mathbb{R}^l$, defined by

$$\|x\|_M := (x^T M x)^{\frac{1}{2}}$$

and the equality can also be written as

$$(a-c)^T M a = \frac{1}{2}\left[\|a-c\|_M^2 + \|a\|_M^2 - \|c\|_M^2\right]$$

By assumption, $\{H^t\}$ is a sequence of symmetric positive definite matrices. Then, the same is true of the sequence $\left\{(H^t)^{-1}\right\}$. Therefore, both $\{H^t\}$ and $\left\{(H^t)^{-1}\right\}$ induce vector norms, and we obtain the following estimates

$$\begin{aligned} (\bar{p}^{t+1}-\bar{p}^t)^T (H^t)^{-1}\bar{p}^{t+1} &= \frac{1}{2}\left[\|\bar{p}^{t+1}-\bar{p}^t\|^2_{(H^t)^{-1}} + \|\bar{p}^{t+1}\|^2_{(H^t)^{-1}} - \|\bar{p}^t\|^2_{(H^t)^{-1}}\right] \\ [B(\bar{z}^{t+1}-\bar{z}^t)]^T H^t B\bar{z}^{t+1} &= \frac{1}{2}\left[\|B(\bar{z}^{t+1}-\bar{z}^t)\|^2_{H^t} + \|B\bar{z}^{t+1}\|^2_{H^t} - \|B\bar{z}^t\|^2_{H^t}\right] \end{aligned} \tag{35}$$

Substitution in (34) yields

$$\begin{aligned} \|\bar{p}^{t+1}-\bar{p}^t\|^2_{(H^t)^{-1}} + \|B(\bar{z}^{t+1}-\bar{z}^t)\|^2_{H^t} + \quad \leq \\ \left(\|\bar{p}^t\|^2_{(H^t)^{-1}} + \|B\bar{z}^t\|^2_{H^t}\right) - \left(\|\bar{p}^{t+1}\|^2_{(H^t)^{-1}} + \|B\bar{z}^{t+1}\|^2_{H^t}\right) \end{aligned} \tag{36}$$

which implies

$$\left(\|\bar{p}^{t+1}\|^2_{(H^t)^{-1}} + \|B\bar{z}^{t+1}\|^2_{H^t}\right) \leq \left(\|\bar{p}^t\|^2_{(H^t)^{-1}} + \|B\bar{z}^t\|^2_{H^t}\right), \quad \forall t \tag{37}$$

This is a key inequality in establishing the convergence of the ADI algorithm. Let the real sequence $\{\alpha^t\}$ be defined as

$$\{\alpha^t\} := \left\{\|\bar{p}^t\|^2_{(H^t)^{-1}} + \|B\bar{z}^t\|^2_{H^t}\right\} \tag{38}$$

Since $\{H^t\}$ is ultimately fixed, $\{\alpha^t\}$ is ultimately non-increasing, for any primal-dual optimal pair (x^*, z^*, p^*) for problem (9), and thus it is bounded from above. This sequence is bounded below by zero, hence it converges to its infimum. The fact that $\{\alpha^t\}$ in (38) is bounded from above, combined with the fact that $\{H^t\}$ is ultimately fixed, implies that $\left\{\|\bar{p}^t\|_2^2 + \|B\bar{z}^t\|_2^2\right\}$ is also bounded from above. It follows then that the sequences $\{\bar{p}^t\}$ and $\{B\bar{z}^t\}$ are bounded, and therefore that the sequences $\{p^t\}$ and $\{Bz^t\}$ are bounded. Then, by (26), the sequence $\{Ax^t\}$ is also bounded. This concludes the proof of lemma 2.3.4. ■

Lemma 2.3.5 *Let $\{(x^t, z^t, p^t)\}$ be a sequence of iterates produced by the ADI algorithm for problem (9). Let the symmetric positive definite matrices $\{H^t\}$ be ultimately fixed. Then* *(i)* $\{p^{t+1} - p^t\} \longrightarrow 0$, $\{Bz^{t+1} - Bz^t\} \longrightarrow 0$, $\{Ax^t + b - Bz^t\} \longrightarrow 0$.

(ii) The sequence $\{G_1(x^t) + G_2(z^t)\}$ *converges to the optimal value of the objective function for problem (9).*

Proof: Let again $(x^*,\ z^*)$ be primal optimal vectors and let p^* be an optimal dual vector for problem (9). It was shown in the proof of lemma 2.3.4 that the sequence $\{\alpha^t\}$, defined in (38), converges. Then from (36) we get that the sequence of non-negative real numbers β^t defined as

$$\{\beta^t\} := \left\{\left\|\bar{p}^{t+1} - \bar{p}^t\right\|^2_{(H^t)^{-1}} + \left\|B(\bar{z}^{t+1} - \bar{z}^t)\right\|^2_{H^t}\right\}$$

is majorized by the sequence $\{\alpha^t - \alpha^{t+1}\}$, which converges to zero, since $\{\alpha^t\}$ converges. Thus $\{\beta^t\}$ must also converge to zero. By the assumptions on $\{H^t\}$, this implies that both sequences $\{\bar{p}^{t+1} - \bar{p}^t\}$ and $\{B(\bar{z}^{t+1} - \bar{z}^t)\}$ converge to zero, and therefore

$$\{p^{t+1} - p^t\} \longrightarrow 0, \quad \{Bz^{t+1} - Bz^t\} \longrightarrow 0, \quad \{Ax^t + b - Bz^t\} \longrightarrow 0$$

the last as a consequence of the multiplier update (14). This concludes the proof of part (i).

To prove part (ii), we begin by using the fact that $\{p^{t+1} - p^t\} \longrightarrow 0$, established in part (i), and the assumptions on $\{H^t\}$, in order to take limits in (24). We get

$$G_1(x^*) + G_2(z^*) \leq \liminf_t \{G_1(x^t) + G_2(z^t)\} \tag{39}$$

Adding (27) and (28) yields

$$\begin{aligned} G_1(x^*) + G_2(z^*) \quad &\geq \quad G_1(x^{t+1}) + G_2(z^{t+1}) + {p^{t+1}}^T (H^t)^{-1}(p^{t+1} - p^t) \\ &\quad + \left[(p^{t+1} - p^t) + H^t B(z^{t+1} - z^t)\right]^T B\bar{z}^{t+1} \end{aligned} \tag{40}$$

We take limits in (40) and use the boundedness of $\{p^t\}$ and $\{B\bar{z}^t\}$ and part (i) of this lemma. We obtain

$$\limsup_t \{G_1(x^t) + G_2(z^t)\} \leq G_1(x^*) + G_2(z^*) \tag{41}$$

Combining (39) and (41) we get

$$\limsup_t \{G_1(x^t) + G_2(z^t)\} \leq G_1(x^*) + G_2(z^*) \leq \liminf_t \{G_1(x^t) + G_2(z^t)\}$$

which implies that

$$\lim_t \{G_1(x^t) + G_2(z^t)\} = G_1(x^*) + G_2(z^*)$$

and concludes the proof of part (ii) of the lemma. ∎

Lemma 2.3.6 *Let $\{(x^t, z^t, p^t)\}$ be a sequence of iterates produced by the ADI algorithm for problem (9). Let the symmetric positive definite matrices $\{H^t\}$ be ultimately fixed. Then*
(i) The sequences $\{Ax^t\}$, $\{Bz^t\}$ and $\{p^t\}$ converge.
(ii) The limit point of $\{p^t\}$ is an optimal dual multiplier for problem (9).

Proof: By lemma 2.3.4, the sequences $\{Bz^t\}$ and $\{p^t\}$ are bounded, hence they possess accumulation points. We will prove first that any accumulation point of $\{p^t\}$ is an optimal dual multiplier for problem (9). Then we will show that the accumulation points of $\{Bz^t\}$ and $\{p^t\}$ are unique. This will establish the convergence of the sequences $\{Bz^t\}$ and $\{p^t\}$, and also part (ii). The convergence of $\{Ax^t\}$ will then follow from the convergence of $\{Bz^t\}$ and part (i) of lemma 2.3.5.

Let now ϱ be an accumulation point of $\{p^t\}$ and suppose that the subsequence $\left\{p^{t_j}\right\}$ converges to ϱ. By lemma 2.3.5 part (ii), $\{Ax^t\}$ is bounded, hence its subsequence $\left\{Ax^{t_j}\right\}$ is also bounded. Let α be an accumulation point of $\left\{Ax^{t_j}\right\}$ and suppose that the subsequence $\left\{Ax^{t_{j_k}}\right\}$ converges to α. Since $\{Bz^t\}$ is bounded, its subsequence $\left\{Bz^{t_{j_k}}\right\}$ is also bounded. Then, for any accumulation point β of $\left\{Bz^{t_{j_k}}\right\}$, we have $\alpha + b = \beta$, i.e. $\left\{Bz^{t_{j_k}}\right\}$ converges to $\alpha + b$. Thus, there is a countably infinite index set $\mathcal{N}$, such that the sequence (in $\mathbb{R}^l \times \mathbb{R}^l \times \mathbb{R}^l$) $\{(Ax^n, p^n, Bz^n)\}$ $n \in \mathcal{N}$, converges to (α, ϱ, β), such that $\alpha + b = \beta$.

We will now show that ϱ is an optimal dual vector for problem (9), that is

$$\begin{array}{ccccccc} G_1(x^*) & + & G_2(z^*) & + & u^T(Ax^* + b - Bz^*) & \leq \\ G_1(x^*) & + & G_2(z^*) & + & \varrho^T(Ax^* + b - Bz^*) & \leq \\ G_1(x) & + & G_2(z) & + & \varrho^T(Ax + b - Bz) & \end{array}$$
$$\forall u \in \mathbb{R}^l, \ \forall x \in \mathbb{R}^n, \ \forall z \in \mathbb{R}^m$$

Since $Ax^* + b = Bz^*$, this is equivalent to showing that

$$G_1(x^*) + G_2(z^*) \leq G_1(x) + G_2(z) + \varrho^T(Ax + b - Bz) \quad \forall x \in \mathbb{R}^n, \ \forall z \in \mathbb{R}^m \tag{42}$$

Adding (19), (21) and (24) yields, after rearranging and adding ${p^{t+1}}^T b$ to each side

$$\begin{aligned} G_1(x^*) + G_2(z^*) &+ {p^{t+1}}^T (Ax^{t+1} + b - Bz^{t+1}) \\ &+ \left[H^t(Bz^{t+1} - Bz^t)\right]^T (Ax^{t+1} - Ax) \quad \leq \\ G_1(x) + G_2(z) &+ {p^*}^T (H^t)^{-1} (p^{t+1} - p^t) \\ &+ {p^{t+1}}^T (Ax + b - Bz) \\ & \forall t \geq 0,\ \forall x \in \mathbb{R}^n,\ \forall z \in \mathbb{R}^m \end{aligned} \tag{43}$$

In particular, if we pick the x^t and z^t iterates from the index set $\mathcal{N}$, this implies

$$\begin{aligned} G_1(x^*) + G_2(z^*) &+ {p^{n+1}}^T (Ax^{n+1} + b - Bz^{n+1}) \\ &+ \left[H^n(Bz^{n+1} - Bz^n)\right]^T (Ax^{n+1} - Ax) \quad \leq \\ G_1(x) + G_2(z) &+ {p^*}^T (H^n)^{-1} (p^{n+1} - p^n) \\ &+ {p^{n+1}}^T (Ax + b - Bz) \\ & \forall n \in \mathcal{N},\ \forall x \in \mathbb{R}^n,\ \forall z \in \mathbb{R}^m \end{aligned} \tag{44}$$

Taking limits on both sides, and using the fact that $\{Ax^n\}$ is bounded and also that, as $n \longrightarrow +\infty$, $\{Ax^n + b - Bz^n\} \longrightarrow 0$, $\{p^n\} \longrightarrow \varrho$ and $\left\{Bz^n - Bz^{n+1}\right\} \longrightarrow 0$, we get (42).

We will now show that the accumulation point (ϱ, β) is unique. We argue by contradiction: suppose $\{(p^t, Bz^t)\}$ has another accumulation point (ϱ_1, β_1). Then there exists a countably infinite index set $\mathcal{K}$ such that

$$\left\{Ax^k + b\right\} \longrightarrow \beta_1, \quad \left\{p^k\right\} \longrightarrow \varrho_1, \quad \lim_{k \in \mathcal{K}} \left\{G_1(x^k) + G_2(z^k)\right\} = G_1(x^*) + G_2(z^*)$$

and ϱ_1 is an optimal dual for problem (9). We will now retrace our previous analysis and show that $\varrho_1 = \varrho$ and $\beta_1 = \beta$.

Since ϱ_1 is an optimal dual we can write the saddle-point inequality (23) as

$$G_1(x^*) + G_2(z^*) \leq G_1(x^{t+1}) + G_2(z^{t+1}) + {\varrho_1}^T (Ax^{t+1} + b - Bz^{t+1}) \tag{45}$$

We can also substitute x^k for x in (19). We get, after rearranging and adding ${p^{t+1}}^T b$ to each side

$$\begin{aligned} G_1(x^{t+1}) \;&+\; {p^{t+1}}^T (Ax^{t+1} + b) \;+\; \left[H^t(Bz^{t+1} - Bz^t)\right]^T Ax^{t+1} \quad \leq \\ G_1(x^k) \;&+\; {p^{t+1}}^T (Ax^k + b) \;+\; \left[H^t(Bz^{t+1} - Bz^t)\right]^T Ax^k \qquad \forall k \in \mathcal{K} \end{aligned} \tag{46}$$

Substituting x^k for x in (21) yields

$$G_2(z^{t+1}) - {p^{t+1}}^T B z^{t+1} \leq G_2(z^k) - {p^{t+1}}^T B z^k \quad \forall k \in \mathcal{K} \tag{47}$$

Adding (45), (46) and (47), yields, after rearranging

$$\begin{aligned} G_1(x^*) + G_2(z^*) + (p^{t+1} - \varrho_1)^T (Ax^{t+1} + b - Bz^{t+1}) \quad \leq \\ G_1(x^k) + G_2(z^k) + {p^{t+1}}^T (Ax^k + b - Bz^k) + [H^t(Bz^{t+1} - Bz^t)]^T (Ax^k - Ax^{t+1}) \end{aligned} \tag{48}$$

After taking the limit over $k \in \mathcal{K}$ and using the fact that $\{Ax^k + b\} \longrightarrow \beta_1$ and $\{Bz^k\} \longrightarrow \beta_1$ and the boundedness of $\{p^t\}$, the right hand side reduces to

$$G_1(x^*) + G_2(z^*) + [H^t(Bz^{t+1} - Bz^t)]^T (\beta_1 - b - Ax^{t+1})$$

and thus, after cancellations, (48) reduces to

$$(p^{t+1} - \varrho_1)^T (Ax^{t+1} + b - Bz^{t+1}) + [H^t(Bz^{t+1} - Bz^t)]^T (Ax^{t+1} + b - \beta_1) \leq 0$$

which, after using the multiplier update (14), becomes

$$\begin{aligned} [(p^{t+1} - \varrho_1)]^T (H^t)^{-1} (p^{t+1} - p^t) \quad + \quad [H^t B(z^{t+1} - z^t)]^T (Bz^{t+1} - \beta_1) \\ + B(z^{t+1} - z^t)^T (p^{t+1} - p^t) \qquad\qquad \leq 0 \end{aligned}$$

which is analogous to (29). Thus, repeating the analysis, we obtain

$$\left(\|p^{t+1} - \varrho_1\|^2_{(H^t)^{-1}} + \|Bz^{t+1} - \beta_1\|^2_{H^t} \right) \leq \left(\|p^t - \varrho_1\|^2_{(H^t)^{-1}} + \|Bz^t - \beta_1\|^2_{H^t} \right), \quad \forall t \tag{49}$$

which is similar to (37). By similar arguments we infer that the sequence

$$\left\{ \|p^t - \varrho_1\|^2_{(H^t)^{-1}} + \|Bz^t - \beta_1\|^2_{H^t} \right\}$$

is convergent, thus all its subsequences must have the same limit. For the subsequence with indices $k \in \mathcal{K}$, we have that $\{Bz^k\} \longrightarrow \beta_1$, $\{p^k\} \longrightarrow \varrho_1$, thus the sequence has limit 0. Thus, for the subsequence with indices $n \in \mathcal{N}$, we have that $\{Bz^n\} \longrightarrow \beta_1$ and $\{p^n\} \longrightarrow \varrho_1$. ∎

We now show how to obtain a primal optimal vector for problem (9) by solving two special ADI minimization subproblems.

Lemma 2.3.7 *Let $\tilde{x}$ solve subproblem (12) in which p^t and Bz^t are fixed at their limit values, and let $\tilde{z}$ solve subproblem (13) in which p^t and Ax^{t+1} are fixed at their limit values, with H^t in both subproblems an arbitrary symmetric positive definite matrix. Then $(\tilde{x}, \tilde{z})$ is primal optimal for problem (9).*

Proof : Let $\{p^t\} \longrightarrow \varrho$ and $\{Bz^t\} \longrightarrow \beta$. Then $\{Ax^t\} \longrightarrow \beta - b$. By assumption, $\tilde{x}$ is a solution of

$$\min_x \quad G_1(x) + \varrho^T Ax + \frac{1}{2}\,(Ax + b - \beta)^T H^t (Ax + b - \beta)$$

thus we must have

$$\begin{aligned} G_1(\tilde{x}) \quad &+ \quad \varrho^T(A\tilde{x} + b - \beta) \quad + \quad \frac{1}{2}\,(A\tilde{x} + b - \beta)^T H^t (A\tilde{x} + b - \beta) \quad \le \\ G_1(x) \quad &+ \quad \varrho^T(Ax + b - \beta) \quad + \quad \frac{1}{2}\,(Ax + b - \beta)^T H^t (Ax + b - \beta), \qquad \forall x \in \mathbb{R}^n \end{aligned}$$

and in particular for the ADI sequence $\{x^t\}$

$$\begin{aligned} G_1(\tilde{x}) \quad &+ \quad \varrho^T(A\tilde{x} + b - \beta) \quad + \quad \frac{1}{2}\,(A\tilde{x} + b - \beta)^T H^t (A\tilde{x} + b - \beta) \quad \le \\ G_1(x^t) \quad &+ \quad \varrho^T(Ax^t + b - \beta) \quad + \quad \frac{1}{2}\,(Ax^t + b - \beta)^T H^t (Ax^t + b - \beta) \end{aligned} \tag{50}$$

Similarly, $\tilde{z}$ is a solution of

$$\min_z \quad G_1(z) - \varrho^T Bz + \frac{1}{2}\,(\beta - Bz)^T H^t (\beta - Bz)$$

and therefore we must have

$$\begin{aligned} G_2(\tilde{z}) \quad &+ \quad \varrho^T(\beta - B\tilde{z}) \quad + \quad \frac{1}{2}\,(\beta - B\tilde{z})^T H^t (\beta - B\tilde{z}) \quad \le \\ G_2(z) \quad &+ \quad \varrho^T(\beta - Bz) \quad + \quad \frac{1}{2}\,(\beta - Bz)^T H^t (\beta - Bz), \qquad \forall z \in \mathbb{R}^m \end{aligned}$$

and in particular for the ADI sequence $\{z^t\}$

$$\begin{aligned} G_2(\tilde{z}) \quad &+ \quad \varrho^T(\beta - B\tilde{z}) \quad + \quad \frac{1}{2}\,(\beta - B\tilde{z})^T H^t (\beta - B\tilde{z}) \quad \le \\ G_2(z^t) \quad &+ \quad \varrho^T(\beta - Bz^t) \quad + \quad \frac{1}{2}\,(\beta - Bz^t)^T H^t (\beta - Bz^t) \end{aligned} \tag{51}$$

Adding (50) and (51) yields

$$\begin{aligned} G_1(\tilde{x}) + G_2(\tilde{z}) \quad &+ \quad \varrho^T(A\tilde{x} + b - B\tilde{z}) \\ + \frac{1}{2}\,(A\tilde{x} + b - \beta)^T H^t (A\tilde{x} + b - \beta) \quad &+ \quad \frac{1}{2}\,(\beta - B\tilde{z})^T H^t (\beta - B\tilde{z}) \quad \le \\ G_1(x^t) + G_2(z^t) \quad &+ \quad \varrho^T(Ax^t + b - Bz^t) \\ + \frac{1}{2}\,(Ax^t + b - \beta)^T H^t (Ax^t + b - \beta) \quad &+ \quad \frac{1}{2}\,(\beta - Bz^t)^T H^t (\beta - Bz^t) \end{aligned}$$

Let now (x^*, z^*) be any primal optimal solution for problem (9). Taking the limit in the right hand side of the last inequality, and using parts (i) and (ii) of theorem 2.3.1 yields

$$\begin{aligned} G_1(\tilde{x}) + G_2(\tilde{z}) \quad &+ \quad \varrho^T(A\tilde{x} + b - B\tilde{z}) \\ + \frac{1}{2}\,(A\tilde{x} + b - \beta)^T H^t (A\tilde{x} + b - \beta) \quad &+ \quad \frac{1}{2}\,(\beta - B\tilde{z})^T H^t (\beta - B\tilde{z}) \quad \le \\ G_1(x^*) + G_2(z^*) \end{aligned} \tag{52}$$

On the other hand, (x^*, z^*, ϱ) is an optimal primal-dual pair for problem (9). Thus

$$G_1(x^*) + G_2(z^*) \leq G_1(\tilde{x}) + G_2(\tilde{z}) + \varrho^T (A\tilde{x} + b - B\tilde{z}) \tag{53}$$

Combining (52) and (53) shows that

$$\frac{1}{2}\,(A\tilde{x} + b - \beta)^T H^t (A\tilde{x} + b - \beta) + \frac{1}{2}\,(\beta - B\tilde{z})^T H^t (\beta - B\tilde{z}) \leq 0$$

Since H^t is a symmetric positive definite matrix, this inequality implies that each quadratic term in the summation must be zero, and thus

$$A\tilde{x} + b = \beta = B\tilde{z} \tag{54}$$

which shows that $(\tilde{x}, \tilde{z})$ is primal feasible for problem (9). Using (54) in (52) and (53) we get that

$$G_1(\tilde{x}) + G_2(\tilde{z}) = G_1(x^*) + G_2(z^*)$$

i.e. $(\tilde{x}, \tilde{z})$ is primal optimal for problem (9). ∎

By combining the results in the previous lemmas we can provide a proof for the ADI convergence theorem 2.3.1.

Proof of theorem 2.3.1: Part (i) of the theorem can be proven by combining part (i) of lemma 2.3.5 and part (i) of lemma 2.3.6. Part (ii) is proven in part (ii) of lemma 2.3.5 and part (iii) is proven in part (ii) of lemma 2.3.6. Finally, part (iv) is proven in lemma 2.3.7. ∎

2.4 A simple example

We want to employ the ADI method to solve the following linear problem.

$$\begin{aligned} \min_{x_1, x_2, x_3, x_4} \quad & x_1 - x_2 + x_3 + x_4 \\ \text{subject to} \quad & x_1 - x_2 = 1 \\ & x_3 + x_4 = 1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned} \tag{55}$$

We begin by defining

$$G_1(x_1, x_2, x_3) := \begin{cases} x_1 - x_2 + x_3 & \text{if } x_1 - x_2 = 1 \text{ and } x_1, x_2, x_3 \geq 0 \\ +\infty & \text{otherwise} \end{cases} \tag{56}$$

and

$$G_2(x_4) := \begin{cases} x_4 & \text{if } x_4 \geq 0 \\ +\infty & \text{otherwise} \end{cases} \tag{57}$$

Both functions G_1 and G_2 are convex, proper and closed. They allow us to write problem (55) as

$$\begin{array}{ll} \min\limits_{x_1,\, x_2,\, x_3,\, x_4} & G_1(x_1, x_2, x_3) + G_2(x_4) \\ \text{subject to} & x_3 + x_4 = 1 \end{array}$$

which is in the form of the general problem (9), with the correspondences

$$A \leftarrow [\,0 \;\; 0 \;\; 1\,], \qquad B \leftarrow [\,-1\,], \qquad b \leftarrow -1$$

Observe that the splitting matrix A has not full column rank.

In the ADI algorithm we take, for simplicity, $H^t = I, \; \forall t$. At each iteration $t = 0, 1, \ldots$ we solve the two least-squares subproblems

$$\begin{array}{rll} (x_1^{t+1}, x_2^{t+1}, x_3^{t+1}) \in & \operatorname*{argmin}\limits_{x_1,\, x_2,\, x_3} & x_1 - x_2 + x_3 + {p^t}^T x_3 + \frac{1}{2} \left\| x_3 + x_4^t - 1 \right\|_2^2 \\ & \text{such that} & x_1 - x_2 = 1 \\ & & x_1, x_2, x_3 \geq 0 \end{array} \tag{58}$$

$$\begin{array}{rll} x_4^{t+1} = & \operatorname*{argmin}\limits_{x_4} & x_4 + {p^t}^T x_4 + \frac{1}{2} \left\| x_3^{t+1} + x_4 - 1 \right\|_2^2 \\ & \text{such that} & x_4 \geq 0 \end{array} \tag{59}$$

and then update the multipliers

$$p^{t+1} = p^t + \left(x_3^{t+1} + x_4^{t+1} - 1 \right) \tag{60}$$

The solutions to the subproblems are given by

$$\left(x_1^{t+1},\; x_2^{t+1},\; x_3^{t+1} \right) = \left(1 + \alpha,\; \alpha,\; [-p^t - x_4^t]_+ \right), \quad \text{for any } \alpha \geq 0$$

and

$$x_4^{t+1} = [-p^t - x_3^{t+1}]_+$$

If we initialize with arbitrary $x_4^0 \geq 0$ and $p^0 \geq 0$, the iterates are given by

$$\left(x_1^{t+1},\; x_2^{t+1},\; x_3^{t+1},\; x_4^{t+1},\; p^{t+1} \right) = \left(1 + \alpha,\; \alpha,\; [-p^t]_+,\; 0,\; [p^t]_+ - 1 \right), \quad \text{for any } \alpha \geq 0$$

and therefore

$$\left\{ x_3^t \right\} \longrightarrow 1, \quad \left\{ x_4^t \right\} \longrightarrow 0, \quad \left\{ p^t \right\} \longrightarrow -1$$

If we take $x_1^t = 1 + t$ and $x_2^t = t$, in which t is the iteration index, then the sequences $\{x_1^t\}$ and $\{x_2^t\}$ are divergent. However, for any $t \geq 0$, the vector $(1 + t,\; t,\; 0,\; 1)$ is primal optimal for problem (55).

2.5 Corollaries

2.5.1 Important special cases

In certain applications we know more about the structure of the minimization problem, and thus can strengthen the convergence result for the ADI algorithm.

In this section we make the following assumption.

Assumption 2.5.1 *$G_1 + G_2$ is continuous in its effective domain.*

An important special case is when the matrices A and B have linearly independent columns. Then we can characterize the behavior of the $\{(x^t, z^t)\}$ iterates, as well. We now prove two properties of full column rank matrices that we will need.

Lemma 2.5.2 *Let $C : \mathbb{R}^n \to \mathbb{R}^m$ be a matrix of full column rank. Then*
(i) the matrix $C^T C : \mathbb{R}^n \to \mathbb{R}^n$ is symmetric positive definite.
(ii) Let $\{x^t\}$ be a sequence in $\mathbb{R}^n$. If $\{Cx^t\}$ converges, then $\{x^t\}$ converges.

Proof: (i) if $C^T C$ were not positive definite, there would exist at least one vector $y \in \mathbb{R}^n \setminus \{0\}$ such that $y^T C^T C y \leq 0$, that is $||Cy||_2^2 \leq 0$, meaning $||Cy||_2^2 = 0$, which is only possible if $Cy = 0$, in contradiction with the full column rank assumption on C.
(ii) If $\{Cx^t\}$ converges, say to $\bar{y}$, then $\{C^T C x^t\}$ converges to $C^T \bar{y}$ and $\{x^t\}$ converges to $(C^T C)^{-1} C^T \bar{y}$, with the existence of the inverse matrix guaranteed by part (i). ∎

The following corollary characterizes the convergence of the ADI iterates in case the constraint matrices A and B have full column rank.

Corollary 2.5.3 *Let assumptions 2.2.1, 2.2.2 and 2.5.1 hold, and let the symmetric positive definite matrices $\{H^t\}$ be ultimately fixed. If A and B have full column rank, then*
(i) the ADI subproblem minimizers $\{x^{t+1}\}$ and $\{z^{t+1}\}$ are uniquely defined.
(ii) the sequence $\{(x^t, z^t)\}$ converges to an optimal primal point for problem (9).

Proof: (i) If A and B have full column rank, then the objective function in the subproblems (12) and (13) is strongly convex, and thus the minimizers are unique.
(ii) Under assumptions 2.2.1 and 2.2.2 we have shown in part (i) of theorem 2.3.1 that the sequences $\{Ax^t\}$ and $\{Bx^t\}$ of ADI iterates converge to a feasible point for problem (9). Then, by part (ii) of lemma 2.5.2 the sequences $\{x^t\}$ and $\{z^t\}$ converge, say to $\bar{x}$ and $\bar{y}$, respectively. By the continuity assumption 2.5.1 and part (ii) of theorem 2.3.1, $G_1(\bar{x}) + G_2(\bar{y})$ is optimal. ∎

A weaker convergence result is obtained if we make no assumptions on the matrices A and B but assume instead that the sequence $\{(x^t, z^t)\}$ has accumulation points.

Corollary 2.5.4 *Let assumptions 2.2.1, 2.2.2 and 2.5.1 hold, and let the symmetric positive definite matrices* $\{H^t\}$ *be ultimately fixed. Then, any accumulation point* $(\bar{x}, \bar{z})$ *of* $\{(x^t, z^t)\}$ *is primal optimal for problem (9).*

Proof: After thinning the sequence $\{(Ax^t, p^t, z^t)\}$, and using part (i) of theorem 2.3.1 and the continuity of the linear operators A and B, we get that $\{(Ax^t, p^t, z^t)\}$ converges to $(A\bar{x}, \varrho, B\bar{z})$ such that ϱ is an optimal dual and $A\bar{x} + b = B\bar{z}$, i.e. $(\bar{x}, \bar{z})$ is primal feasible. Part (ii) of theorem 2.3.1 and the continuity assumption on $G_1 + G_2$ then shows that the value $G_1(\bar{x}) + G_2(\bar{z})$ of the objective function is optimal. ∎

Some sufficient conditions for the existence of accumulation points are: compactness of the effective domains of G_1 and G_2, or compactness of the nonempty level sets of $G_1 + G_2$.

For the latter case, we demonstrate the sufficiency: Let (x^*, z^*) be primal optimal for problem (9). From part (ii) of theorem 2.3.1 we have that $\lim_{t \to +\infty} G_1(x^t) + G_2(z^t) = G_1(x^*) + G_2(z^*) =: \gamma$. Then, for any $\epsilon > 0$, all but a finite number of terms of $\{(x^t, z^t)\}$ lie in the nonempty level set $\Lambda_{\gamma + \epsilon}$. Since this set is compact, there exists in it at least one accumulation point $(\bar{x}, \bar{z})$ of $\{(x^t, z^t)\}$.

2.5.2 Termination criteria

One termination criterion is to check whether the current iterate $\{x^t, p^t, z^t\}$ meets, to within a specified tolerance, the Karush–Kuhn–Tucker conditions for problem (9). Another criterion is suggested by the following corollary.

Corollary 2.5.5 *If two consecutive ADI iterates* (x^t, p^t, z^t) *and* $(x^{t+1}, p^{t+1}, z^{t+1})$ *are such that* $p^t = p^{t+1}$ *and* $Bz^t = Bz^{t+1}$*, then* $(x^{t+1}, p^{t+1}, z^{t+1})$ *is an optimal primal-dual pair for problem (9).*

Proof: From the multiplier update (14) we see that $p^t = p^{t+1}$ implies

$$Ax^{t+1} + b = Bz^{t+1} \tag{61}$$

i.e. (x^{t+1}, z^{t+1}) is primal feasible for problem (9). Let now (x^*, z^*, p^*) be an optimal point for problem (9). Then,

$$G_1(x^*) + G_2(z^*) \leq G_1(x^{t+1}) + G_2(z^{t+1}) \tag{62}$$

Also, using (61) and the assumptions of the corollary in (40) yields

$$G_1(x^{t+1}) + G_2(z^{t+1}) \leq G_1(x^*) + G_2(z^*) \tag{63}$$

Combining (62) and (63) we get

$$G_1(x^{t+1}) + G_2(z^{t+1}) = G_1(x^*) + G_2(z^*) \tag{64}$$

i.e. the value of the objective function at (x^{t+1}, z^{t+1}) is optimal.

Also, using using (61) and the assumptions of the corollary in (44) shows that

$$\begin{aligned} G_1(x^*) + G_2(z^*) &\leq \\ G_1(x) + G_2(z) + {p^{t+1}}^T(Ax + b - Bz) & \qquad \forall x \in \mathbb{R}^n,\ \forall z \in \mathbb{R}^m \end{aligned} \tag{65}$$

Combining (64) and (65) results in

$$\begin{aligned} G_1(x^{t+1}) + G_2(z^{t+1}) &\leq \\ G_1(x) + G_2(z) + {p^{t+1}}^T(Ax + b - Bz) & \qquad \forall x \in \mathbb{R}^n,\ \forall z \in \mathbb{R}^m \end{aligned} \tag{66}$$

This shows that p^{t+1} is an optimal dual multiplier for problem (9). ∎

Then, a reasonable termination criterion is

$$\left\| p^t - p^{t+1} \right\| + \left\| Bz^t - Bz^{t+1} \right\| \leq \epsilon$$

for some positive tolerance ϵ. As a safeguard, one could also then check the difference in the value of the objective function

$$G_1(x^{t+1}) + G_2(z^{t+1}) - G_1(x^t) - G_2(z^t)$$

which should be close to zero.

2.6 Variations on the algorithm

In case the linear constraints in problem (9) are of the form $Ax = z$, with A having full column rank, and a single positive penalty parameter λ is used in the ADI method, the following two theorems provide characterization of convergence for two interesting variants.

In the first variant the subproblems are solved inexactly and a relaxation parameter ω is added. The following theorem, describing the variant, is from Eckstein and Bertsekas [28, theorem 8]. These authors report that an implementation of this variant for a linear programming application with over-relaxation factor $\omega = 1.5$ produced about 15% faster convergence than the standard ADI method with $\omega = 1$.

Theorem 2.6.1 (Over-relaxation plus inexact minimization) *Consider the finite-dimensional problem*

$$\begin{aligned} \min_{x,\, z} \quad & G_1(x) + G_2(z) \\ \text{subject to} \quad & Ax = z \end{aligned} \tag{67}$$

with A a full-column rank $m \times n$ matrix and G_1, G_2 closed, proper, convex, extended-real-valued functions. Let the non-negative real sequences $\{\mu^t\}$, $\{\nu^t\}$ and $\{\omega^t\}$ satisfy

$$\sum_t \mu^t + \sum_t \nu^t < +\infty, \quad \{\omega^t\} \subset (0,2), \quad 0 < \inf_t \omega^t \leq \sup_t \omega^t < 2$$

and let the real sequences $\{x^t\}$, $\{z^t\}$ and $\{p^t\}$ satisfy

$$\begin{aligned} &\left\| x^{t+1} - \operatorname*{argmin}_x \left\{ G_1(x) + {p^t}^T Ax + \frac{\lambda}{2} \left\| Ax - z^t \right\|_2^2 \right\} \right\| \leq \mu^t \\ &\left\| z^{t+1} - \operatorname*{argmin}_z \left\{ G_2(z) - {p^t}^T z + \frac{\lambda}{2} \left\| \omega^t Ax^{t+1} + (1-\omega^t) z^t - z \right\|_2^2 \right\} \right\| \leq \nu^t \\ &p^{t+1} = p^t + \lambda \left[\omega^t Ax^{t+1} + (1-\omega^t) z^t - z^{t+1} \right] \end{aligned}$$

in which λ is a real positive scalar and (p^0, z^0) is arbitrary. If the problem has a Kuhn-Tucker pair, then $\{x^t\}$ converges to a primal solution x^* and $\{p^t\}$ to a solution of the dual. If the dual has no optimal solution, then at least one of the sequences $\{x^t\}$, $\{p^t\}$ is unbounded.

In the second variant, called the *Peaceman–Rachford* method, a multiplier update is interpolated between the two subproblems. The intention is to incorporate in the objective function of the second minimization the most recent information on the violation of the constraints. However, the resulting algorithm requires more stringent assumptions for convergence. It is also less robust in general, as Fortin and Glowinski point out [34], citing experience in a variety of numerical analysis applications. The method is named after the authors of [80], in which a related ADI method for solving partial differential equations is introduced. The correspondence is brought forth by Gabay [37] and by Glowinski and Le Tallec [41]. The following theorem is from Eckstein [25, pp. 123].

Theorem 2.6.2 (Peaceman–Rachford method for convex optimization) *Assume that the problem (67) has a Kuhn-Tucker pair. Let either of the following conditions hold*

(i) G_1 is strictly convex, ∂G_1 is single-valued, A is square and invertible,

(ii) G_2 is strictly convex, ∂G_2 is single-valued, A has full column rank.

Let the iterative method

$$
\begin{aligned}
x^{t+1} &\in \underset{x}{\operatorname{argmin}} \quad G_1(x) + {p^t}^T Ax + \frac{\lambda}{2}\left\|Ax - z^t\right\|_2^2 \\
q^{t+1} &= p^t + \lambda(Ax^{t+1} - z^t) \\
z^{t+1} &\in \underset{z}{\operatorname{argmin}} \quad G_2(z) - {q^{t+1}}^T z + \frac{\lambda}{2}\left\|Ax^{t+1} - z\right\|_2^2 \\
p^{t+1} &= q^{t+1} + \lambda\left(Ax^{t+1} - z^{t+1}\right)
\end{aligned}
\tag{68}
$$

be started from arbitrary vectors p^0, z^0 *and let the scalar* $\lambda > 0$ *be ultimately fixed. Then,* $\{x^t\}$ *and* $\{p^t\}$ *converge to solutions of the primal and dual problem, respectively.*

Chapter 3

Application to the Block-Angular Problem

3.1 Overview

In this chapter and the three subsequent ones we specialize the ADI algorithm to the block-angular problem (**CBA**). We map (**CBA**) onto problem (9) by specifying functions G_1 and G_2 so that the objective, variables and constraints of (**CBA**) are partitioned across G_1 and G_2. Good splittings exhibit high degree of block-separability, which can then be exploited in designing parallel algorithms.

In section 3.2 we discuss the structure of the constraint set of problem (**CBA**), and the related issues of feasibility and solvability. In section 3.3 we present a straightforward splitting that fails to be block-decomposable. A discussion of its shortcomings motivates the search for good, decomposable splittings, that are derived in the next three chapters.

3.2 The structure of the block-angular problem

We now examine in more detail the block-angular convex minimization problem (**CBA**) introduced in section 1.4. Let $\mathcal{B}_{[i]}$, $i = 1, \ldots, K$ be the polyhedral sets in $\mathbb{R}^{n_i}$ representing the feasible region for the block constraints in (5)

$$\mathcal{B}_{[i]} := \{x_{[i]} \in \mathbb{R}^{n_i} \mid A_{[i]}x_{[i]} = b_{[i]} \text{ and } 0 \leq x_{[i]} \leq u_{[i]}\} \tag{69}$$

and let $\mathcal{M}$ be the polyhedral set in $\prod_{i=1}^{K} \mathbb{R}^{n_i}$

$$\mathcal{M} := \{(x_{[1]}, \ldots, x_{[K]}) \mid \sum_{i=1}^{K} D_{[i]} x_{[i]} \leq \mathtt{d}\} \tag{70}$$

Then the feasible set of (**CBA**) can be written as

$$\mathcal{F} := \prod_{i=1}^{K} \mathcal{B}_{[i]} \bigcap \mathcal{M} \tag{71}$$

We now make a basic assumption.

Assumption 3.2.1 *Problem (**CBA**) admits a lagrangean saddlepoint.*

We note that feasibility of (**CBA**) is not enough to guarantee that (**CBA**) is solvable, except for special cases. We discuss two of these cases here.

If all upper bounds $u_{[i]}$ are finite, then each polyhedral (thus: closed) set $\mathcal{B}_{[i]}$ is bounded, therefore compact in finite-dimensional $\mathbb{R}^{n_i}$, and the same is true of their set-product. Also the set $\mathcal{M}$ is closed. Then, the feasible set $\mathcal{F}$ of (**CBA**) is a closed subset of a compact set, therefore it is compact itself. Since the functions $f_{[i]}$, $i = 1, \ldots, K$ are continuous, by assumption, the value

$$\begin{array}{ll} \displaystyle\inf_{x_{[i]} \in \mathbb{R}^{n_i}} & \displaystyle\sum_{i=1}^{K} f_{[i]}(x_{[i]}) \\ \text{subject to} & (x_{[1]}, \ldots, x_{[K]}) \in \mathcal{F} \end{array} \tag{72}$$

is attained, by the Bolzano–Weierstrass theorem 1.6.15.

In the case of problem (**LQ**), if we assume that the feasible set $\mathcal{F}$ is nonempty, we may guarantee the existence of a minimizer (and therefore, the existence of a saddlepoint) even when (some of) the upper bounds are $+\infty$, by the following theorem.

Theorem 3.2.2 (Frank–Wolfe [35]) *A quadratic function that is bounded below (resp. above) on a polyhedral set achieves its minimum (resp. maximum) on that set.*

For (**LQ**), each $f_{[i]}$ is bounded below on $\mathcal{B}_{[i]}$ if, for instance, $c_{[i]}$ is non-negative, or if $Q_{[i]}$ is positive definite. For the former case the proof is trivial: the non-negativity of $c_{[i]}$ plus the positive semi-definiteness, by assumption, of $Q_{[i]}$ guarantee that $f_{[i]} \geq 0$ on $\mathcal{B}_{[i]}$. For the latter case we sketch a proof, based on a rate-of-growth argument.

Suppose that in block i, with $u_{[i]} = +\infty$ and $Q_{[i]}$ positive definite, $f_{[i]}$ is unbounded below. This implies that there exist a feasible point $\hat{x}_{[i]} \in \mathbb{R}^{n_i}$ and a feasible direction $y_{[i]} \in \mathbb{R}^{n_i} \setminus \{0\}$, such that:

$y_{[i]} \geq 0$, with strict inequality holding for at least one component, $y_{[i]} \in \ker A_{[i]}$, and, for positive scalar λ,

$$\lim_{\lambda \to +\infty} \; {c_{[i]}}^T (\hat{x}_{[i]} + \lambda y_{[i]}) + (\hat{x}_{[i]} + \lambda y_{[i]})^T Q_{[i]} (\hat{x}_{[i]} + \lambda y_{[i]}) = -\infty$$

which yields, after simplification

$$\lim_{\lambda \to +\infty} \; \lambda \, (c_{[i]} + 2 Q_{[i]} \hat{x}_{[i]})^T y_{[i]} + \lambda^2 \; {y_{[i]}}^T Q_{[i]} y_{[i]} = -\infty$$

The second term in the sum is positive, since $Q_{[i]}$ is positive definite. Thus, for λ sufficiently large, the whole expression is positive, and even tends to $+\infty$ as $\lambda \to +\infty$. (Equivalently: the above limit could be $-\infty$ only if $y_{[i]} \in \ker Q_{[i]}$. But $\ker Q_{[i]} = \{0\}$, since $Q_{[i]}$ is positive definite.) ∎

3.3 Two-block splitting

In a straightforward approach for "splitting" (**CBA**), we assign half of the blocks to G_1 and the rest to G_2, and we treat the coupling constraints as explicit equality constraints. We define for each block $i = 1, \ldots, K$ the extended objective function

$$h_{[i]}(x_{[i]}) := \begin{cases} f_{[i]}(x_{[i]}) & \text{if } A_{[i]} x_{[i]} = b_{[i]} \text{ and } 0 \leq x_{[i]} \leq u_{[i]} \\ +\infty & \text{otherwise} \end{cases} \tag{73}$$

Let now $\psi(\cdot \mid \mathcal{B}_{[i]})$ be the indicator function of the polyhedral set $\mathcal{B}_{[i]}$ defined in (69). These sets are non-empty, by assumption 3.2.1. Then

$$h_{[i]} = f_{[i]} \; + \; \psi(\cdot \mid \mathcal{B}_{[i]})$$

and the extended-real-valued function $h_{[i]}$ is proper, closed and convex.

Let now $J = \lfloor K/2 \rfloor$. We induce a split by defining

$$G_1 \left(x_{[1]}, \ldots, x_{[J]} \right) := \sum_{i=1}^{J} h_{[i]}(x_{[i]}) \tag{74}$$

and

$$G_2 \left(x_{[J+1]}, \ldots, x_{[K]} \right) := \sum_{i=J+1}^{K} h_{[i]}(x_{[i]}) \tag{75}$$

Both G_1 and G_2 are proper, by assumption 3.2.1. Problem (**CBA**) is equivalent to

$$\begin{array}{ll} \min\limits_{x_{[1]},\ldots,x_{[K]}} & G_1\left(x_{[1]},\ldots,x_{[J]}\right) + G_2\left(x_{[J+1]},\ldots,x_{[K]}\right) \\ \text{subject to} & \sum\limits_{i=1}^{J} D_{[i]}x_{[i]} - \mathtt{d} = -\sum\limits_{i=J+1}^{K} D_{[i]}x_{[i]} \end{array} \tag{76}$$

which is in the form of problem (9), with the correspondences

$$A \leftarrow [D_{[1]} \ \ldots \ D_{[J]}], \qquad B \leftarrow [D_{[J+1]} \ \ldots \ D_{[K]}], \qquad b \leftarrow -\,\mathtt{d}$$

We let the penalty matrix be of the form $H^t = \operatorname{diag}(H_1^t, H_2^t)$, with H_1^t and H_2^t both symmetric positive definite matrices. The iterative step of the ADI algorithm consists of solving the two subproblems

$$\begin{array}{lll} \left(x_{[1]}^{t+1},\ldots,x_{[J]}^{t+1}\right) & \in \ \operatorname*{argmin}\limits_{x_{[1]},\ldots,x_{[J]}} & \sum\limits_{i=1}^{J} f_{[i]}(x_{[i]}) + {p^t}^T \sum\limits_{i=1}^{J} D_{[i]}x_{[i]} \\ & & +\dfrac{1}{2}\left\| \sum\limits_{i=1}^{J} D_{[i]}x_{[i]} - \mathtt{d} + \sum\limits_{i=J+1}^{K} D_{[i]}x_{[i]}^{t} \right\|_{H_1^t}^2 \\ & \quad \text{subject to} & x_{[i]} \in \mathcal{B}_{[i]}, \quad i = 1,\ldots,J \end{array}$$

$$\begin{array}{lll} \left(x_{[J+1]}^{t+1},\ldots,x_{[K]}^{t+1}\right) & \in \ \operatorname*{argmin}\limits_{x_{[J+1]},\ldots,x_{[K]}} & \sum\limits_{i=J+1}^{K} f_{[i]}(x_{[i]}) + {p^t}^T \sum\limits_{i=J+1}^{K} D_{[i]}x_{[i]} \\ & & +\dfrac{1}{2}\left\| \sum\limits_{i=J+1}^{K} D_{[i]}x_{[i]} - \mathtt{d} + \sum\limits_{i=J+1}^{K} D_{[i]}x_{[i]}^{t+1} \right\|_{H_2^t}^2 \\ & \quad \text{subject to} & x_{[i]} \in \mathcal{B}_{[i]}, \quad i = J+1,\ldots,K \end{array}$$

and then updating the multipliers

$$p^{t+1} = p^t + H^t \left(\sum_{i=1}^{K} D_{[i]}x_{[i]}^{t+1} - \mathtt{d} \right)$$

When implementing this algorithm on a coarse-grain parallel environment, we would like to assign each block i, $i = 1,\ldots,K$ to exactly one processor. In case there are more blocks than processors, we would make the assignment via a load-balancing scheme, such as a "bin packing" heuristic [38].

The feasible set in the first subproblem is just $\prod_{i=1}^{J} \mathcal{B}_{[i]}$ and thus we would like to decompose the subproblem into J decoupled ones, each having $\mathcal{B}_{[i]}$ as its feasible set, and an objective function involving only the block variables $x_{[i]}$. Thus each processor, in the *fork* phase, would solve a problem defined on local data only. Ignoring constant terms, the objective function in the first subproblem is

$$\sum_{i=1}^{J} \left\{ f_{[i]}(x_{[i]}) + (D_{[i]}x_{[i]})^T \left[p^t + 2\left(\sum_{i=J+1}^{K} D_{[i]}x_{[i]}^t - \mathtt{d} \right) + D_{[i]}x_{[i]} + \sum_{\substack{j \neq i \\ j=1}}^{J} D_{[j]}x_{[j]} \right] \right\}$$

in which the block variables $x_{[i]}$, $i = 1, \ldots, J$ are coupled via the bilinear terms

$$(D_{[i]}x_{[i]})^T D_{[j]}x_{[j]}, \quad j \neq i, \ j = 1, \ldots, J$$

Thus the first subproblem is not block-decomposable, except for the case of a two-block (**CBA**) problem. This is also the case for the second subproblem.

We could circumvent this lack of decomposability by linearizing each bilinear term around a previous iterate. Then, at iteration t, $x_{[i]}^{t+1}$ would be a minimizer with respect to an objective function that would include the Jacobi-like term

$$(D_{[i]}x_{[i]})^T D_{[j]}x_{[j]}^t, \quad j \neq i, \ j = 1, \ldots, J$$

The application of linearization to the Augmented Lagrangian is one of the key ideas in the approximate method of multipliers of Stephanopoulos and Westerberg [95] and in the Diagonal Quadratic Approximation method of Mulvey and Ruszczyński [71].

Chapter 4

The Activity-and-Resource Proximization splitting (ARP)

4.1 Overview

In this chapter we study the Activity-and-Resource Proximization splitting (**ARP**), the first of the three block-decomposable splittings for the convex block-angular problem (**CBA**). All three splittings are such that the first subproblem for each ADI iteration (12) decomposes into independent block-subproblems that can be solved in parallel, while the computation of the closed-form solution of the second ADI subproblem (13) can be parallelized.

Section 4.2 presents the basic idea behind the (**ARP**) splitting: the introduction of vectors describing an allocation of the shared resource to each block. In section 4.3 we derive the resulting ADI scheme, and in section 4.4 we simplify and parallelize the iterative step, and present certain properties of the iterates. The simplified algorithm is presented in section 4.5. In section 4.6 we examine the convergence of the method. In the remaining sections we study and characterize two variants: primal relaxation, and the Peaceman–Rachford method. We conclude by proving, in section 4.10, that these variants are related: the Peaceman–Rachford method is the relaxed Douglas–Rachford method with over-relaxation factor 2.

4.2 The basic idea

In the (**ARP**) splitting we introduce for each block i a vector of additional variables $\tilde{d}_{[i]} \in \mathrm{I\!R}^{m_0}$, that serves as an upper bound for the vector $D_{[i]}x_{[i]}$. Since $D_{[i]}x_{[i]}$ is the amount of the shared resource vector $\mathtt{d}$ consumed by activity $x_{[i]}$, $\tilde{d}_{[i]}$ is an allocation of the shared resource to block i. In this scheme the constraint terms in the objective correspond to proximal terms for both the activities $x_{[i]}$ and the resource allocation vectors $\tilde{d}_{[i]}$.

4.3 Derivation

We define

$$G_1\left(x_{[1]},\ldots,x_{[K]},\tilde{d}_{[1]},\ldots,\tilde{d}_{[K]}\right) := \begin{cases} \sum_{i=1}^{K} h_{[i]}(x_{[i]}) & \text{if } D_{[i]}x_{[i]} \leq \tilde{d}_{[i]},\ \forall i=1,\ldots,K \\ +\infty & \text{otherwise} \end{cases} \tag{77}$$

in which the extended block-objective function $h_{[i]}(x_{[i]})$ is as in (73). We also define

$$G_2\left(y_{[1]},\ldots,y_{[K]},d_{[1]},\ldots,d_{[K]}\right) := \begin{cases} 0 & \text{if } \sum_{i=1}^{K} d_{[i]} = \mathtt{d} \\ +\infty & \text{otherwise} \end{cases} \tag{78}$$

in which each $y_{[i]} \in \mathrm{I\!R}^{n_i}$ and each $d_{[i]} \in \mathrm{I\!R}^{m_0}$. Both functions G_1 and G_2 are closed, convex and also proper, because of our assumption that (**CBA**) is solvable. Problem (**CBA**) is equivalent to

$$\begin{array}{lll} \min\limits_{x_{[i]},\,\tilde{d}_{[i]},\,y_{[i]},\,d_{[i]}} & G_1\left(x_{[1]},\ldots,x_{[K]},\tilde{d}_{[1]},\ldots,\tilde{d}_{[K]}\right) + G_2\left(y_{[1]},\ldots,y_{[K]},d_{[1]},\ldots,d_{[K]}\right) & \\ \text{subject to} & \left.\begin{array}{c} x_{[i]} = y_{[i]} \\ \tilde{d}_{[i]} = d_{[i]} \end{array}\right\} \quad i=1,\ldots,K & \end{array} \tag{79}$$

which is in the form of the general problem (9), with the correspondences

$$A \leftarrow I, \qquad B \leftarrow I, \qquad b \leftarrow 0 \tag{80}$$

The introduction of $\tilde{d}_{[i]}$ thus allows us to split implicitly the coupling constraints of (**CBA**) between G_1 and G_2.

We associate a multiplier vector $q_{[i]} \in \mathrm{I\!R}^{n_i}$ with each constraint $x_{[i]} = y_{[i]}$, and a multiplier vector $p_{[i]} \in \mathrm{I\!R}^{m_0}$ with each constraint $\tilde{d}_{[i]} = d_{[i]}$. For added flexibility, we let each block $i =$

$1, \ldots, K$ maintain its own symmetric positive definite penalty matrix $H^t_{x[i]}$ for the proximization of the activities $x_{[i]}$. For simplicity of the global updates, we will have all blocks maintain the same symmetric positive definite penalty matrix H^t_d for the proximization of the allocation of the shared resource $\tilde{d}_{[i]}$.

We now present the extended ADI algorithm for this splitting. We will use a shorthand notation, and write x^{t+1} for the concatenation of the vectors $x^{t+1}_{[1]}, \ldots, x^{t+1}_{[K]}$, and similarly for y^{t+1} etc. We will also write $f(x)$ for $\sum_{i=1}^{K} f_{[i]}(x_{[i]})$. We define $H^t_x := \operatorname{diag}\left(H^t_{x[1]}, \ldots, H^t_{x[K]}\right)$ and let the symmetric positive definite matrix H^t_K consist of K copies of H^t_d placed along the diagonal.

We initialize with arbitrary vectors q^0, p^0, y^0, d^0 and symmetric positive definite matrices $H^0_{x[i]}$ and H^0_d. At each iteration $t = 0, 1, \ldots$ we solve the two subproblems

$$\begin{aligned} (x^{t+1}, \tilde{d}^{t+1}) = \quad & \underset{x,\, d}{\operatorname{argmin}} \quad f(x) + {q^t}^T x + {p^t}^T d + \frac{1}{2}\left\|x - y^t\right\|^2_{H^t_x} + \frac{1}{2}\left\|d - d^t\right\|^2_{H^t_d} \\ & \text{s. t.} \qquad \left.\begin{aligned} A_{[i]} x_{[i]} &= b_{[i]} \\ D_{[i]} x_{[i]} &\le d_{[i]} \\ 0 \le x_{[i]} &\le u_{[i]} \end{aligned}\right\} \quad i = 1, \ldots, K \end{aligned} \tag{81}$$

$$\begin{aligned} (y^{t+1}, d^{t+1}) = \quad & \underset{y,\, d}{\operatorname{argmin}} \quad -{q^t}^T y - {p^t}^T d + \frac{1}{2}\left\|y - x^{t+1}\right\|^2_{H^t_x} + \frac{1}{2}\left\|d - \tilde{d}^{t+1}\right\|^2_{H^t_K} \\ & \text{s. t.} \qquad \sum_{i=1}^{K} d_{[i]} = \mathtt{d} \end{aligned} \tag{82}$$

Then we update the multipliers

$$q^{t+1} = q^t + H^t_x \left(x^{t+1} - y^{t+1}\right) \tag{83}$$

$$p^{t+1} = p^t + H^t_K \left(\tilde{d}^{t+1} - d^{t+1}\right) \tag{84}$$

and possibly the penalty matrices H^t_x and H^t_d.

4.4 The iterative step in detail

We observe that the objective in both subproblems is strongly convex, and thus the minimizer in each, if it exists, it is unique.

We also observe that, in the first subproblem (81) the objective and the constraints are fully decomposable per block. Thus we can solve the following K block-subproblems in parallel and then concatenate their solutions.

$$
\begin{array}{lll}
\min\limits_{x_{[i]},\, d_{[i]}} & f_{[i]}(x_{[i]}) + {q^t_{[i]}}^T x_{[i]} + {p^t_{[i]}}^T d_{[i]} + \frac{1}{2}\left\| x_{[i]} - y^t_{[i]} \right\|^2_{H^t_{x[i]}} + \frac{1}{2}\left\| \tilde{d}_{[i]} - d^t_{[i]} \right\|^2_{H^t_d} & \\
\text{s. t.} & A_{[i]} x_{[i]} = b_{[i]} & (85) \\
 & D_{[i]} x_{[i]} \leq d_{[i]} & \\
 & 0 \leq x_{[i]} \leq u_{[i]} &
\end{array}
$$

We now rename the vectors $\tilde{d}^{t+1}$ as $d^{t+\frac{1}{2}}$. The reason for this is that we want the superscript $t+1$ to designate the vectors that need to be passed from iteration t to the next. While the algorithm, as presented, appears to require six vectors per iteration, i.e. x^t, d^t, y^t, $\tilde{d}^t$, q^t and p^t, in the notation of (81)–(84), we will show that only three of these are actually needed: x^t, d^t and p^t. Using the notation $d^{t+\frac{1}{2}}$ for the d-vector that minimizes subproblem (85) emphasizes the fact that it is an intermediate quantity that need not be passed from an iteration to the next.

Let now $(x^{t+1}, d^{t+\frac{1}{2}})$ be the solution of the block subproblems (85). We will now obtain a solution in closed form for subproblem (82) by solving the Karush–Kuhn–Tucker conditions for it. This is a linearly-constrained least-squares problem. It is uncoupled in the y and d variables, and it is also unconstrained in the y variables. Thus, by setting the y-gradient equal to zero, we obtain per block

$$
y^{t+1}_{[i]} = x^{t+1}_{[i]} + (H^t_{x[i]})^{-1} q^t_{[i]} \tag{86}
$$

A solution in closed form for the d variables can then be obtained by solving the Karush–Kuhn–Tucker conditions for the subproblem

$$
\begin{array}{rll}
d^{t+1} = & \operatorname*{argmin}\limits_{d} & -{p^t}^T d + \frac{1}{2}\left\| d - d^{t+\frac{1}{2}} \right\|^2_{H^t_K} \\
 & \text{s. t.} & \sum\limits_{i=1}^{K} d_{[i]} = \mathtt{d}
\end{array} \tag{87}
$$

The conditions are

$$
-\, p^t_{[i]} + H^t_d \left(d^{t+1}_{[i]} - d^{t+\frac{1}{2}}_{[i]} \right) + u = 0, \quad i = 1, \ldots, K \tag{88}
$$

$$
\sum_{i=1}^{K} d^{t+1}_{[i]} = \mathtt{d} \tag{89}
$$

in which $u \in \mathrm{I\!R}^{m_0}$ is a dual vector paired with the equality constraints. Summing (88) over all blocks we have that

$$
-\sum_{i=1}^{K} p^t_{[i]} + H^t_d \sum_{i=1}^{K} \left(d^{t+1}_{[i]} - d^{t+\frac{1}{2}}_{[i]} \right) + Ku = 0
$$

or, after using (89),

$$-\sum_{i=1}^{K} p_{[i]}^{t} + H_d^t \left(\mathtt{d} - \sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}}\right) + Ku = 0$$

which yields

$$u = \frac{1}{K}\left[\sum_{i=1}^{K} p_{[i]}^{t} - H_d^t \left(\mathtt{d} - \sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}}\right)\right]$$

Substituting in (88) yields

$$d_{[i]}^{t+1} = d_{[i]}^{t+\frac{1}{2}} + \frac{1}{K}\left(\mathtt{d} - \sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}}\right) + (H_d^t)^{-1}\left(p_{[i]}^{t} - \frac{1}{K}\sum_{i=1}^{K} p_{[i]}^{t}\right) \tag{90}$$

The second subproblem has thus a closed-form solution that is simple to compute. Then, the computationally intensive part per iteration is the solution of the K subproblems (85), which can be carried out in parallel.

For a given block, the subproblems (85) are such that they all have the same constraints but different objective function per iteration. This difference has the form of a linear-quadratic perturbation that tends to zero towards the "tail" of the iteration sequence, as one nears convergence. This is also a property of the general ADI scheme. Thus "warm start" techniques can be used in their solution.

The iterates in this scheme have a number of interesting properties, which we state and prove in a sequence of lemmas.

Lemma 4.4.1 (Invariants of the (ARP) splitting) *Let the ADI scheme given by (81)–(84) be started from any q^0, p^0, y^0, d^0 and any symmetric positive definite penalty matrices $H_{x[i]}^0$ and H_d^0, $i = 1, \ldots, K$. Then, the iterates are such that:*
(i) $q^{t+1} = 0$, $t \geq 0$.
(ii) $y^{t+1} = x^{t+1}$, $t \geq 1$
(iii) $\sum_{i=1}^{K} d_{[i]}^{t+1} = \mathtt{d}$, $t \geq 0$
(iv) $p_{[1]}^{t+1} = p_{[2]}^{t+1} = \ldots = p_{[K]}^{t+1}$, $t \geq 0$

Proof: (i) Part (i) follows from substituting the value of y^{t+1} from (86) in the q update (83). Part (ii) then follows from taking $t+1$ for t in (86) and then using part (i). Part (iii) follows from the fact that the vectors $d_{[i]}^{t+1}$, for $t \geq 0$, solve the subproblem (82) and thus satisfy the equality constraints therein. For part (iv), we substitute the d update (90) in the p update (84). We get

$$p_{[i]}^{t+1} = \frac{1}{K}\sum_{i=1}^{K} p_{[i]}^{t} - \frac{1}{K} H_d^t \left(\mathtt{d} - \sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}}\right), \quad i = 1, \ldots, K, \ t \geq 0 \tag{91}$$

Thus the p multipliers are equal across blocks at iteration $t+1$, $t \geq 0$. ■

We will now derive some explicit formulas for the update of the p and d vectors that are equivalent to (90) and (84). One of these will use dual information from the first subproblem. We begin by defining the vectors in $\mathbb{R}^{m_0}$

$$v_{[i]}^{t+1} := p_{[i]}^t + H_d^t \, (d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^t) \tag{92}$$

for $i = 1, \ldots, K$ and $t \geq 0$. The vectors $v_{[i]}^{t+1}$ are related to some of the optimal dual multipliers obtained in the solution of the first subproblem, as the next lemma shows.

Lemma 4.4.2 *The vector $v_{[i]}^{t+1}$, defined in (92), is an optimal non-negative dual associated with the $D_{[i]}x_{[i]} \leq d_{[i]}$ constraints in the subproblem (85).*

Proof: We first state and prove the following lemma, which is similar to the saddle-point lemma 2.3.3.

Lemma 4.4.3 *Let J and G be extended-real valued convex functions, and let F be any real valued function. Let the problem*

$$\begin{array}{rl} \min\limits_{x,\, z} & J(x) + G(z) \\ \text{subject to} & F(x) \leq z \end{array} \tag{93}$$

have a saddle-point at (x^*, z^*, p^*), where (x^*, z^*) are primal optimal and p^* is an optimal dual. Let G be differentiable in an open ball around z^*. Then, $\nabla G(z^*)$ is an optimal dual p^* for this problem.

Proof: The saddle-point (x^*, z^*, p^*) satisfies the conditions

$$\begin{array}{lllllll} J(x^*) & + & G(z^*) & + & p^T[F(x^*) - z^*] & \leq & \\ J(x^*) & + & G(z^*) & + & {p^*}^T[F(x^*) - z^*] & \leq & \\ J(x) & + & G(z) & + & {p^*}^T[F(x) - z] & & \forall\ x,\ \forall\ z,\ \forall\ p \geq 0. \end{array} \tag{94}$$

which implies

$${p^*}^T[F(x^*) - z^*] = 0, \quad p^* \geq 0 \tag{95}$$

Taking $x = x^*$ in (94) we obtain

$$G(z^*) - {p^*}^T z^* \leq G(z) - {p^*}^T z, \quad \forall\ z \tag{96}$$

By assumption G is differentiable in an open ball around z^*. Then so is $G - {p^*}^T(\bullet)$, and the inequality (96) implies stationarity at z^*, that is

$$\nabla \left(G(z) - {p^*}^T z \right) \Big|_{z\, =\, z^*} = 0 \quad \Longrightarrow \quad p^* = \nabla G(z^*)$$

which proves the lemma. ∎

Now, to prove lemma 4.4.2, we take for each block i

$$\begin{aligned} J(x_{[i]}) &:= f_{[i]}(x_{[i]}) + \psi(x_{[i]} \mid \mathcal{B}_{[i]}) + {q_{[i]}^t}^T x_{[i]} + \frac{1}{2}\left\| x_{[i]} - y_{[i]}^t \right\|^2_{H^t_{x[i]}} \\ G(d_{[i]}) &:= {p_{[i]}^t}^T d_{[i]} + \frac{1}{2}\left\| d_{[i]} - d_{[i]}^t \right\|^2_{H^t_d} \\ F(x_{[i]}) &:= D_{[i]} x_{[i]} \end{aligned}$$

Then the minimization problem

$$\begin{array}{rccc} \displaystyle\min_{x_{[i]},\, d_{[i]}} & J(x_{[i]}) & + & G(d_{[i]}) \\ \text{subject to} & F(x_{[i]}) & \leq & d_{[i]} \end{array} \tag{97}$$

is exactly the ADI subproblem (85) in the format of problem (93). Both $J(x_{[i]})$ and $G(d_{[i]})$ are convex, and the latter is also differentiable. The primal optimal point $(x_{[i]}^{t+1}, d_{[i]}^{t+\frac{1}{2}})$ for problem (85) is part of a saddle-point. By lemma 4.4.3, $\nabla G(d_{[i]}^{t+\frac{1}{2}})$ is an optimal dual multiplier. But

$$\nabla G(d_{[i]}^{t+\frac{1}{2}}) = p_{[i]}^t + H_d^t\,(d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^t)$$

which is just $v_{[i]}^{t+1}$, as defined in (92). ∎

The following two lemmas present equivalent updates for the d^t proximal vector and the p multipliers. We let $\mathtt{p}^{t+1}$ denote the common value of the p multiplier across all blocks for $t \geq 0$, in accordance with part (iv) of lemma 4.4.1.

Lemma 4.4.4 (Updates for d^t) *Let the ADI scheme for the (**ARP**) splitting be started from any q^0, p^0, y^0, d^0 and any symmetric positive definite penalty matrices $H^0_{x[i]}$ and H^0_d. Then, the following updates are equivalent to (90).*

$$d_{[i]}^{t+1} = d_{[i]}^{t+\frac{1}{2}} + (H_d^t)^{-1}\left(p_{[i]}^t - \mathtt{p}^{t+1}\right), \quad t \geq 0 \tag{98}$$

$$d_{[i]}^{t+1} = d_{[i]}^{t} + (H_d^t)^{-1}\left(v_{[i]}^{t+1} - \mathtt{p}^{t+1}\right), \quad t \geq 0 \tag{99}$$

$$d_{[i]}^{t+1} = d_{[i]}^{t+\frac{1}{2}} + \frac{1}{K}\left(\mathtt{d} - \sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}}\right), \quad t \geq 1 \tag{100}$$

Proof: The update (98) is just a rearrangement of the p update (84), while (99) is derived by summing and re-arranging (84) and (92). Finally, (100) is obtained by using part (iv) of lemma 4.4.1 in simplifying (90). ∎

Lemma 4.4.5 (Multipliers p are averages) *Let the ADI scheme for the (***ARP***) splitting be initialized from any q^0, p^0, y^0, d^0 and any symmetric positive definite penalty matrices $H^0_{x[i]}$ and H^0_d. Then, the following* p *updates are equivalent to (84).*

$$\mathtt{p}^{t+1} = p^t_{[i]} - \frac{1}{K} H^t_d \left(\mathtt{d} - \sum_{i=1}^{K} d^{t+\frac{1}{2}}_{[i]} \right), \quad t \geq 1 \tag{101}$$

$$\mathtt{p}^{t+1} = \frac{1}{K} \sum_{i=1}^{K} v^{t+1}_{[i]}, \quad t \geq 1 \tag{102}$$

Proof: The first update is obtained by using part (iv) of lemma 4.4.1 in simplifying (91). For the second update we average (92) over all blocks. We get

$$\begin{aligned}
\frac{1}{K} \sum_{i=1}^{K} v^{t+1}_{[i]} &= \frac{1}{K} \sum_{i=1}^{K} p^t_{[i]} + H^t_d \frac{1}{K} \sum_{i=1}^{K} \left(d^{t+\frac{1}{2}}_{[i]} - d^t_{[i]} \right) \\
&= p^t_{[i]} - H^t_d \frac{1}{K} \left(\sum_{i=1}^{K} d^t_{[i]} - \sum_{i=1}^{K} d^{t+\frac{1}{2}}_{[i]} \right) \\
&= p^t_{[i]} - H^t_d \frac{1}{K} \left(\mathtt{d} - \sum_{i=1}^{K} d^{t+\frac{1}{2}}_{[i]} \right) \\
&= p^{t+1}_{[i]} \\
&= \mathtt{p}^{t+1}, \quad t \geq 1.
\end{aligned}$$

where we have used: part (ii) of lemma 4.4.1 on the equality of $p^t_{[i]}$ across blocks for $t \geq 1$, to get the second and the last equality; part (iii) of the same lemma to replace $\sum_{i=1}^{K} d^t_{[i]}$ by $\mathtt{d}$, for $t \geq 1$, to get the third equality; and (101) to get the fourth equality. ∎

This lemma has an important corollary.

Corollary 4.4.6 *For any initialization of the ADI scheme for the (***ARP***) splitting, the multiplier* $\mathtt{p}^t$ *is non-negative, for* $t \geq 2$.

Proof: By lemma 4.4.5, for $t \geq 2$ the multiplier $\mathtt{p}^t$ is the average of the vectors $v^t_{[i]}$, which are non-negative, since they are optimal dual vectors corresponding to inequality constraints, as shown in lemma 4.4.2. ∎

The properties of the iterates, that we established in the previous lemmas, allow us to construct an iteration-by-iteration description of the algorithm, as follows:

We choose arbitrary initial proximal terms $y^0_{[i]}$ and $d^0_{[i]}$, and multipliers $q^0_{[i]}$ and $p^0_{[i]}$. The vectors $d^1_{[i]}$, $q^1_{[i]}$ and $p^1_{[i]}$ produced in the first iteration are such that $\sum_{i=1}^{K} d^1_{[i]} = \mathtt{d}$, the multipliers $q^1_{[i]}$ are zero, and the $p^1_{[i]}$ are equal across the blocks. Moreover, these properties will now hold for the q^t, d^t and p^t vectors produced by all subsequent iterations. The second iteration will produce non-negative multipliers $p^2_{[i]}$. This will also be true for all subsequent iterations. The third iteration and all subsequent ones will have in the first subproblem as proximal term for each activity $x_{[i]}$ the optimal level of this activity in the subproblem of the previous iteration.

An appropriate choice of the initial vectors allows us to have these properties established one iteration earlier. We call such a choice a canonical initialization.

Definition 4.4.7 *An initialization d^0, q^0, p^0 for the (**ARP**) scheme is* CANONICAL *if*

$$q^0 = 0, \quad p^0_{[1]} = p^0_{[2]} = \ldots = p^0_{[K]} \geq 0, \quad \sum_{i=1}^{K} d^0_{[i]} = \mathtt{d}$$

4.5 The iterative step simplified

We now use the results of the previous section and canonical initialization in order to present a simpler version of the iterative step: we pass only three vectors from an iteration to the next, instead of the original six. We denote by $\mathtt{p}$ the common value of the p multiplier across all blocks.

Simplified (**ARP**) algorithm

(0) Pick a vector $\mathtt{p}^0 \geq 0$ and proximal vectors $x^0_{[i]}$ and $d^0_{[i]}$ $i = 1, \ldots, K$, such that $\sum_{i=1}^{K} d^0_{[i]} = \mathtt{d}$. Also pick symmetric positive definite matrices $H^0_{x[i]}$ and H^0_d. Set $t = 0$.

(1) Compute (in parallel) for each block $i = 1, \ldots, K$, $(x^{t+1}_{[i]}, d^{t+\frac{1}{2}}_{[i]})$, the solution to

$$\begin{array}{lll} \min\limits_{x_{[i]},\, d_{[i]}} & f_{[i]}(x_{[i]}) + \mathtt{p}^{t\,T} d_{[i]} + \frac{1}{2} \left\| x_{[i]} - x^t_{[i]} \right\|^2_{H^t_{x[i]}} + \frac{1}{2} \left\| d_{[i]} - d^t_{[i]} \right\|^2_{H^t_d} \\ \text{s. t.} & A_{[i]} x_{[i]} = b_{[i]}, \quad 0 \leq x_{[i]} \leq u_{[i]} \\ & D_{[i]} x_{[i]} \leq d_{[i]} \end{array} \tag{103}$$

(2) Adjust allocations to achieve feasibility

$$d^{t+1}_{[i]} = d^{t+\frac{1}{2}}_{[i]} + \frac{1}{K} \left(\mathtt{d} - \sum_{i=1}^{K} d^{t+\frac{1}{2}}_{[i]} \right) \tag{104}$$

(3) Update the multipliers

$$\mathtt{p}^{t+1} = \mathtt{p}^t - \frac{1}{K} H^t_d \left(\mathtt{d} - \sum_{i=1}^{K} d^{t+\frac{1}{2}}_{[i]} \right) \tag{105}$$

(4) Update the penalty factors $H^t_{x[i]}$ and H^t_d.

(5) If termination criteria are met, stop. Else set $t = t + 1$ and go to (1).

Notice that the d and $\mathtt{p}$ updates, which constitute the *join* phase, involve just the summation of the $d^{t+\frac{1}{2}}_{[i]}$ vectors over all blocks; after the new values are distributed to the PEs, the subproblems in step (1) are solved independently, in the next *fork* phase.

The d^t and $\mathtt{p}^t$ updates in the above scheme employ the vectors $d^{t+\frac{1}{2}}_{[i]}$. We can also have a reverse-order update employing the $v^{t+1}_{[i]}$ vectors: first, $\mathtt{p}^{t+1}$ is computed as average

$$\mathtt{p}^{t+1} = \frac{1}{K} \sum_{i=1}^{K} v^{t+1}_{[i]}$$

and this new value is used in updating the proximal d term, by

$$d^{t+1}_{[i]} = d^{t+\frac{1}{2}}_{[i]} + (H^t_d)^{-1} \left(\mathtt{p}^t - \mathtt{p}^{t+1} \right)$$

or, if we do not want to use at all the vectors $d^{t+\frac{1}{2}}_{[i]}$, by

$$d^{t+1}_{[i]} = d^t_{[i]} + (H^t_d)^{-1} \left(v^{t+1}_{[i]} - \mathtt{p}^{t+1} \right)$$

The validity of these updates was demonstrated in lemmas 4.4.4 and 4.4.5.

As we showed in lemma 4.4.2, the $v_{[i]}^{t+1}$ vectors are optimal duals for subproblem (85). Standard optimization packages, such as MINOS [76, 77], which can be employed to solve this subproblem, return both primal and dual vectors at optimality, and thus we can have the $v_{[i]}^{t+1}$ vectors available at no extra computational cost. The arithmetic operation count is roughly the same for both the $d_{[i]}^{t+\frac{1}{2}}$-based and the $v_{[i]}^{t+1}$-based updates.

4.6 Convergence of the method

The following theorem addresses the convergence of this ADI method in the primal space. In the general ADI method we needed to assume solvability of the original problem and of the ADI subproblems. For the (**ARP**) splitting we only need to assume that the original problem (**CBA**) is solvable.

Theorem 4.6.1 (Primal convergence) *Let assumption 3.2.1 hold. Let the ADI method for the (**ARP**) splitting, given by (81)–(84), be started from arbitrary q^0, p^0, y^0, d^0 and arbitrary symmetric positive definite penalty matrices H_x^0 and H_d^0, and let the symmetric positive definite matrices $\{H_x^t\}$ and $\{H_d^t\}$ be ultimately fixed. Then, the sequence of ADI iterates $\{x^t\}$ converges to an optimal primal solution for (**CBA**).*

Proof: By their definition in (77) and (78), the functions G_1 and G_2 are convex and closed. They are also proper, since (**CBA**) is solvable.

We now have to guarantee solvability of the subproblems (85) and (82) for each iteration. Since G_1 and G_2 are proper, the subproblems are feasible. The objective function of each consists of a proximal quadratic term plus a convex function. Each objective is thus strongly convex, and therefore has bounded level sets. Since each objective is also continuous, its level sets are also closed. Hence, they are compact. Also, the feasible region for each subproblem is a (non-empty) polyhedral set, thus closed. Thus, for both subproblems, the intersection of the feasible region (a closed set) with any level set (a compact set) is a compact set. Then, by the Bolzano–Weierstass theorem 1.6.15, the infimum of the continuous objective over such a set is attained; hence both subproblems are solvable.

Since the splitting matrices A and B in the correspondences (80) are both identity matrices, they have full column rank. Also the functions G_1 and G_2 are continuous in their effective domain, by construction. Then, by corollary 2.5.3 to the ADI convergence theorem 2.3.1, the sequence $\left\{(x_{[i]}^t, d_{[i]}^{t+\frac{1}{2}})\right\}$

converges to a primal optimal solution (x^*, d^*) of problem (79), and also $\left\{d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^t\right\} \rightarrow 0$.

Since the x iterates are feasible with respect to the block constraints, i.e. $x_{[i]} \in \mathcal{B}_i$, $t \geq 1$, and each set $\mathcal{B}_i$ is closed, the limit point $x^*_{[i]}$ is also feasible, i.e. $x^*_{[i]} \in \mathcal{B}_i$.

Now, for the coupling constraints: We have that $D_{[i]} x_{[i]}^{t+1} \leq d_{[i]}^{t+\frac{1}{2}}$, and thus, after taking limits, $D_{[i]} x^*_{[i]} \leq d^*_{[i]}$. Since $\left\{d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^t\right\} \rightarrow 0$, while $\sum_{i=1}^{K} d_{[i]}^{t+1} = \mathtt{d}$, we have that $\sum_{i=1}^{K} D_{[i]} x^*_{[i]} \leq \mathtt{d}$.

Thus x^* is feasible for (**CBA**). Then, x^* is optimal for (**CBA**) since on $\mathrm{dom}(G_1 + G_2)$ the value of $G_1 + G_2$ is exactly the value of the objective function of (**CBA**). ∎

By the general ADI theorem 2.3.1 we also get the convergence of the sequence $\left\{p_{[i]}^t\right\}$, $i = 1, \ldots, K$ to an optimal multiplier $p^*{}_{[i]}$ for the $d_{[i]} = \tilde{d}_{[i]}$ constraints of the extended problem (79), and from part (iv) of lemma 4.4.1 we get that $p^*{}_{[1]} = \ldots = p^*{}_{[K]}$. Since (**CBA**) and the two subproblems have polyhedral constraints, they satisfy a Constraint Qualification, therefore each minimizer is a Karush–Kuhn–Tucker point and associated optimal duals exist for each. If (**CBA**) has a differentiable objective, and we assume that the sequence of duals for the first subproblem has an accumulation point, we can show that the algorithm generates also a sequence of duals that converges to a set of optimal duals for (**CBA**), with p^* an optimal dual for the coupling constraints. This is shown in the following theorem.

Theorem 4.6.2 (Convergence of duals) *Assume, in addition, that $f_{[i]}$, $i = 1, \ldots, K$ is differentiable. Let the optimal dual multipliers for subproblem (85)*

$$w_{[i]}^{t+1} \in \mathbb{R}_+^{m_i}, \quad v_{[i]}^{t+1} \in \mathbb{R}_+^{m_0}, \quad r_{[i]}^{t+1} \in \mathbb{R}_+^{n_i}, \quad s_{[i]}^{t+1} \in \mathbb{R}_+^{n_i}$$

be paired with the constraints as follows:

$$\begin{array}{llrlll} w_{[i]}^{t+1} & \text{with} & A_{[i]} x_{[i]} = b_{[i]}, & v_{[i]}^{t+1} & \text{with} & D_{[i]} x_{[i]} \leq d_{[i]} \\ r_{[i]}^{t+1} & \text{with} & -x_{[i]} \leq 0, & s_{[i]}^{t+1} & \text{with} & x_{[i]} - u_{[i]} \leq 0 \end{array}$$

*Also assume that the upper bound vector $u_{[i]}$ has no $+\infty$ component. If the sequence $\{(w^t, r^t, s^t)\}$ has an accumulation point $(\bar{w}, \bar{r}, \bar{s})$, then $(\bar{w}, p^*, \bar{r}, \bar{s})$ are optimal duals for (**CBA**), for the pairing*

$$\begin{array}{llrlll} \bar{w}_{[i]} & \text{with} & A_{[i]} x_{[i]} = b_{[i]}, & p^* & \text{with} & \sum_{i=1}^{K} D_{[i]} x_{[i]} \leq \mathtt{d} \\ \bar{r}_{[i]} & \text{with} & -x_{[i]} \leq 0, & \bar{s}_{[i]} & \text{with} & x_{[i]} - u_{[i]} \leq 0 \end{array}$$

in which p^* is the limit of $\dfrac{1}{K} \sum_{i=1}^{K} v_{[i]}^{t+1}$.

Proof: Let $(x_{[i]}^{t+1}, d_{[i]}^{t+\frac{1}{2}})$ be optimal for subproblem (85). The Karush–Kuhn–Tucker conditions with differentiability yield

$$\nabla f_{[i]}(x_{[i]}^{t+1}) + H_d^t(x_{[i]}^{t+1} - y_{[i]}^t) + w_{[i]}^{t+1^T} A_{[i]} + v_{[i]}^{t+1^T} D_{[i]} - r_{[i]}^{t+1} + s_{[i]}^{t+1} = 0 \tag{106}$$

$$p_{[i]}^t + H_d^t(d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^t) = v_{[i]}^{t+1} \tag{107}$$

$$w_{[i]}^{t+1^T}(A_{[i]}x_{[i]}^{t+1} - b_{[i]}) + v_{[i]}^{t+1^T}(D_{[i]}x_{[i]}^{t+1} - d_{[i]}^{t+\frac{1}{2}}) - r_{[i]}^{t+1^T} x_{[i]}^{t+1} + s_{[i]}^{t+1^T}(x_{[i]}^{t+1} - u_{[i]}) = 0 \tag{108}$$

As $t \longrightarrow +\infty$, $\left\{d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^t\right\} \longrightarrow 0$, so that $\left\{v_{[i]}^{t+1} - p_{[i]}^t\right\} \longrightarrow 0$, from (107). By the general ADI theorem, $\{x^t\}$ converges to some x^*, $\{x^t - y^t\} \longrightarrow 0$ and, by also using part (iv) of lemma 4.4.1, $\left\{p_{[i]}^t\right\} \longrightarrow p^*$. Thus $\left\{v_{[i]}^{t+1}\right\} \longrightarrow p^*$.

By the hypothesis, there exists an infinite index set $\mathcal{N}$ so that $(\{w^n\}, \{r^n\}, \{s^n\})$, $n \in \mathcal{N}$, converges to $(\bar{w}, \bar{r}, \bar{s})$. Then $(\{w^n\}, \{v^n\}, \{r^n\}, \{s^n\})$ converges to $(\bar{w}, p^*, \bar{r}, \bar{s})$, and $\{x^n\}$ converges to x^*.

Summing (106) and (108) over all blocks, and then taking the limit over $n \in \mathcal{N}$ shows that $(x^*, \bar{w}, p^*, \bar{r}, \bar{s})$ satisfy the Karush–Kuhn–Tucker conditions for (**CBA**). ∎

In case $u_{[i]}$ has $+\infty$ components, then the corresponding components of the dual multiplier s_i do not exist, and can be discarded in the statement of the theorem.

4.7 Variants of the (ARP) splitting

In the following sections we examine two variants of the (**ARP**) splitting. They both share the good characteristics of the ADI method we just examined: the first subproblem decomposes into independent block-subproblems that can be solved in parallel, and the computation of the solution of the second subproblem can be parallelized.

The first variant, presented in section 4.8, involves relaxing the primal iterates between the two subproblems. The second one, presented in section 4.9, is the Peaceman–Rachford algorithm, in which an additional multiplier is introduced and updated between the solution of the subproblems. Both variants have p^t and d^t updates similar to those for the Douglas–Rachford method. The subproblems are also similar, with one important difference: the proximal term for the $x_{[i]}$ activities is not just the optimal value of the activity at the previous iteration.

Convergence for these variants is based on the fact that, for the (**ARP**) splitting, the correspondences (80) to the general ADI method are such that: A has full column rank, b is zero and B is the identity matrix. Thus theorems 2.6.1 and 2.6.2 apply.

The Peaceman–Rachford method requires more stringent assumptions for convergence than the Douglas–Rachford method: each component function $f_{[i]}$ of the (**CBA**) objective has to be strictly convex and differentiable. This is the case for ex. the (**LQ**) problem with the matrix Q positive definite.

Both theorems 2.6.1 and 2.6.2 require that a positive scalar penalty parameter λ be used, which has to be ultimately fixed. We have not been able to derive a variable penalty ADI method for the relaxation and the Peaceman–Rachford variants, because the proof technique employed in theorem 2.3.1 does not generalize to these variants. However, it might be possible to extend a laborious proof of Glowinski and Fortin [34, chapter3] to cover the case; this is a topic of future research.

Finally, in section 4.10 we demonstrate the connection between the two variants and the Douglas–Rachford method.

Additional (**ARP**) splittings can be derived by suitable redefinitions of G_1 and G_2, such as "transferring" the box constraints on the $x_{[i]}$ activities from G_1 to G_2, or allowing in G_2 the sum of the allocations of the shared resource to the blocks to be less than or equal to the shared resource. In the latter case, the derivation of simplified updates is more involved than in the method we examined, yet, if we use a diagonal positive penalty matrix, the invariants and the update formulas are similar.

4.8 Primal Relaxation

4.8.1 The iterative step in detail

For simplicity, we use a single fixed relaxation parameter ω, with $0 < \omega < 2$; we also assume that the subproblems are solved exactly. At each iteration $t = 0, 1, \ldots$ each block solves the subproblem

$$
\begin{aligned}
(x_{[i]}^{t+1}, d_{[i]}^{t+\frac{1}{2}}) = \quad \underset{x,\, d}{\operatorname{argmin}} \quad & f_{[i]} x_{[i]} + {q_{[i]}^{t}}^T x_{[i]} + {p_{[i]}^{t}}^T d_{[i]} \\
& + \frac{1}{2}\lambda \left\| x_{[i]} - y_{[i]}^t \right\|_2^2 + \frac{1}{2}\lambda \left\| d_{[i]} - d_{[i]}^t \right\|_2^2 \\
\text{subject to} \quad & A_{[i]} x_{[i]} = b_{[i]} \\
& 0 \le x_{[i]} \le u_{[i]} \\
& D_{[i]} x_{[i]} \le d_{[i]}
\end{aligned} \tag{109}
$$

and then relaxes the primal solutions obtained

$$
x_{[i],\omega}^{t+1} := \omega x_{[i]}^{t+1} + (1-\omega) y_{[i]}^t \tag{110}
$$

$$d_{[i],\omega}^{t+\frac{1}{2}} := \omega d_{[i]}^{t+\frac{1}{2}} + (1-\omega) d_{[i]}^{t} \tag{111}$$

The relaxed values are the proximal terms in the objective of the second subproblem

$$\begin{aligned}(y^{t+1}, d^{t+1}) = \quad & \underset{y,\, d}{\text{argmin}} \quad -{q^t}^T y - {p^t}^T d + \frac{1}{2}\lambda \left\| y - x_\omega^{t+1} \right\|_2^2 + \frac{1}{2}\lambda \left\| d - d_\omega^{t+\frac{1}{2}} \right\|_2^2 \\ & \text{subject to} \quad \sum_{i=1}^{K} d_{[i]} = \mathtt{d} \end{aligned} \tag{112}$$

Finally, the relaxed values are also used in updating the multipliers

$$q_{[i]}^{t+1} = q_{[i]}^{t} + \lambda(x_{[i],\omega}^{t+1} - y_{[i]}^{t+1}) \tag{113}$$

$$p_{[i]}^{t+1} = p_{[i]}^{t} + \lambda(d_{[i],\omega}^{t+\frac{1}{2}} - d_{[i]}^{t+1}) \tag{114}$$

We will now simplify the updates. From the minimization in (112) we get

$$y_{[i]}^{t+1} = x_{[i],\omega}^{t+1} + \frac{1}{\lambda} q_{[i]}^{t} \tag{115}$$

and also, similar to (90), we get that for $i = 1, \ldots, K$, $t \geq 0$,

$$\begin{aligned} d_{[i]}^{t+1} &= d_{[i],\omega}^{t+\frac{1}{2}} + \frac{1}{\lambda} p_{[i]}^{t} + \frac{1}{K}\left(\mathtt{d} - \sum_{i=1}^{K} d_{[i],\omega}^{t+\frac{1}{2}} - \frac{1}{\lambda}\sum_{i=1}^{K} p_{[i]}^{t}\right) \\ &= \omega d_{[i]}^{t+\frac{1}{2}} + (1-\omega) d_{[i]}^{t} + \frac{1}{\lambda}\left(p_{[i]}^{t} - \frac{1}{K}\sum_{i=1}^{K} p_{[i]}^{t}\right) \\ &\quad + \frac{1}{K}\left[\mathtt{d} - \sum_{i=1}^{K}\left(\omega d_{[i]}^{t+\frac{1}{2}} + (1-\omega) d_{[i]}^{t}\right)\right] \end{aligned} \tag{116}$$

Then, the p multiplier update yields

$$\begin{aligned} p_{[i]}^{t+1} &= \frac{1}{K}\sum_{i=1}^{K} p_{[i]}^{t} - \frac{\lambda}{K}\left(\mathtt{d} - \sum_{i=1}^{K} d_{[i],\omega}^{t+\frac{1}{2}}\right) \\ &= \frac{1}{K}\sum_{i=1}^{K} p_{[i]}^{t} - \frac{\lambda}{K}\left[\mathtt{d} - \sum_{i=1}^{K}\left(\omega d_{[i]}^{t+\frac{1}{2}} + (1-\omega) d_{[i]}^{t}\right)\right] \end{aligned} \tag{117}$$

We can then state the properties of the iterates, in a lemma similar to lemma 4.4.1.

Lemma 4.8.1 (Invariants) *Let the ADI scheme given by (109)–(114) be started from any* $(q^0,\ p^0,\ y^0, d^0)$. *Then, the iterates are such that:*

(i) $q^{t+1} = 0$, $t \geq 0$.

(ii) $y^{t+1} = \omega x^{t+1} + (1-\omega) y^t$, $t \geq 1$

(iii) $\sum_{i=1}^{K} d_{[i]}^{t+1} = \mathtt{d},\ t \geq 0$

(iv) $p_{[1]}^{t+1} = p_{[2]}^{t+1} = \ldots = p_{[K]}^{t+1},\ t \geq 0$

Proof: Part (i) follows after substitution of (115) in (113). Then, part (ii) follows from taking $t+1$ for t in (115) and then using part (i). Part (iii) follows from the fact that the vectors $d_{[i]}^{t+1}$, for $t \geq 0$, solve subproblem (109) and thus satisfy the equality constraint therein. Part (iv) follows from (117). ∎

We now use the above lemma to provide simplified update formulas for the p^t and d^t vectors. We begin by defining

$$v_{[i]}^{t+1} := p_{[i]}^{t} + \lambda(d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^{t}) \tag{118}$$

Then, again, we can prove the following lemma.

Lemma 4.8.2 *The vector $v_{[i]}^{t+1}$, defined in (118), is an optimal dual associated with the $D_{[i]}x_{[i]} \leq d_{[i]}$ constraints in the first subproblem (109) of the relaxation method.*

Proof: The first subproblem in the method with relaxation is exactly the same as in the method without relaxation, and the result follows from lemma 4.4.2. ∎

Lemma 4.8.3 (Updates for d^t and p^t) *Let the relaxed ADI scheme given by (109)–(114) be started from any $(q^0,\ p^0,\ y^0, d^0)$. Then, for $t \geq 1$,*

$$\begin{aligned} d_{[i]}^{t+1} &= (1-\omega)\, d_{[i]}^{t} + \omega \left[d_{[i]}^{t+\frac{1}{2}} + \frac{1}{K}\left(\mathtt{d} - \sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}}\right)\right] \\ &= (1-\omega)\, d_{[i]}^{t} + \omega \left[d_{[i]}^{t+\frac{1}{2}} + \frac{1}{\lambda}\left(\mathtt{p}^t - \frac{1}{K}\sum_{i=1}^{K} v_{[i]}^{t+1}\right)\right] \\ &= (1-\omega)\, d_{[i]}^{t} + \omega\, d_{[i]}^{t+\frac{1}{2}} + \frac{1}{\lambda}\left(\mathtt{p}^t - \mathtt{p}^{t+1}\right) \\ &= (1-\omega)\left(d_{[i]}^{t} + \frac{1}{\lambda}\mathtt{p}^t\right) + \omega\, d_{[i]}^{t} + \frac{1}{\lambda}\left(v_{[i]}^{t+1} - \mathtt{p}^{t+1}\right) \end{aligned} \tag{119}$$

and also

$$\begin{aligned} \mathtt{p}^{t+1} &= \mathtt{p}^t - \omega \frac{\lambda}{K}\left(\mathtt{d} - \sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}}\right) \\ \mathtt{p}^{t+1} &= (1-\omega)\,\mathtt{p}^t + \omega \frac{1}{K}\sum_{i=1}^{K} v_{[i]}^{t+1} \end{aligned} \tag{120}$$

in which $\mathtt{p}^t$ denotes the common value of p across all blocks for $t \geq 1$, in accordance with part (iv) of lemma 4.8.1.

Proof: The first d^t update is obtained after using parts (iii) and (iv) of lemma 4.8.1 to simplify the d^t update (116). Using these parts to simplify the p update (117) yields the first update in (120). The third d^t update is obtained by substituting the first p update of (120) into the first d^t update. The other d^t and p updates are obtained by averaging (118) over all blocks, and using part (iii) of lemma 4.8.1. ∎

Remarks: (i) For $\omega = 1$, the above formulas yield the updates for the (**ARP**) scheme with no relaxation, derived in section 4.4.

(ii) The non-negativity of $\mathtt{p}^{t+1}$ cannot be guaranteed in general. In the special case of under-relaxation ($0 < \omega < 1$), an inductive argument suffices to show that for $\mathtt{p}^0 \geq 0$, $\mathtt{p}^{t+1}$ is non-negative, as convex combination of non-negative vectors.

4.8.2 The iterative step simplified

By a canonical initialization we can have the above updates be valid also for $t = 0$. The simplified algorithm is as follows.

SIMPLIFIED RELAXED (ARP) ALGORITHM

(0) Pick a vector $\mathtt{p}^0 \geq 0$ and proximal vectors $y^0_{[i]}$ and $d^0_{[i]}$ $i = 1, \ldots, K$, such that $\sum_{i=1}^{K} d^0_{[i]} = \mathtt{d}$. Also pick a scalar penalty factor $\lambda \geq 0$ and a relaxation parameter ω, $0 < \omega < 2$. Set $t = 0$.

(1) Compute (in parallel) for each block $i = 1, \ldots, K$, $(x^{t+1}_{[i]}, d^{t+\frac{1}{2}}_{[i]})$, the solution to

$$\begin{array}{ll} \min\limits_{x_{[i]},\, d_{[i]}} & f_{[i]}(x_{[i]}) + \mathtt{p}^{t\,T} d_{[i]} + \frac{1}{2}\lambda \left\| x_{[i]} - y^t_{[i]} \right\|_2 + \frac{1}{2}\lambda \left\| d_{[i]} - d^t_{[i]} \right\|_2 \\ \text{s. t.} & A_{[i]} x_{[i]} = b_{[i]}, \quad 0 \leq x_{[i]} \leq u_{[i]} \\ & D_{[i]} x_{[i]} \leq d_{[i]} \end{array}$$

(2) Update the y proximal terms

$$y^{t+1}_{[i]} = \omega \; x^{t+1}_{[i]} + (1-\omega) \; y^t_{[i]}$$

(3) Adjust allocations to achieve feasibility

$$d^{t+1}_{[i]} = (1-\omega) d^t_{[i]} + \omega \left[d^{t+\frac{1}{2}}_{[i]} + \frac{1}{K} \left(\mathtt{d} - \sum_{i=1}^{K} d^{t+\frac{1}{2}}_{[i]} \right) \right]$$

(4) Update the multipliers

$$\mathtt{p}^{t+1} = \mathtt{p}^t - \omega \frac{\lambda}{K} \left(\mathtt{d} - \sum_{i=1}^{K} d^{t+\frac{1}{2}}_{[i]} \right)$$

(5) If termination criteria are met, stop. Else set $t = t+1$ and go to (1).

These updates are organized around $d^{t+\frac{1}{2}}_{[i]}$. In case we prefer to organize the updates around $v^{t+1}_{[i]}$, and not use at all the vectors $d^{t+\frac{1}{2}}_{[i]}$, a reverse-order update for canonical initialization would first update the p multipliers

$$\mathtt{p}^{t+1} = (1-\omega)\mathtt{p}^t + \omega \frac{1}{K} \sum_{i=1}^{K} v^{t+1}_{[i]}$$

and then update the d^t proximal terms

$$d_{[i]}^{t+1} = (1-\omega)\left(d_{[i]}^{t} + \frac{1}{\lambda}\mathtt{p}^{t}\right) + \omega d_{[i]}^{t} + \frac{1}{\lambda}\left(v_{[i]}^{t+1} - \mathtt{p}^{t+1}\right)$$

This update depends on the availability of the subproblem duals $v_{[i]}^{t+1}$ at each iteration.

4.8.3 Convergence

Convergence results for the relaxed ADI method are identical to those for the method without relaxation.

Theorem 4.8.4 (Primal convergence) *Let assumption 3.2.1 hold. Then, for any starting point* $(q^0,\ p^0,\ y^0, d^0)$*, any* $\lambda > 0$*, and any* $\omega \in (0,2)$*,* $\left\{x_{[i]}^{t}\right\}$ *converges to an optimal primal solution for (***CBA***).*

Proof: Follows from theorem 2.6.1, with the correspondences as in (80). The proof is similar to that for theorem 4.6.1. ∎

Theorem 4.8.5 (Convergence of duals) *Assume that* $f_{[i]},\ i = 1,\ldots,K$ *is differentiable. Let the optimal dual multipliers for the first subproblem (109)* $(w_{[i]}^{t+1},\ v_{[i]}^{t+1},\ r_{[i]}^{t+1},\ s_{[i]}^{t+1})$ *be paired with constraints as in theorem 4.6.2. If the sequence* $\{(w^t, r^t, s^t)\}$ *has an accumulation point* $(\bar{w}, \bar{r}, \bar{s})$ *, then* $(\bar{w}, p^*, \bar{r}, \bar{s})$ *are optimal duals for (***CBA***).*

Proof: Similar to the proof of theorem 4.6.2 for the case without relaxation. ∎

4.9 The Peaceman–Rachford variant

4.9.1 The iterative step in detail

In this method, at each iteration $t = 0, 1, \ldots$ each block solves the subproblem

$$\begin{aligned}
(x_{[i]}^{t+1}, d_{[i]}^{t+\frac{1}{2}}) = \quad \underset{x_{[i]},\, d_{[i]}}{\text{argmin}} \quad & f_{[i]}x_{[i]} + {q_{[i]}^{t}}^{T} x_{[i]} + {p_{[i]}^{t}}^{T} d_{[i]} \\
& + \frac{1}{2}\lambda \left\| x_{[i]} - y_{[i]}^{t} \right\|_2^2 + \frac{1}{2}\lambda \left\| d_{[i]} - d_{[i]}^{t} \right\|_2^2 \\
\text{subject to} \quad & A_{[i]}x_{[i]} = b_{[i]}, \quad 0 \le x_{[i]} \le u_{[i]} \\
& D_{[i]}x_{[i]} \le d_{[i]} \qquad (121)
\end{aligned}$$

and then computes intermediate multipliers

$$q_{[i]}^{t+\frac{1}{2}} := q_{[i]}^{t} + \lambda(x_{[i]}^{t+1} - y_{[i]}^{t}) \tag{122}$$

$$p_{[i]}^{t+\frac{1}{2}} := p_{[i]}^{t} + \lambda(d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^{t}) \tag{123}$$

These multipliers are then used as linear terms in the objective of the second subproblem. (Contrast with the relaxation method of section 4.8, in which only primal terms were modified between the subproblems.) The second subproblem is

$$\begin{aligned}(y^{t+1}, d^{t+1}) = \quad & \underset{y,\, d}{\text{argmin}} \quad -q^{t+\frac{1}{2}T} y - p^{t+\frac{1}{2}T} d + \frac{1}{2}\lambda \left\|y - x^{t+1}\right\|_2^2 + \frac{1}{2}\lambda \left\|d - d^{t+\frac{1}{2}}\right\|_2^2 \\ & \text{subject to} \quad \sum_{i=1}^{K} d_{[i]} = \mathtt{d}\end{aligned} \tag{124}$$

This has a closed form solution, given by

$$y_{[i]}^{t+1} = x_{[i]}^{t+1} + \frac{1}{\lambda} q_{[i]}^{t+\frac{1}{2}} \tag{125}$$

and

$$d_{[i]}^{t+1} = d_{[i]}^{t+\frac{1}{2}} + \frac{1}{\lambda}\left(p_{[i]}^{t+\frac{1}{2}} - \frac{1}{K}\sum_{i=1}^{K} p_{[i]}^{t+\frac{1}{2}}\right) + \frac{1}{K}\left(\mathtt{d} - \sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}}\right) \tag{126}$$

The iterative step concludes with one more multiplier update

$$q_{[i]}^{t+1} = q_{[i]}^{t+\frac{1}{2}} + \lambda(x_{[i]}^{t+1} - y_{[i]}^{t+1}) \tag{127}$$

$$p_{[i]}^{t+1} = p_{[i]}^{t+\frac{1}{2}} + \lambda(d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^{t+1}) \tag{128}$$

To simplify the updates, we eliminate $q_{[i]}^{t+\frac{1}{2}}$ between (122) and (125) and get

$$y_{[i]}^{t+1} = 2x_{[i]}^{t+1} - y_{[i]}^{t} + \frac{1}{\lambda} q_{[i]}^{t} \tag{129}$$

For the d^t update, we substitute (123) in (126) and get

$$d_{[i]}^{t+1} = 2d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^{t} + \frac{1}{K}\left[\mathtt{d} - \sum_{i=1}^{K}\left(2d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^{t}\right)\right] + \frac{1}{\lambda}\left(p_{[i]}^{t} - \frac{1}{K}\sum_{i=1}^{K} p_{[i]}^{t}\right) \tag{130}$$

For the p update, we eliminate $p_{[i]}^{t+\frac{1}{2}}$ between (123) and (128) and get

$$p_{[i]}^{t+1} = p_{[i]}^{t} + \lambda\left(2d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^{t} - d_{[i]}^{t+1}\right)$$

Substituting in this the value of $d_{[i]}^{t+1}$ from (130) yields

$$p_{[i]}^{t+1} = \frac{1}{K}\sum_{i=1}^{K} p_{[i]}^{t} - \frac{\lambda}{K}\left[\mathtt{d} - \sum_{i=1}^{K}\left(2d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^{t}\right)\right] \tag{131}$$

We state the properties of the iterates in the following lemma.

Lemma 4.9.1 (Invariants) *Let the Peaceman–Rachford scheme for the* (**ARP**) *splitting be started from any* $(q^0,\ p^0,\ y^0, d^0)$. *Then, the iterates are such that:*
(i) $q^{t+1} = 0$, $t \geq 0$.
(ii) $y^{t+1} = 2x^{t+1} - y^t$, $t \geq 1$
(iii) $\sum_{i=1}^{K} d_{[i]}^{t+1} = \mathtt{d}$, $t \geq 0$
(iv) $p_{[1]}^{t+1} = p_{[2]}^{t+1} = \ldots = p_{[K]}^{t+1}$, $t \geq 0$

Proof: Part (i) follows after substitution of (125) in (127). Then, part (ii) follows from taking $t+1$ for t in (129) and then using part (i). Part (iii) follows from the fact that the vectors $d_{[i]}^{t+1}$, for $t \geq 0$, solve (124), and thus satisfy the equality constraints therein. Part (iv) follows from (131). ∎

With the help of this lemma we can provide simplified updates for the p and d^t vectors. We begin by defining

$$v_{[i]}^{t+1} := p_{[i]}^t + \lambda(d_{[i]}^{t+\frac{1}{2}} - d_{[i]}^t) \tag{132}$$

Then, again, we can prove the following lemma.

Lemma 4.9.2 *The vector* $v_{[i]}^{t+1}$, *defined in (132), is an optimal dual associated with the* $D_{[i]}x_{[i]} \leq d_{[i]}$ *constraints in the subproblem (121).*

Proof: The first subproblem in the Peaceman–Rachford method is exactly the same as in the Douglas–Rachford method, and the result follows from lemma 4.4.2. ∎

Lemma 4.9.3 (Updates for d^t and p^t) *Let the Peaceman–Rachford scheme, given by (121)–(128), be started from any* $(q^0,\ p^0,\ y^0, d^0)$. *Then, for* $t \geq 1$,

$$\begin{aligned} d_{[i]}^{t+1} &= - \quad d_{[i]}^t \quad + \ 2 \left[d_{[i]}^{t+\frac{1}{2}} + \frac{1}{K}\left(\mathtt{d} - \sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}}\right)\right] \\ &= - \quad d_{[i]}^t \quad + \ 2 \left[d_{[i]}^{t+\frac{1}{2}} + \frac{1}{\lambda}\left(\mathtt{p}^t - \frac{1}{K}\sum_{i=1}^{K} v_{[i]}^{t+1}\right)\right] \\ &= - \quad d_{[i]}^t \quad + \ 2 \ d_{[i]}^{t+\frac{1}{2}} \ + \ \frac{1}{\lambda}\left(\mathtt{p}^t - \mathtt{p}^{t+1}\right) \\ &= - \left(d_{[i]}^t + \frac{1}{\lambda}\mathtt{p}^t\right) + \ 2 \ d_{[i]}^t \ + \ \frac{1}{\lambda}\left(v_{[i]}^{t+1} - \mathtt{p}^{t+1}\right) \end{aligned} \tag{133}$$

and also

$$\begin{aligned} \mathtt{p}^{t+1} &= \quad \mathtt{p}^t \ - \ 2 \ \frac{\lambda}{K}\left(\mathtt{d} - \sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}}\right) \\ \mathtt{p}^{t+1} &= - \ \mathtt{p}^t \ + \ 2 \ \frac{1}{K}\sum_{i=1}^{K} v_{[i]}^{t+1} \end{aligned} \tag{134}$$

in which $\mathtt{p}^t$ *denotes the common value of* p *across all blocks for* $t \geq 1$*, in accordance with part (iv) of lemma 4.9.1.*

Proof: The first d^t update is obtained after using parts (iii) and (iv) of lemma 4.9.1 to simplify the d^t update (130). Using these parts to simplify the p update (131) yields the first update in (134). The third d^t update is obtained by substituting the first p update into the first d^t update. The other d^t and p updates are obtained by averaging (132) over all blocks and using part (iii) of lemma 4.9.1. ∎

The last two updates in (133) are reverse-order formulas. A reverse-order update for canonical initialization, organized around the subproblem multipliers $v_{[i]}^{t+1}$, would first update the p multipliers

$$\mathtt{p}^{t+1} = -\mathtt{p}^t + 2\frac{1}{K}\sum_{i=1}^{K} v_{[i]}^{t+1}$$

and then update the d^t proximal terms

$$d_{[i]}^{t+1} = -\left(d_{[i]}^t + \frac{1}{\lambda}\mathtt{p}^t\right) + 2d_{[i]}^t + \frac{1}{\lambda}\left(v_{[i]}^{t+1} - \mathtt{p}^{t+1}\right)$$

In the next section we present the simplified algorithm, with updates organized around the vector $d_{[i]}^{t+\frac{1}{2}}$.

4.9.2 The iterative step simplified

By the use of canonical initialization we have the following version of the algorithm.

SIMPLIFIED PEACEMAN–RACHFORD (**ARP**) ALGORITHM

(0) Pick a vector $\mathtt{p}^0 \geq 0$ and proximal vectors $y^0_{[i]}$ and $d^0_{[i]}$ $i = 1, \ldots, K$, such that $\sum_{i=1}^{K} d^0_{[i]} = \mathtt{d}$. Also pick a scalar penalty factor $\lambda \geq 0$. Set $t = 0$.

(1) Compute (in parallel) for each block $i = 1, \ldots, K$, $(x^{t+1}_{[i]}, d^{t+\frac{1}{2}}_{[i]})$, the solution to

$$\begin{array}{ll} \min\limits_{x_{[i]},\, d_{[i]}} & f_{[i]}(x_{[i]}) + \mathtt{p}^{t\,T} d_{[i]} + \frac{1}{2}\lambda \left\| x_{[i]} - y^t_{[i]} \right\|_2 + \frac{1}{2}\lambda \left\| d_{[i]} - d^t_{[i]} \right\|_2 \\ \text{s. t.} & A_{[i]} x_{[i]} = b_{[i]}, \quad 0 \leq x_{[i]} \leq u_{[i]} \\ & D_{[i]} x_{[i]} \leq d_{[i]} \end{array}$$

(2) Update the y proximal terms

$$y^{t+1}_{[i]} = 2\, x^{t+1}_{[i]} - y^t_{[i]}$$

(3) Adjust allocations to achieve feasibility

$$d^{t+1}_{[i]} = -d^t_{[i]} + 2 \left[d^{t+\frac{1}{2}}_{[i]} + \frac{1}{K} \left(\mathtt{d} - \sum_{i=1}^{K} d^{t+\frac{1}{2}}_{[i]} \right) \right]$$

(4) Update the multipliers

$$\mathtt{p}^{t+1} = \mathtt{p}^t - 2\frac{\lambda}{K} \left(\mathtt{d} - \sum_{i=1}^{K} d^{t+\frac{1}{2}}_{[i]} \right)$$

(5) If termination criteria are met, stop. Else set $t = t + 1$ and go to (1).

4.9.3 Convergence

The convergence results for the Peaceman–Rachford method are similar to the ones for the Douglas–Rachford method.

Theorem 4.9.4 (Primal convergence) *Let assumption 3.2.1 hold, and let the functions $f_{[i]}$, $i = 1, \ldots, K$, be strictly convex and differentiable. Then, for any starting point $(q^0,\ p^0,\ y^0, d^0)$ and any $\lambda > 0$, $\left\{ x^t_{[i]} \right\}$ converges to an optimal primal solution for (**CBA**).*

Proof: Follows from theorem 2.6.2, with the correspondences as in (80). The proof is similar to that for theorem 4.6.1. ∎

Theorem 4.9.5 (Convergence of duals) *Let the assumptions of theorem 4.9.4 hold. Let the optimal dual multipliers for the subproblem (121)* $(w_{[i]}^{t+1},\ v_{[i]}^{t+1},\ r_{[i]}^{t+1},\ s_{[i]}^{t+1})$ *be paired with constraints as in theorem 4.6.2. If the sequence* $\{(w^t, r^t, s^t)\}$ *has an accumulation point* $(\bar{w}, \bar{r}, \bar{s})$ *, then* $(\bar{w}, p^*, \bar{r}, \bar{s})$ *are optimal duals for (***CBA***).*

Proof: Similar to the proof of theorem 4.6.2 for the standard method. ∎

4.10 A unifying view

The update formulas for the ADI (or Douglas–Rachford) method, the relaxation variant, and the Peaceman–Rachford scheme for the (**ARP**) splitting can be grouped together, as a family parameterized by a scalar. This is shown in the following lemma.

Lemma 4.10.1 *For the (***ARP***) splitting for (***CBA***), the Peaceman–Rachford method is the relaxed Douglas–Rachford method with over-relaxation factor 2.*

Proof: We can write the (**ARP**) update formulas as follows:

$$\begin{array}{rclcccl} y_{[i]}^{t+1} & = & (1-\omega) & y_{[i]}^{t} & + & \omega & x_{[i]}^{t+1} \\ d_{[i]}^{t+1} & = & (1-\omega) & d_{[i]}^{t} & + & \omega & \left[d_{[i]}^{t+\frac{1}{2}} + \dfrac{1}{K}\left(\mathtt{d} - \displaystyle\sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}} \right) \right] \\ \mathtt{p}^{t+1} & = & (1-\omega) & \mathtt{p}^{t} & + & \omega & \dfrac{1}{K}\displaystyle\sum_{i=1}^{K} v_{[i]}^{t} \end{array} \tag{135}$$

Setting $\omega = 1$ yields the standard Douglas–Rachford update, while $\omega = 2$ yields the Peaceman–Rachford update. Letting $\omega \in (0,2) \setminus \{1\}$ yields the relaxed Douglas–Rachford update. This can be verified by checking the update formulas for each scheme: part (ii) in lemma 4.4.1, and equations (100) and (102) for Douglas–Rachford; part (ii) in lemma 4.8.1, and equations (119) and (120) for the relaxed scheme; and part (ii) in lemma 4.9.1 and equations (133) and (134) for Peaceman–Rachford.

A comparison of (109) and (121) shows that the first subproblems per iteration are the same for both the relaxation and the Peaceman–Rachford methods. ∎

Chapter 5

The Resource Proximization splitting (RP)

5.1 Overview

In this chapter we study the Resource Proximization splitting (**RP**) for the convex block-angular problem (**CBA**). This splitting has good features, similar to (**ARP**): the first subproblem per iteration is block-decomposable and can be solved in parallel, and the computation of the closed-form solution of the second subproblem can be parallelized.

The cost of these features for (**ARP**) was the introduction of additional variables $\tilde{d}_{[i]}$ for each block, which augment the size of the computationally intensive first subproblem. The (**RP**) splitting does not introduce additional variables; thus a smaller subproblem needs to be solved per iteration. The trade-off is that convergence of the full set of primal variables is not guaranteed, although the objective function value converges to the optimal value. In addition, in order to solve the second subproblem in closed form, we need to assume that the penalty matrix is diagonal. In this splitting proximal terms are added only for the vectors $D_{[i]}x_{[i]}$ that reflect the consumption of the shared resource $\mathtt{d}$.

In section 5.2 we derive the algorithm, and in section 5.3 we analyze and simplify the iterative step. Section 5.4 presents the simplified algorithm, and section 5.5 discusses convergence. Finally, in section 5.6 we show how the (**RP**) algorithm can be thought of as a limiting case of an (**ARP**) algorithm.

5.2 Derivation

We introduce again the extended objective

$$h_{[i]}(x_{[i]}) := \begin{cases} f_{[i]}(x_{[i]}) & \text{if } A_{[i]}x_{[i]} = b_{[i]} \text{ and } 0 \leq x_{[i]} \leq u_{[i]} \\ +\infty & \text{otherwise} \end{cases} \tag{136}$$

Then we define

$$G_1\left(x_{[1]}, \ldots, x_{[K]}\right) := \sum_{i=1}^{K} h_{[i]}(x_{[i]}) \tag{137}$$

and

$$G_2\left(d_{[1]}, \ldots, d_{[K]}\right) := \begin{cases} 0 & \text{if } \sum_{i=1}^{K} d_{[i]} \leq \mathtt{d} \\ +\infty & \text{otherwise} \end{cases} \tag{138}$$

with $d_{[1]}, \ldots, d_{[K]} \in \mathbb{R}^{m_0}$. Functions G_1 and G_2 are both closed, convex and also proper, because (**CBA**) is solvable, by assumption 3.2.1.

We take the splitting matrix $\mathcal{D}$ to be block-diagonal, of size $(Km_0) \times \sum_{i=1}^{K} n_i$.

$$\mathcal{D} := \operatorname{diag}\left(D_{[1]}, D_{[2]}, \ldots, D_{[K]}\right) \tag{139}$$

Problem (**CBA**) is equivalent to

$$\begin{array}{ll} \min\limits_{x_{[i]},\, d_{[i]}} & G_1\left(x_{[1]}, \ldots, x_{[K]}\right) + G_2\left(d_{[1]}, \ldots, d_{[K]}\right) \\ \text{subject to} & D_{[i]}x_{[i]} = d_{[i]}, \quad i = 1, \ldots, K \end{array} \tag{140}$$

which is in the form of the general problem (9), with the correspondences

$$A \leftarrow \mathcal{D}, \qquad B \leftarrow I, \qquad b \leftarrow 0 \tag{141}$$

The coupling constraints of (**CBA**) are thus split between the explicit equality constraints and G_2. If certain variables in block i do not appear in the coupling constraints, then the corresponding columns of $D_{[i]}$ are zeros, and in this case the matrix $\mathcal{D}$ is not of full column rank. This is the case in at least one important practical instance of (**CBA**), the multicommodity network flow problem. We will examine the consequences of this when we discuss the convergence of this splitting.

We pair a tentative Lagrange multiplier vector $p_{[i]} \in \mathbb{R}^{m_0}$ with each block of constraints $D_{[i]}x_{[i]} = d_{[i]}$, $i = 1, \ldots, K$. For simplicity of the global updates we will have all blocks maintain the same diagonal positive penalty matrix Λ^t. We let the matrix Λ^t_K consist of K copies of Λ^t placed along the diagonal. The iterative step of the ADI algorithm (in shorthand notation) consists of solving the two subproblems

$$\begin{array}{rccl} x^{t+1} \in & \underset{x}{\operatorname{argmin}} & f(x) + {p^t}^T \mathcal{D}x + \frac{1}{2} \left\| \mathcal{D}x - d^t \right\|^2_{\Lambda^t_K} & \\ & \text{subject to} & \left. \begin{array}{c} A_{[i]} x_{[i]} = b_{[i]} \\ 0 \leq x_{[i]} \leq u_{[i]} \end{array} \right\} & i = 1, \ldots, K \end{array} \tag{142}$$

$$\begin{array}{rccl} d^{t+1} = & \underset{d}{\operatorname{argmin}} & -{p^t}^T d + \frac{1}{2} \left\| \mathcal{D}x^{t+1} - d \right\|^2_{\Lambda^t_K} \\ & \text{subject to} & \sum_{i=1}^{K} d_{[i]} \leq \mathtt{d} \end{array} \tag{143}$$

and then updating the multipliers

$$p^{t+1}_{[i]} = p^t_{[i]} + \Lambda^t \left(D_{[i]} x^{t+1}_{[i]} - d^{t+1}_{[i]} \right) \tag{144}$$

and possibly the penalty matrix Λ^t.

5.3 The iterative step in detail

The minimizer for the second subproblem (143), if it exists, is unique, since the objective is strongly convex. The minimizer for the first subproblem (142) may not be unique. Uniqueness is guaranteed in special cases, for instance if each $f_{[i]}$ is strictly convex, or if each matrix $D_{[i]}$ has full column rank.

The first subproblem is decomposable per block. Thus we need to solve the following K subproblems in parallel and concatenate the solutions.

$$\begin{array}{rl} \underset{x_{[i]}}{\min} & f_{[i]}(x_{[i]}) + {p^t_{[i]}}^T D_{[i]} x_{[i]} + \frac{1}{2} \left\| D_{[i]} x_{[i]} - d^t_{[i]} \right\|^2_{\Lambda^t} \\ \text{subject to} & A_{[i]} x_{[i]} = b_{[i]} \\ & 0 \leq x_{[i]} \leq u_{[i]} \end{array} \tag{145}$$

The second subproblem is a least-squares problem with linear constraints. A solution in closed form can be obtained by solving the Karush–Kuhn–Tucker conditions. The minimizer $(d^{t+1}_{[1]}, \ldots, d^{t+1}_{[K]})$ and any optimal dual $\mu^{t+1} \in \mathbb{R}^{m_0}_+$ must satisfy

$$-p^t_{[i]} - \Lambda^t \left(D_{[i]} x^{t+1}_{[i]} - d^{t+1}_{[i]} \right) + \mu^{t+1} = 0, \quad i = 1, \ldots, K \tag{146}$$

$${\mu^{t+1}}^T \left(\sum_{i=1}^{K} d^{t+1}_{[i]} - \mathtt{d} \right) = 0 \tag{147}$$

$$\sum_{i=1}^{K} d^{t+1}_{[i]} \leq \mathtt{d} \tag{148}$$

After calculations, we obtain that the minimizer is given by

$$d_{[i]}^{t+1} = D_{[i]}x_{[i]}^{t+1} + (\Lambda^t)^{-1}\left(p_{[i]}^t - \mu^{t+1}\right) \tag{149}$$

in which

$$\mu^{t+1} = \frac{1}{K}\left[\sum_{i=1}^{K} p_{[i]}^t + \Lambda^t\left(\sum_{i=1}^{K} D_{[i]}x_{[i]}^{t+1} - \mathtt{d}\right)\right]_+ \tag{150}$$

Use of (149) in (144) yields the following simple p update

$$p_{[i]}^{t+1} = \mu^{t+1} \tag{151}$$

The (**RP**) splitting has invariants similar to (**ARP**).

Lemma 5.3.1 (Invariants of the (RP) splitting) *Let the ADI scheme described by (142)–(144) be started from any (p^0, d^0) and any diagonal positive penalty matrix Λ^0. Then,*
(i) $p_{[1]}^{t+1} = p_{[2]}^{t+1} = \ldots = p_{[K]}^{t+1} \geq 0,\ t \geq 0.$
(ii) $\sum_{i=1}^{K} d_{[i]}^{t+1} \leq \mathtt{d},\ t \geq 0.$

Proof: Part (i) follows from (151) and (150). Part (ii) follows from the fact that the vectors $d_{[i]}^{t+1}$, for $t \geq 0$, solve (143) and thus satisfy the inequality constraints therein. ∎

A canonical initialization establishes these properties also for the starting vectors.

Definition 5.3.2 *An initialization d^0, p^0 for the ADI scheme for (**RP**) is* CANONICAL *if*

$$p_{[1]}^0 = p_{[2]}^0 = \ldots = p_{[K]}^0 \geq 0, \quad \sum_{i=1}^{K} d_{[i]}^0 = \mathtt{d}$$

5.4 The iterative step simplified

We now present a simpler version of the iterative step, by making use of the reverse-order update formulas and of canonical initialization. We denote by $\mathtt{p}$ the common value of the p multiplier across all blocks, in accordance with part (i) of lemma 5.3.1.

Simplified (RP) algorithm

(0) Pick a vector $\mathtt{p}^0 \geq 0$, proximal vectors $d^0_{[i]}$, $i = 1, \ldots, K$ such that $\sum_{i=1}^{K} d^0_{[i]} = \mathtt{d}$, and a diagonal positive matrix Λ^0. Set $t = 0$.

(1) Compute (in parallel) per block $i = 1, \ldots, K$, $x^{t+1}_{[i]}$, a solution to

$$\begin{array}{ll} \min\limits_{x_{[i]}} & f_{[i]}(x_{[i]}) + {\mathtt{p}^t}^T D_{[i]} x_{[i]} + \frac{1}{2} \left\| D_{[i]} x_{[i]} - d^t_{[i]} \right\|^2_{\Lambda^t} \\ \text{subject to} & A_{[i]} x_{[i]} = b_{[i]} \\ & 0 \leq x_{[i]} \leq u_{[i]} \end{array} \tag{152}$$

(2) Update the multipliers p

$$\mathtt{p}^{t+1} = \left\{ \mathtt{p}^t + \frac{1}{K} \Lambda^t \left(\sum_{i=1}^{K} D_{[i]} x^{t+1}_{[i]} - \mathtt{d} \right) \right\}_+ \tag{153}$$

(3) Adjust allocations to achieve feasibility

$$d^{t+1}_{[i]} = D_{[i]} x^{t+1}_{[i]} + (\Lambda^t)^{-1} \left(\mathtt{p}^t - \mathtt{p}^{t+1} \right) \tag{154}$$

(4) Update the penalty matrix Λ^t.

(5) If termination criteria are met, stop. Else set $t = t + 1$ and go to (1).

As in the previous splitting, the updates are simple to compute; they involve vector operations on local data, with the exception of the global quantity $\sum_{i=1}^{K} D_{[i]} x^{t+1}_{[i]}$ that requires communication between all blocks.

5.5 Convergence of the method

For this method we have a weaker convergence result than the one for (**ARP**). This is because the splitting matrix $\mathcal{D}$ in (139) may not have full column rank.

Theorem 5.5.1 (Primal convergence) *Let assumption 3.2.1 hold. Assume that each component function $f_{[i]}$ is either linear or quadratic, or has level sets that are bounded over $\mathcal{B}_{[i]}$, the feasible set for the block constraints. Let the ADI method given by (142)–(144) be started from arbitrary vectors p^0, d^0 and an arbitrary diagonal positive penalty matrix Λ^0. Let the diagonal positive matrices $\{\Lambda^t\}$ be ultimately fixed. Then, any sequence $\{x^t, d^t\}$ produced by the algorithm is such that* *(i)* $\left\{D_{[i]}x^t_{[i]}\right\}$*,* $i = 1, \ldots, K$*, converges to* ${d^*}_{[i]}$ *such that* $\sum_{i=1}^{K} {d^*}_{[i]} \leq \mathtt{d}$*.*

(ii) $\sum_{i=1}^{K} f_{[i]}(x^t_{[i]})$ *converges to the optimal value for (*$\mathbf{CBA}$*).*

(iii) For all $i \in \{1, \ldots, K\}$ *such that* $D_{[i]}$ *has full column rank,* $\left\{x^t_{[i]}\right\}$ *converges.*

(iv) $\left\{p^t_{[i]}\right\}$ *converges to the same limit* p^**, for all* $i = 1, \ldots, K$*.*

Proof: By their construction in (136) and (137) the functions G_1 and G_2 are convex and closed. They are also proper, since (**CBA**) is solvable, and continuous in their effective domains. We now have to guarantee solvability of subproblems (142) and (143). Since G_1 and G_2 are proper, both subproblems are feasible.

The objective in the second subproblem is strongly convex, and the feasible region is a non-empty polyhedral set. An argument similar to that in the proof of theorem 4.6.1, shows that the second subproblem is solvable. On the other hand, the assumptions on the solvability of (**CBA**) and on $f_{[i]}$ are sufficient to guarantee the solvability of the first subproblem.

The splitting matrix B in (141) is the identity, thus has full row rank. Then, by corollary 2.5.3, the sequence $\left\{d^t_{[i]}\right\}$ converges. By part (iii) of lemma 5.3.1, $\sum_{i=1}^{K} d^{t+1}_{[i]} \leq \mathtt{d}$, $t \geq 0$. Thus, for the limit point ${d^*}_{[i]}$ must also have $\sum_{i=1}^{K} {d^*}_{[i]} \leq \mathtt{d}$. Since $x^{t+1}_{[i]}$ solves the first subproblem, it is feasible with respect to the block constraints $x_{[i]} \in \mathcal{B}_{[i]}$. The ADI convergence theorem 2.3.1 also yields that $\left\{D_{[i]}x^t_{[i]} - d^t_{[i]}\right\} \longrightarrow 0$. Combining this with the fact that $\sum_{i=1}^{K} {d^*}_{[i]} \leq \mathtt{d}$ proves (i) of the theorem.

Part (ii) follows from theorem 2.3.1, and the fact that on dom $(G_1 + G_2)$, the value of $G_1 + G_2$ is just the value of the objective function of (**CBA**). Part (iii) follows from corollary 2.5.3. If all the matrices $D_{[i]}$ have full column rank, then $\{x^t\}$ converges to a primal optimal point for (**CBA**). From theorem 2.3.1 and part (i) of lemma 5.3.1 we get part (iv). ∎

The following theorem presents a method of obtaining optimal duals for (**CBA**).

Theorem 5.5.2 *Assume, in addition, that* $f_{[i]}$*,* $i = 1, \ldots, K$ *is differentiable. Let the optimal dual multipliers for subproblem (145)*

$$w^{t+1}_{[i]} \in \mathrm{IR}^{m_i}_+, \quad r^{t+1}_{[i]} \in \mathrm{IR}^{n_i}_+, \quad s^{t+1}_{[i]} \in \mathrm{IR}^{n_i}_+$$

be paired with the constraints as follows:

$$w^{t+1}_{[i]} \text{ with } A_{[i]}x_{[i]} = b_{[i]}, \quad r^{t+1}_{[i]} \text{ with } -x_{[i]} \leq 0, \quad s^{t+1}_{[i]} \text{ with } x_{[i]} - u_{[i]} \leq 0$$

Also assume that the upper bound vector $u_{[i]}$ has no $+\infty$ component. Let p^* be the limit of $\{p^t\}$. Then an optimal set of duals $(\bar{w}, p^*, \bar{r}, \bar{s})$ for (**CBA**)*, for the pairing*

$$\begin{array}{llll} \bar{w}_{[i]} & \text{with} \quad A_{[i]}x_{[i]} = b_{[i]}, & p^* & \text{with} \quad \sum_{i=1}^{K} D_{[i]}x_{[i]} \leq \mathtt{d} \\ \bar{r}_{[i]} & \text{with} \quad -x_{[i]} \leq 0, & \bar{s}_{[i]} & \text{with} \quad x_{[i]} - u_{[i]} \leq 0 \end{array}$$

can be obtained by solving subproblem (145) with $d^t_{[i]}$ and $p^t_{[i]}$ at their limit values.

Proof: Let $x^{t+1}_{[i]}$ be optimal for subproblem (145). The Karush–Kuhn–Tucker conditions with differentiability yield

$$\nabla f_{[i]}(x^{t+1}_{[i]}) + {p^t}^T D_{[i]} + \Lambda^t(D_{[i]}x^{t+1}_{[i]} - d^t_{[i]}) + {w^{t+1}_{[i]}}^T A_{[i]} - r^{t+1}_{[i]} + s^{t+1}_{[i]} = 0 \tag{155}$$

$${w^{t+1}_{[i]}}^T (A_{[i]}x^{t+1}_{[i]} - b_{[i]}) - {r^{t+1}_{[i]}}^T x^{t+1}_{[i]} + {s^{t+1}_{[i]}}^T (x^{t+1}_{[i]} - u_{[i]}) = 0 \tag{156}$$

As $t \longrightarrow +\infty$, $\left\{D_{[i]}x^{t+1}_{[i]} - d^t_{[i]}\right\} \longrightarrow 0$, $\left\{d^t_{[i]}\right\} \longrightarrow {d^*}_{[i]}$, and $\left\{p^t_{[i]}\right\} \longrightarrow p^*$. By lemma 2.3.7, we can solve a subproblem (145) with $p^t_{[i]} = p^*$ and $d^t_{[i]} = {d^*}_{[i]}$, and find an optimal $x^{t+1}_{[i]}$ so that $D_{[i]}x^{t+1}_{[i]} = {d^*}_{[i]}$. Then, if we sum the Karush–Kuhn–Tucker conditions for that subproblem over all blocks, we get that the primal and dual vectors satisfy the optimality conditions for (**CBA**).

5.6 The connection between the (ARP) and (RP) splittings

The algorithm we developed for the (**RP**) splitting can be thought of as a limiting case for an (**ARP**) algorithm in which the proximal terms for the $x_{[i]}$ activities have been set to zero, and all blocks use the same diagonal matrix with positive entries Λ to proximize the $d_{[i]}$ variables. Then the $D_{[i]}x_{[i]}$ terms for (**RP**) behave like $d^{t+\frac{1}{2}}_{[i]}$ terms for (**ARP**).

In more detail: we derive an (**ARP**) splitting by defining a function G_1 as

$$G_1\left(x_{[1]}, \ldots, x_{[K]}, \tilde{d}_{[1]}, \ldots, \tilde{d}_{[K]}\right) := \begin{cases} \sum_{i=1}^{K} f_{[i]}(x_{[i]}) + \psi(x_{[i]} \mid \mathcal{B}_{[i]}) & \text{if } D_{[i]}x_{[i]} = \tilde{d}_{[i]},\ \forall i \\ +\infty & \text{otherwise} \end{cases}$$

and by defining a function G_2 of the $d_{[i]}$ variables only, employing inequality constraints.

$$G_2\left(d_{[1]}, \ldots, d_{[K]}\right) := \begin{cases} 0 & \text{if } \sum_{i=1}^{K} d_{[i]} \leq \mathtt{d} \\ +\infty & \text{otherwise} \end{cases} \tag{157}$$

Then we can write the (**CBA**) problem as

$$\begin{array}{lll} \min\limits_{x_{[i]},\, \tilde{d}_{[i]},\, d_{[i]}} & G_1\left(x_{[1]},\ldots,x_{[K]},\tilde{d}_{[1]},\ldots,\tilde{d}_{[K]}\right) + G_2\left(d_{[1]},\ldots,d_{[K]}\right) & \\ \text{subject to} & d_{[i]} = \tilde{d}_{[i]}, \quad i=1,\ldots,K & (158) \end{array}$$

in which the $x_{[i]}$ variables are not included in the explicit constraints.

In terms of correspondences with the general problem (9), this splitting has

$$A \leftarrow [0 \; I], \qquad B \leftarrow I, \qquad b \leftarrow 0 \tag{159}$$

The splitting matrix A does not have full column rank; moreover, multipliers and primal proximal terms are associated only with the resource variables $\tilde{d}$.

After simplification, the algorithm requires solving, at each iteration, the subproblems

$$\begin{array}{ll} \min\limits_{x_{[i]},\, d_{[i]}} & f_{[i]}(x_{[i]}) + \mathtt{p}^{t^T} d_{[i]} + \frac{1}{2}\left\|d - d^t\right\|^2_{\Lambda^t} \\ \text{subject to} & A_{[i]}x_{[i]} = b_{[i]} \\ & D_{[i]}x_{[i]} = d_{[i]} \\ & 0 \le x_{[i]} \le u_{[i]} \end{array} \tag{160}$$

to obtain the $x_{[i]}^{t+1}$ and $d_{[i]}^{t+\frac{1}{2}}$ vectors. Then we update in reverse order, by

$$\mathtt{p}^{t+1} = \Lambda^t \left[\frac{1}{K}\left(\sum_{i=1}^{K} d_{[i]}^{t+\frac{1}{2}} - \mathtt{d}\right) + (\Lambda^t)^{-1}\, \mathtt{p}^t\right]_+ \tag{161}$$

and

$$d_{[i]}^{t+1} = d_{[i]}^{t+\frac{1}{2}} + (\Lambda^t)^{-1}\, \left(\mathtt{p}^t - \mathtt{p}^{t+1}\right) \tag{162}$$

Now, if we let $D_{[i]}x_{[i]}^{t+1} = d_{[i]}^{t+\frac{1}{2}}$, subproblem (160) is comparable to subproblem (145) for the (**RP**) splitting, while the updates (161) and (162) are equivalent to (153) and (154), respectively.

Chapter 6

The Activity Proximization splitting (AP)

6.1 Overview

In this chapter we study the Activity Proximization splitting (**AP**) for (**CBA**). In this splitting, the first subproblem for each iteration is block-decomposable and can be solved in parallel. This is similar to the (**ARP**) and (**RP**) splittings we have examined in chapters 4 and 4. The solution of the second subproblem per iteration of (**AP**) requires the solution of a linearly-constrained least-squares problem, or, equivalently, of a Linear Complementarity Problem. For special cases of (**CBA**), such as multicommodity network flow, if the penalty matrix is diagonal, the subproblem can be solved explicitly and the computation of the updates can be parallelized.

The cost of decomposability for (**ARP**) is the augmentation of the size of the first subproblem, because of the introduction of extra variables. The (**RP**) splitting does not introduce additional variables, but at a price: the subproblem objective function is not strongly convex and may lack desirable properties such as uniqueness of minimizers, and convergence is not guaranteed, in general, for the full set of primal variables. The (**AP**) splitting avoids both shortcomings by introducing proximal terms only for the activity vectors $x_{[i]}$, thus guaranteeing strong convexity of the objective.

The (**AP**) algorithm is derived in section 6.2, and is analyzed and simplified in section 6.3. The simplified algorithm is presented in section 6.4, and convergence is discussed in section 6.5. Finally, in section 6.6 we present two variants of the algorithm, primal relaxation and the Peaceman–Rachford method. We conclude by proving that these two are related: the Peaceman–Rachford method is the relaxed method with over-relaxation factor 2. This result is similar to the one we prove for the (**ARP**) splitting.

6.2 Derivation

We introduce again the extended objective function

$$h_{[i]}(x_{[i]}) := \begin{cases} f_{[i]}(x_{[i]}) & \text{if } A_{[i]}x_{[i]} = b_{[i]} \text{ and } 0 \leq x_{[i]} \leq u_{[i]} \\ +\infty & \text{otherwise} \end{cases} \tag{163}$$

and we define, as for (**RP**),

$$G_1\left(x_{[1]}, \ldots, x_{[K]}\right) := \sum_{i=1}^{K} h_{[i]}(x_{[i]}) \tag{164}$$

The difference with the (**RP**) splitting lies in the definition of G_2. We take here

$$G_2\left(y_{[1]}, \ldots, y_{[K]}\right) := \begin{cases} 0 & \text{if } \sum_{i=1}^{K} D_{[i]}y_{[i]} \leq \mathtt{d} \\ +\infty & \text{otherwise} \end{cases} \tag{165}$$

with $(y_{[1]}, \ldots, y_{[K]}) \in \prod_{i=1}^{K} \mathbb{R}^{n_i}$. Thus the coupling constraints of (**CBA**) are incorporated in G_2. Functions G_1 and G_2 are both closed, convex and also proper, because (**CBA**) is solvable, by assumption 3.2.1.

We take the splitting matrix to be the identity in $\prod_{i=1}^{K} \mathbb{R}^{n_i}$. We write (**CBA**) as

$$\begin{array}{ll} \min\limits_{x_{[i]},\, y_{[i]}} & G_1\left(x_{[1]}, \ldots, x_{[K]}\right) + G_2\left(y_{[1]}, \ldots, y_{[K]}\right) \\ \text{subject to} & x_{[i]} = y_{[i]}, \quad i = 1, \ldots, K \end{array} \tag{166}$$

which is in the form of the general problem (9), with the correspondences

$$A \leftarrow I, \qquad B \leftarrow I, \qquad b \leftarrow 0 \tag{167}$$

A tentative Lagrange multiplier vector $p_{[i]} \in \mathbb{R}^{n_i}$ is associated with each block of constraints $x_{[i]} = y_{[i]}$, $i = 1, \ldots, K$. For added flexibility we let each block i maintain its own symmetric positive definite penalty matrix $H^t_{[i]}$. We define $H^t := \operatorname{diag}\left(H^t_{[1]}, \ldots, H^t_{[K]}\right)$.

We now present the resulting ADI method using shorthand notation. At each iteration we solve the two subproblems

$$\begin{array}{rll} x^{t+1} = & \underset{x}{\operatorname{argmin}} & f(x) + {p^t}^T x + \frac{1}{2}\left\|x - y^t\right\|_{H^t}^2 \\ & \text{subject to} & \left.\begin{array}{c} A_{[i]} x_{[i]} = b_{[i]} \\ 0 \leq x_{[i]} \leq u_{[i]} \end{array}\right\} \quad i = 1, \ldots, K \end{array} \tag{168}$$

$$\begin{array}{rll} y^{t+1} = & \underset{y}{\operatorname{argmin}} & -{p^t}^T y + \frac{1}{2}\left\|y - x^{t+1}\right\|_{H^t}^2 \\ & \text{subject to} & \sum_{i=1}^{K} D_{[i]} y_{[i]} \leq \mathtt{d} \end{array} \tag{169}$$

and then update the multipliers

$$p^{t+1} = p^t + H^t\left(x^{t+1} - y^{t+1}\right) \tag{170}$$

and possibly the penalty matrix H^t.

6.3 The iterative step in detail

The minimizer in the first subproblem (168), if it exists, is unique, since the objective function is strongly convex. Again, objective and constraints decompose per block, so we can solve the following K block-problems in parallel and concatenate the solutions.

$$\begin{array}{ll} \underset{x_{[i]}}{\min} & f_{[i]}(x_{[i]}) + {p_{[i]}^t}^T x_{[i]} + \frac{1}{2}\left\|x_{[i]} - y_{[i]}^t\right\|_{H_{[i]}^t}^2 \\ \text{subject to} & A_{[i]} x_{[i]} = b_{[i]} \\ & 0 \leq x_{[i]} \leq u_{[i]} \end{array} \tag{171}$$

The second subproblem (169) is a linearly-constrained least-squares problem with a strongly convex objective function. Thus the (unique) minimizer y^{t+1} and any optimal dual vector $\mu^{t+1} \in \mathbb{R}_+^{m_0}$ must satisfy

$$-p_{[i]}^t - H_{[i]}^t\left(x_{[i]}^{t+1} - y_{[i]}^{t+1}\right) + {D_{[i]}}^T \mu^{t+1} = 0, \quad i = 1, \ldots, K \tag{172}$$

$${\mu^{t+1}}^T\left[\mathtt{d} - \sum_{i=1}^{K} D_{[i]} y_{[i]}^{t+1}\right] = 0 \tag{173}$$

$$\mathtt{d} - \sum_{i=1}^{K} D_{[i]} y_{[i]}^{t+1} \geq 0 \tag{174}$$

We may use any method for quadratic programming in order to obtain $y_{[i]}^{t+1}$ and μ^{t+1}. One approach is to formulate the problem as a Linear Complementarity Problem (LCP) and then solve it by an algorithm amenable to parallelization, such as an iterative matrix-splitting scheme [18, chapter 5], [21].

After calculations, we have that the vector $\mu^{t+1} \geq 0$ must satisfy

$$\begin{aligned} q + D(H^t)^{-1}D^T\mu^{t+1} &\geq 0 \\ {\mu^{t+1}}^T\left[q + D(H^t)^{-1}D^T\mu^{t+1}\right] &= 0 \end{aligned} \tag{175}$$

for $D(H^t)^{-1}D^T := \sum_{i=1}^{K} D_{[i]}(H_{[i]}^t)^{-1}{D_{[i]}}^T$ and $q := \mathtt{d} - \sum_{i=1}^{K} D_{[i]}\left(x_{[i]}^{t+1} + (H_{[i]}^t)^{-1}p_{[i]}^t\right)$.

Thus μ^{t+1} solves the Linear Complementarity Problem LCP $\left(q, D(H^t)^{-1}D^T\right)$. For certain cases such as the multicommodity flow problem, if we penalize by a diagonal matrix H^t, the solution μ^{t+1} can be written in closed form

$$\mu^{t+1} = \left([D(H^t)^{-1}D^T]^{-1}\left[\sum_{i=1}^{K} D_{[i]}\left(x_{[i]}^{t+1} + (H_{[i]}^t)^{-1}p_{[i]}^t\right) - \mathtt{d}\right]\right)_+ \tag{176}$$

In all cases, (172) yields the proximal y^t update

$$y_{[i]}^{t+1} = x_{[i]}^{t+1} + (H_{[i]}^t)^{-1}\left(p_{[i]}^t - {D_{[i]}}^T\mu^{t+1}\right) \tag{177}$$

Lemma 6.3.1 (Invariants of the (AP) splitting) *Let the ADI scheme described by (168)–(170) be started from any (p^0, y^0) and any symmetric positive definite penalty matrix H^0. Then, for all $t \geq 0$,*

(i) $p_{[i]}^{t+1} = {D_{[i]}}^T\mu^{t+1}$, with $\mu^{t+1} \geq 0$.

(ii) $\sum_{i=1}^{K} D_{[i]}y_{[i]}^{t+1} \leq \mathtt{d}$.

Proof: Part (i) follows from substituting (177) in (170). (Notice that in general we cannot guarantee that the multipliers $p_{[i]}$ will be non-negative, or equal across the blocks.) Part (ii) follows from the fact that the vectors $y_{[i]}^{t+1}$, for $t \geq 0$, solve subproblem (169) and thus satisfy the inequality constraints therein. ∎

These properties can also hold for the starting vectors by a canonical initialization:

Definition 6.3.2 *An initialization p^0, y^0 for the ADI scheme for (**AP**) is* CANONICAL *if*

$$p_{[i]}^0 = {D_{[i]}}^T\mu^0, \text{ for } \mu^0 \geq 0 \quad \text{and} \quad \sum_{i=1}^{K} D_{[i]}y_{[i]}^0 \leq \mathtt{d}$$

6.4 The iterative step simplified

We now present a version of the algorithm that uses the μ vectors, rather than the p multipliers. We also make use of canonical initialization.

SIMPLIFIED (**AP**) ALGORITHM

(0) Pick a vector $\mu^0 \geq 0$, proximal vectors $y^0_{[i]}$ such that $\sum_{i=1}^{K} D_{[i]} y^0_{[i]} \leq \mathtt{d}$ and a symmetric positive definite matrix H^0.

(1) Compute (in parallel) per block $i = 1, \ldots, K$, $x^{t+1}_{[i]}$, the solution to

$$\begin{array}{ll} \min\limits_{x_{[i]}} & f_{[i]}(x_{[i]}) + {\mu^t}^T D_{[i]} x_{[i]} + \left\| x_{[i]} - y^t_{[i]} \right\|^2_{H^t_{[i]}} \\ \text{subject to} & A_{[i]} x_{[i]} = b_{[i]} \\ & 0 \leq x_{[i]} \leq u_{[i]} \end{array} \tag{178}$$

(2) Compute the primal solution y^{t+1} and a dual solution μ^{t+1} of

$$\begin{array}{ll} \underset{y}{\text{argmin}} & -{\mu^t}^T D y + \frac{1}{2} \left\| y - x^{t+1} \right\|^2_{H^t} \\ \text{subject to} & \sum_{i=1}^{K} D_{[i]} y_{[i]} \leq \mathtt{d} \end{array} \tag{179}$$

(4) Update the penalty matrix $H^t_{[i]}$.

(5) If termination criteria are met, stop. Else set $t = t + 1$ and go to (1).

If $D(H^t)^{-1}D^T$ is diagonal and invertible, then μ^t can be simply updated by

$$\mu^{t+1} = \left(\mu^t + [D(H^t)^{-1}D^T]^{-1} \left[\sum_{i=1}^{K} D_{[i]} x^{t+1}_{[i]} - \mathtt{d} \right] \right)_+ \tag{180}$$

Then all updates are of a very simple computational nature, involving vector operations on local data, except for the vector $\sum_{i=1}^{K} D_{[i]} x^{t+1}_{[i]}$ which is computed from contributions from all PEs.

In case the least-squares subproblem is solved via an LCP algorithm, in step (2) above we solve for μ^{t+1} the problem

$$LCP\ \left(\mathtt{d} - Dx^{t+1} - D(H^t)^{-1}D^T\mu^t,\ D(H^t)^{-1}D^T \right) \tag{181}$$

and then update the proximal y-terms by

$$y^{t+1}_{[i]} = x^{t+1}_{[i]} + (H^t_{[i]})^{-1} {D_{[i]}}^T \left(\mu^t - \mu^{t+1} \right) \tag{182}$$

In the general case we can have a *master* processor solve the LCP, and then broadcast the solution μ^{t+1} to all other processors; alternatively, to reduce communication overhead, each processor can solve the LCP locally. In both cases the LCP matrix can be synthesized by globally adding the local products $D_{[i]}(H^t_{[i]})^{-1}{D_{[i]}}^T$. Their sum is stored in all processors that will be solving the LCP. In case a fixed penalty matrix is used, the LCP matrix needs to be computed only once, initially.

6.5 Convergence of the method

For this method we have a strong convergence result, similar to the (**ARP**) splitting, because the splitting matrix has full row rank.

Theorem 6.5.1 (Primal convergence) *Let assumption 3.2.1 hold. Let the ADI method given by (168)–(170) be started from any (p^0, y^0) and any symmetric positive definite penalty matrix H^0. Let the symmetric positive definite matrices $\{H^t\}$ be ultimately fixed. Then, the sequence $\{x^t\}$ of ADI iterates converges to an optimal primal solution for (**CBA**).*

Proof: By their construction in (163) and (164) the functions G_1 and G_2 are convex and closed. They are also proper, since (**CBA**) is solvable.

We now have to guarantee solvability of subproblems (168) and (169). Since G_1 and G_2 are proper, the subproblems are feasible. The objective function in each consists of a proximal quadratic term plus a convex, continuous function. Each objective function is thus strongly convex, and therefore with bounded level sets. The feasible region for each subproblem is a (non-empty) polyhedral set, thus closed. Thus, for each subproblem, the intersection of the feasible region with any level set is a compact set. Then, by the Bolzano–Weierstass theorem 1.6.15, the infimum of the continuous objective over that set is attained; hence both subproblems are solvable.

Since the splitting matrices A and B in the correspondences (167) are both identity matrices, they have full column rank. Also, the functions G_1 and G_2 are continuous in their effective domain, by construction. Then, by corollary 2.5.3, the sequence $\left\{x^t_{[i]}\right\}$ convergences to a primal optimal solution x^* of problem (166).

Since the x iterates are feasible with respect to the block constraints $x_{[i]} \in \mathcal{B}_i$, and since the sets $\mathcal{B}_i$ are closed, the limit points per block $x^*_{[i]}$ are also feasible, i.e. $x^*_{[i]} \in \mathcal{B}_i$.

Now, for the coupling constraints: From the general ADI convergence theorem we also get that $\left\{x^t_{[i]} - y^t_{[i]}\right\} \to 0$, and thus $\left\{y^t_{[i]}\right\} \to x^*_{[i]}$. We have that $\sum_{i=1}^{K} D_{[i]} y^{t+1}_{[i]} \leq \mathtt{d}$, and thus, after taking the limit, $\sum_{i=1}^{K} D_{[i]} x^*_{[i]} \leq \mathtt{d}$. Thus x^* is feasible for (**CBA**).

Optimality of x^* for (**CBA**) follows from the fact that on dom $(G_1 + G_2)$, the value of $G_1 + G_2$ is just the value of the objective function of (**CBA**). ■

6.6 Variants of the (AP) splitting

The correspondences of (**AP**) to the problem for the general ADI method are such that A has full column rank, b is zero and B is the identity matrix. Thus theorems 2.6.1 and 2.6.2 apply, and we can use primal relaxation in the Douglas–Rachford scheme, or employ the Peaceman–Rachford variant. In both algorithms we use a positive scalar penalty parameter λ that is ultimately fixed, as required by the convergence theorems.

6.6.1 Primal relaxation

We will present a brief exposition of the derivation of the relaxed scheme here. The development is similar to section 4.8 for the (**ARP**) splitting.

At each iteration $t = 0, 1 \ldots$ each block computes $x^{t+1}_{[i]}$ as the solution to

$$\begin{array}{ll} \underset{x_{[i]} \in \mathrm{I\!R}^{n_i}}{\text{minimize}} & f_{[i]}(x_{[i]}) + {p^t_{[i]}}^T x_{[i]} + \frac{1}{2}\lambda \left\| x_{[i]} - y^t_{[i]} \right\|_2^2 \\ \text{subject to} & A_{[i]} x_{[i]} = b_{[i]} \\ & 0 \leq x_{[i]} \leq u_{[i]} \end{array} \tag{183}$$

and then calculates a relaxation of the primal solution, with $\omega \in (0, 2)$

$$x^{t+1}_{[i],\omega} := \omega x^{t+1}_{[i]} + (1 - \omega) y^t_{[i]} \tag{184}$$

The second subproblem uses these relaxed values as proximal terms

$$\begin{array}{rcll} y^{t+1} & = & \underset{y}{\text{argmin}} & -{p^t}^T d + \frac{1}{2}\lambda \left\| x^{t+1}_\omega - y \right\|_2^2 \\ & & \text{subject to} & \sum_{i=1}^{K} D_{[i]} y_{[i]} \leq \mathtt{d} \end{array} \tag{185}$$

The multipliers are then updated, using the relaxed values

$$p^{t+1}_{[i]} = p^t_{[i]} + \lambda \, (x^{t+1}_{[i],\omega} - y^{t+1}_{[i]}), \quad i = 1, \ldots, K \tag{186}$$

The solution to subproblem (185) is given by

$$\begin{aligned} y_{[i]}^{t+1} &= x_{[i],\omega}^{t+1} + \frac{1}{\lambda}\left(p_{[i]}^{t} - {D_{[i]}}^{T}\mu^{t+1}\right) \\ &= \omega x_{[i]}^{t+1} + (1-\omega)y_{[i]}^{t} + \frac{1}{\lambda}\left(p_{[i]}^{t} - {D_{[i]}}^{T}\mu^{t+1}\right) \end{aligned} \tag{187}$$

for μ^{t+1} the solution to the problem

$$LCP\ \left(\ \lambda\left[d - \omega\sum_{i=1}^{K} D_{[i]}x_{[i]}^{t+1} - (1-\omega)\sum_{i=1}^{K} D_{[i]}y_{[i]}^{t}\right] - DD^T\mu^t,\ DD^T\ \right) \tag{188}$$

in which $D := [D_{[1]} \ldots D_{[K]}]$. Then the p multiplier update yields

$$p_{[i]}^{t+1} = {D_{[i]}}^{T}\mu^{t+1} \tag{189}$$

so that, for canonical initialization, we have that

$$y_{[i]}^{t+1} = \omega x_{[i]}^{t+1} + (1-\omega)y_{[i]}^{t} + \frac{1}{\lambda}D_{[i]}\left(\mu^t - \mu^{t+1}\right) \tag{190}$$

One has a strong convergence result for this variant.

Theorem 6.6.1 (Primal convergence) *Let assumption 3.2.1 hold. Then, for any starting point (p^0, q^0), any $\lambda > 0$, and any $\omega \in (0,2)$, $\left\{x_{[i]}^t\right\}$ converges to an optimal primal solution for (**CBA**).*

Proof: Follows from theorem 2.6.1, with the correspondences as in (167). The proof is similar to that for theorem 6.5.1. ∎

6.6.2 The Peaceman–Rachford variant

In case each function $f_{[i]}$ is strictly convex and differentiable we can also use the Peaceman–Rachford variant. At each iteration $t = 0, 1 \ldots$ each block computes $x_{[i]}^{t+1}$ as the solution to

$$\begin{array}{ll} \underset{x_{[i]} \in \mathbb{R}^{n_i}}{\text{minimize}} & f_{[i]}(x_{[i]}) + {p_{[i]}^{t}}^{T} x_{[i]} + \frac{1}{2}\lambda\left\|x_{[i]} - y_{[i]}^{t}\right\|_2^2 \\ \text{subject to} & A_{[i]}x_{[i]} = b_{[i]} \\ & 0 \le x_{[i]} \le u_{[i]} \end{array} \tag{191}$$

and then computes intermediate multipliers

$$p_{[i]}^{t+\frac{1}{2}} := p_{[i]}^{t} + \lambda(x_{[i]}^{t+1} - y_{[i]}^{t}) \tag{192}$$

which are then used in the objective function of the second subproblem

$$\begin{array}{rcll} y^{t+1} & \in & \underset{y}{\operatorname{argmin}} & -{p^{t+\frac{1}{2}}}^T d + \frac{1}{2}\lambda \left\| x^{t+1} - y \right\|_2^2 \\ & & \text{subject to} & \sum_{i=1}^{K} D_{[i]} y_{[i]} \leq \mathtt{d} \end{array} \tag{193}$$

Finally, there is one more multiplier update

$$p_{[i]}^{t+1} = p_{[i]}^{t+\frac{1}{2}} + \lambda(x_{[i]}^{t+1} - y_{[i]}^{t+1}) \tag{194}$$

The Karush–Kuhn–Tucker conditions for (193) yield that

$$\begin{array}{rcl} y_{[i]}^{t+1} & = & x_{[i]}^{t+1} + \frac{1}{\lambda}\left(p_{[i]}^{t+\frac{1}{2}} - {D_{[i]}}^T \mu^{t+1}\right) \\ & = & 2x_{[i]}^{t+1} - y_{[i]}^{t} + \frac{1}{\lambda}\left(p_{[i]}^{t} - {D_{[i]}}^T \mu^{t+1}\right) \end{array} \tag{195}$$

in which μ^{t+1} is the solution to

$$LCP\ \left(\ \lambda\left[d - 2Dx^{t+1} + Dy^t\right] - DD^T\mu^t,\ DD^T\ \right) \tag{196}$$

Substituting $y_{[i]}^{t+1}$ from (195) in the p update (194) yields that

$$p_{[i]}^{t+1} = {D_{[i]}}^T \mu^{t+1} \tag{197}$$

and thus we have the reverse-order update

$$y_{[i]}^{t+1} = 2x_{[i]}^{t+1} - y_{[i]}^{t} + \frac{1}{\lambda}\left(p_{[i]}^{t} - p_{[i]}^{t+1}\right) \tag{198}$$

Also for this variant one has a strong convergence result.

Theorem 6.6.2 (Primal convergence) *Let assumption 3.2.1 hold, and let the functions $f_{[i]}$ $i = 1, \ldots, K$, be strictly convex and differentiable. Then, for any starting point (p^0, y^0) and any $\lambda > 0$, $\left\{x_{[i]}^t\right\}$ converges to an optimal primal solution for (**CBA**).*

Proof: Follows from theorem 2.6.2, with the correspondences as in (167). The proof is similar to that for theorem 4.6.1. ∎

6.6.3 A unifying view

We proved in section 4.10 that for the (**ARP**) splitting the ADI (Douglas–Rachford) method, the relaxation variant and Peaceman–Rachford are a family parameterized by a scalar. This is also the case for the (**AP**) splitting, as the following lemma shows.

Lemma 6.6.3 *For the (**AP**) splitting for the (**CBA**) problem, the Peaceman–Rachford method is the relaxed Douglas–Rachford method with over-relaxation factor 2.*

Proof: Both methods solve the same first subproblem at each iteration. We can write the y update for arbitrary initialization as

$$y^t_{[i]} = \omega x^{t+1}_{[i]} + (1-\omega) y^t_{[i]} + \frac{1}{\lambda}\left(p^t_{[i]} - D_{[i]}{}^T \mu^{t+1}\right)$$

For $\omega = 1$ we get the Douglas–Rachford update (177); for $\omega = 2$ we get the Peaceman–Rachford update (195); for $\omega \in (0,2) \setminus \{1\}$ we get the relaxed update (187).

All three methods update the p multipliers by

$$p^{t+1} = D_{[i]}{}^T \mu^{t+1}$$

in which μ^{t+1} solves the following problem

$$LCP\left(\lambda\left[d - \omega \sum_{i=1}^{K} D_{[i]} x^{t+1}_{[i]} - (1-\omega)\sum_{i=1}^{K} D_{[i]} y^t_{[i]}\right] - DD^T \mu^t,\ DD^T\right) \quad . \blacksquare$$

Chapter 7

Computational results

7.1 Overview

In this chapter we report computational experience with the three splitting algorithms on the Connection Machine CM-5 for both the Douglas–Rachford and the Peaceman–Rachford methods. We are mainly interested in comparing relative performance in terms of number of iterations and total computational time to achieve a prescribed solution accuracy, as well as comparing against a standard serial package, MINOS 5.4, and parallel implementations of other decomposition algorithms.

In section 7.2 we briefly describe the CM-5, with emphasis on the communication features that allow for an efficient implementation of the splitting algorithms. In section 7.3 we present a suite of three hundred randomly generated linear and quadratic multicommodity network test problems. In section 7.4 we discuss implementation issues, including heuristics and termination criteria. Finally, in section 7.5 we present and discuss the computational results, and the related issues of parallel efficiency, speedup and scalability.

7.2 The Connection Machine CM-5

The latest parallel supercomputer from Thinking Machines Corp., the Connection Machine CM-5, is a scalable MIMD system, that can also execute SIMD programs. It can contain between 32 and 16,384 processing elements (64 for the configuration at the Computer Sciences Department of the University of Wisconsin–Madison). Each PE consists of a 32-MHz SPARC processor, 32 Mbytes of physical memory and four vector units, each capable of delivering a peak performance of 32 megaflops for either floating point, integer, or logical calculations. Control processors (typically Sun 4 workstations) execute "front-end" tasks such as system administration and serial computations. Input and output is provided via high-bandwidth I/O interfaces to graphics devices, mass secondary storage and high-performance networks. Additional low speed I/O is provided by Ethernet connections to the control processors.

The PEs, control processors and I/O interfaces are interconnected by three networks: a data network, a control network, and a diagnostic network. The data network provides point-to-point communication via transmission of message packets. It has a quad "fat-tree" structure: the PEs, control processors and I/O interfaces are located at the leaves of the tree, and the internal nodes are packet switches. The bandwidth of the tree increases as one ascends from the leaves to the root, and messages between PEs are routed via their least common ancestor: this reduces "blocking" in packet transmission.

The control network (a full binary tree with PEs as leaves) provides global cooperative functions, such as broadcast, barrier synchronization, and combining operations such as reduction and forward and backward scans. The reduction operation is of particular interest in the implementation of the ADI algorithms. It combines messages from each PE with logical, addition or maximum operators on 32-bit words. The values provided by all PEs are combined, and a copy of the global result is delivered to each.

The machine is currently running the CMOST Version 7.3 Beta.2.7 operating system. For interprocessor communication we have employed version 3.1 of the CMMD message-passing library, which includes the `vector_reduce` functions.

Our description of the CM-5 architecture is based on the article of Leiserson et al. [61], which also discusses the philosophy behind the design of the interconnection networks. The article by Best et al. [12] presents the parallel I/O modes of the machine, while the article by Hillis and Tucker [49] provides a general overview and presents examples of current applications of the CM-5. For details the interested reader can also refer to the CM-5 technical summary [96] and the CMMD library guide [97].

7.3 The suite of test problems

To assess the relative performance of the three splittings we used MNETGEN [4], a derivative of NETGEN [57], to generate three hundred random multicommodity network problems (**MC**), one hundred with linear objectives and two hundred with quadratic objective functions. The quadratic objective functions have the form

$$\sum_{i=1}^{K} \left\{ {c_{[i]}}^T x_{[i]} + r \left\| x_{[i]} - \hat{x}_{[i]} \right\|_2^2 \right\} \tag{199}$$

in which r is a positive scalar and $\hat{x}$ is a solution of (**MC**) using only the linear objective terms and discarding the coupling constraints, i.e. each $\hat{x}_{[i]}$, $i = 1, \ldots, K$ solves

$$\begin{array}{ll} \min\limits_{x_{[i]}} & {c_{[i]}}^T x_{[i]} \\ \text{subject to} & A_{[i]} x_{[i]} = b_{[i]} \\ & 0 \leq x_{[i]} \leq u_{[i]} \end{array} \tag{200}$$

For (**MC**) the constraint matrices $A_{[i]}$ and $D_{[i]}$ have a special, sparse structure: each column of $A_{[i]}$ has exactly two non-zero entries, and each matrix $D_{[i]}$ consists of columns with one $+1$ entry and of zero columns. A crucial property of the $D_{[i]}$ matrices for (**MC**) is that the matrix $D_{[i]} {D_{[i]}}^T$ is also diagonal and invertible. Thus, in the (**AP**) splitting, if we penalize with a diagonal matrix H^t, then the matrix $D(H^t)^{-1} D^T$ is diagonal and invertible: then a closed form solution for the Linear Complementarity Problem (175) exists, and is given by (176).

Although we mainly experimented with (**MC**) problems, our parallel code is written for the general (**CBA**) problem: we do not take into account the special structure of $A_{[i]}$ and $D_{[i]}$, other than treating them as sparse. This enables the code to handle much more general convex block–angular problems. In the case of (**MC**) one might be able to solve the quadratic network subproblems faster by employing specialized algorithms, such as the two-segment linearization simplex method of Meyer and Kamesam [54] or the quasi-Newton method of Toint and Tuyttens [100], [99].

Table 1 displays the characteristics of the test problems. Five instances were randomly generated for each case; the table reports average characteristics. The column labeled "LP size" refers to the size of the multicommodity problem formulated as a full Linear Problem; "Constraints" is the total number of rows in the constraint matrix for this LP, excluding the box constraints, and "Variables" is the total number of LP variables, excluding slacks.

The number of blocks for our test problems varies between 4 and 31, and therefore each PE of the CM-5 is assigned exactly one block. The number of network nodes (rows) per block varies between

Multicommodity network problems						
Problem id	Blocks	Subproblem size		Coupling	Equivalent LP size	
		Nodes	Arcs	constraints	Constraints	Variables
1	4	50	111	65	265	444
2	8	50	120	82	482	960
3	16	50	120	88	888	1920
4	31	50	121	83	1633	3751
5	4	100	227	139	539	908
6	8	100	230	142	942	1840
7	16	100	241	148	1748	3856
8	31	100	244	151	3251	7564
9	4	200	450	301	1101	1800
10	8	200	450	288	1888	3600
11	16	200	450	316	3516	7200
12	31	200	449	306	6506	13919
13	4	300	683	432	1632	2732
14	8	300	687	435	2835	5496
15	16	300	713	451	5251	11408
16	31	300	682	436	9736	21142
17	4	400	844	525	2125	3376
18	8	400	839	543	3743	6712
19	16	400	818	520	6920	13088
20	31	400	832	538	12938	25792

TABLE 1: *Test problem characteristics.*

50 and 400, while the number of arcs (variables) varies between 111 and 844. The number of coupling constraints is proportional to the number of nodes and varies between 65 and 543. The networks have 60% to 75% of their arcs capacitated; 55% to 70% of the arcs participate in the coupling constraints. All linear cost coefficients are positive numbers in the range 1 to 100, generated randomly. Two values of r were used, 0.05 and 0.5.

7.4 Implementation

7.4.1 General issues

The CM-5 can be programmed in both a SIMD and a MIMD mode. In the former, the control processor regulates the execution of the program; it executes the serial part, and distributes, via broadcasts, sections of the data-parallel part to the PEs, which then execute it locally. The PEs are synchronized via "barriers" – designated rendezvous points in the code. In the MIMD mode each PE executes a local program; since the system has no shared memory, all exchange of information between PEs is done via message passing.

In all three ADI algorithms, the decomposition is such that in the *fork* phase PEs can work independently, each solving a block subproblem. The global coordination *join* phase requires calculating the updates via adding or averaging vectors that are spread over the PEs. The aggregated values can be computed on the CM-5 with a global call to the `CMMD_vector_reduce` function. What needs to be reduced in all cases is a collection of vectors of size m_0, i.e. the number of coupling constraints. Each PE will contribute to the reduction the vectors $d_{[i]}^{t+\frac{1}{2}}$ for (**ARP**), and $D_{[i]}x_{[i]}^{t+1}$ for (**RP**) and (**AP**).

For efficiency, we used a master-less MIMD approach in our final code; it had been our experience that computation involving the control processor introduces a communications overhead, that can result in reduced speedup.

The solution of the quadratic block subproblems was obtained using the standard optimization package MINOS 5.4 ([76], [77]). MINOS 5.4 uses a reduced gradient method to solve problems with nonlinear objective and linear constraints. Since the feasible region for the block constraints remains the same over all iterations, "warm start" techniques were used, and the optimal solution at iteration t was employed as the starting point for iteration $t+1$.

MINOS 5.4 can treat matrices as sparse. Since there is considerable sparsity in the (**MC**) problems, we found this to be a significant advantage: computational experience with the LSSOL package [40], which employs a specialized two-phase active-set method to solve quadratic problems but treats matrices as dense, has shown sparse MINOS 5.4 to be much faster than LSSOL for larger problems.

Our code is written in ANSI C; it is a modern language with a variety of control constructs and data types. MINOS 5.4 is written in FORTRAN 77. In our implementation we do not make use of the CM-5 vector units.

7.4.2 Fixed versus variable penalty

In the basic ADI method a single positive penalty scalar is used and is kept fixed over all iterations. In this case the performance of the algorithm depends strongly on the choice of the penalty. To illustrate this, we ran the (**RP**) algorithm of the Douglas–Rachford method with fixed penalty on one of the five instances of the linear multicommodity problem 7. The penalties chosen were in the

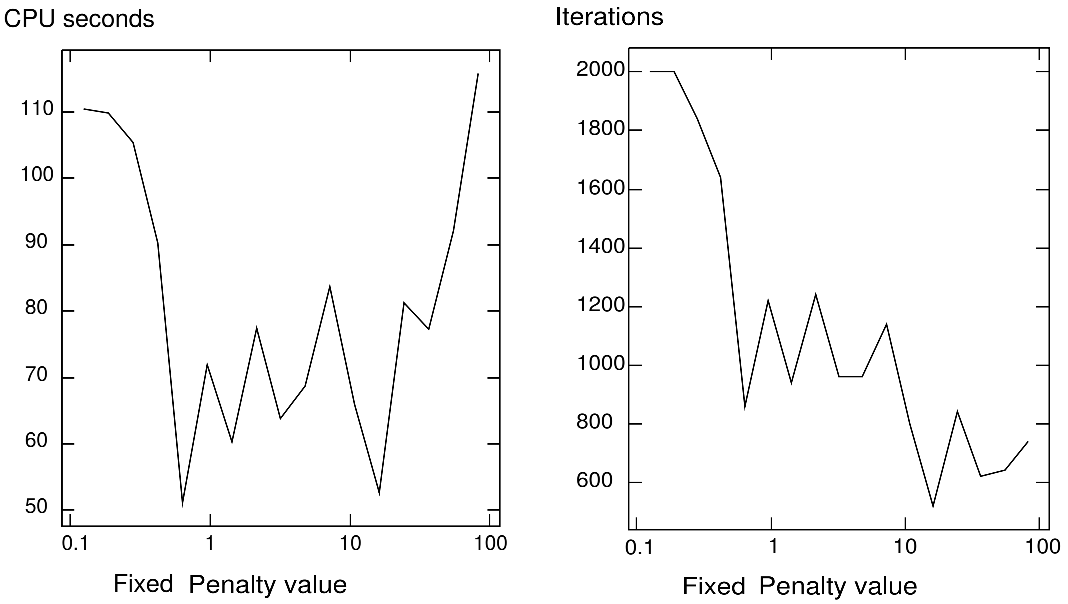

FIGURE 4: *CPU seconds (left) and number of iterations (right) for the* (**RP**) *splitting with* **fixed** *penalty, on an instance of the linear multicommodity problem 7. (Results for the variable penalty heuristic are presented in Table 2.)*

range 0.1 to 100. In Figure 4 we display the effect of the penalty choice on the CPU seconds on the CM-5 and on the number of iterations to termination.

We observe that if the penalty is chosen too small or too large the solution time can significantly increase. This can be attributed to the following: if the penalty λ is too large, the proximal terms in the objective function of the block subproblems (12) and (13) outweigh the "true" objective and thus a small step in the primal space may be taken; on the other hand, as the update (14) shows, a large step will be taken in the dual space of multipliers. In the case where the penalty is too small, a large step might be taken in the primal space but a small step will be taken in the dual space. Large penalty values also cause the subproblem minimization to become ill-conditioned and harder to solve; as a result, the average time per *fork-join* iteration increases, although the number of iterations decreases, in general. For problem 7, good choices for the penalty value in (**RP**) would be 0.6, resulting in 51 CPU seconds and 860 iterations, and 16, resulting in 52 CPU seconds and 520 iterations.

We found that the case reported in Figure 4 is typical for (**MC**) problems. Similar experience for a variety of applications is reported by Fukushima [36], Glowinski and Marrocco [34, chapter 5], and Eckstein [25, chapter 7].

Since it is not, in general feasible, for any given optimization problem, to determine a good value of the penalty λ a priori, one has to resort to ad hoc techniques or to heuristics which can allow initial large steps in both the primal and dual space, such as Eckstein's "twin lambda" [25, section 7.2.4]: in this heuristic the initial primal penalty is small, 0.1 λ, and is slowly increased to the final value λ, while the penalty used in the initial dual step is large, 10 λ, and is slowly decreased to the final value λ.

Allowing the penalty term to vary per iteration by the heuristics we describe in the following section, we were able to solve the above problem 7 using (**RP**) in 240 iterations that took 21.3 CPU seconds.

7.4.3 Heuristics

Our code uses diagonal positive matrices as penalty factors for all splittings. For simplicity in the computations, we also use canonical initialization.

In the initial iterations we want the algorithm to make good progress towards a solution, and not be too constrained by the proximal terms: thus we initially allow for inexact solution of the subproblems, we use over-relaxation and we let the penalty factors increase only slowly.

7.4.3.1 Selection of initial values

We found that the choice of the starting point x^0 and of the initial proximal terms, multipliers and penalty matrices can be quite important. In all three splittings x^0 was chosen as a solution of (**MC**) using only the linear objective terms and discarding the coupling constraints, i.e. as a solution of problem (200). The choice of initial proximal vectors was based on x^0, as we now describe in detail for each splitting.

(ARP) : We need to assign values to d^0, the initial allocation of the shared resource $\mathtt{d}$ to the K blocks, which has to be such that $\sum_{i=1}^{K} d^0_{[i]} = \mathtt{d}$. The simplest choice is equipartition.

$$d^0_{[i]} = \frac{1}{K}\mathtt{d}$$

We can also allocate to each block the amount that the initial vector $x^0_{[i]}$ consumes, adjusted by the equipartition of the global surplus or deficiency.

$$d^0_{[i]} = D_{[i]} x^0_{[i]} + \frac{1}{K}\left[\mathtt{d} - \sum_{i=1}^{K} D_{[i]} x^0_{[i]}\right] \tag{201}$$

The scheme we finally implemented equipartitions surplus as above, but partitions deficiency in proportion to the consumption of the shared resource, as shown in the expression below, which is to be interpreted component-wise.

$$d^0_{[i]} = D_{[i]} x^0_{[i]} - \frac{D_{[i]} x^0_{[i]}}{\sum_{i=1}^{K} D_{[i]} x^0_{[i]}} \left[\sum_{i=1}^{K} D_{[i]} x^0_{[i]} - \mathtt{d}\right] = \frac{D_{[i]} x^0_{[i]}}{\sum_{i=1}^{K} D_{[i]} x^0_{[i]}}\, \mathtt{d} \tag{202}$$

The initial multipliers are chosen proportional to the violation of the coupling constraints.

$$p^0 = H^0_d \left[\sum_{i=1}^{K} D_{[i]} x^0_{[i]} - \mathtt{d}\right]_+ \tag{203}$$

The diagonal penalty matrices H_d are initialized as follows: we determine a reference penalty value λ_{ref}, using a multiple of the largest linear cost coefficient,

$$\lambda_{\mathrm{ref}} := \theta\ \|c\|_\infty \tag{204}$$

for $\theta = 0.25$, after experimentation, and $\|c\|_\infty = 100$. Then we initially penalize the resource allocations uniformly

$$H^0_d = \lambda_{\mathrm{ref}}\ I$$

and the primal variables in proportion to the corresponding linear cost coefficient.

$$H^0_{x[i]}\,(j,j) \ = \ 0.1\,\theta\, c_{[i]}\,(j), \quad j = 1, \ldots, n_i$$

(RP) : We also chose $\theta = 0.25$, and d^0 as in (201) and (202) for (**ARP**). We set

$$H^0 = 0.08\ \lambda_{\mathrm{ref}}\ I$$

and chose p^0 as in (203).

(AP) : We chose $\theta = 0.15$, and set $\mu_0 = 0$ and $y^0 = x^0$. For all primal variables not participating in the coupling constraints we set the corresponding penalty to a small positive value, that is never updated. This value can also be zero, in which case the algorithm converges to the optimal value for (**MC**) without guaranteed convergence for these variables. For the remaining variables $x_{[i]}(j)$ we chose, as for (**ARP**),

$$H^0_{[i]}\,(j,j) \ = \ 0.1\,\theta\, c_{[i]}\,(j), \quad j = 1, \ldots, n_i$$

7.4.3.2 Update of the penalty matrices

We experimented with several schemes for the updates. The best results for this class of problems were obtained with an updating scheme similar to the ones used in the method of multipliers for the Augmented Lagrangian algorithm: a penalty parameter is increased when the violation in the corresponding coupling constraint is not sufficiently reduced [69], [10, chapter 2], [78]. In our implementation all penalty parameters remain within positive bounds, as required by the convergence theory. In the initial iterations we allow for reduced accuracy in solving the subproblems, and also employ over-relaxation.

(ARP) : The penalty factors are updated every $T = 15$ iterations. For all primal variables not participating in the coupling constraints the penalty is never updated; for the rest, it is slowly increased up to a bound.

$$H^t_{x[i]}\,(j,j) \;=\; \min\left\{\lambda_{\text{ref}},\; 1.05\; H^{t-T}_{x[i]}\,(j,j)\right\}$$

For the first 100 iterations the update for the resource allocation penalty is similar.

$$H^t_d\,(j,j) \;=\; \min\left\{2\;\lambda_{\text{ref}},\; 1.05\; H^{t-T}_d\,(j,j)\right\}$$

From there on, the update is based on the progress made in the last T iterations towards satisfying the coupling constraints. We define the vector of violation v^t as

$$v^t := \left[\sum_{i=1}^{K} D_{[i]} x^0_{[i]} - \mathtt{d}\right]_+ \tag{205}$$

Then, for any coupling constraint $j = 1, \ldots, m_0$: if it is satisfied at iteration t, we lower the corresponding penalty by a factor of three.

$$H^t_d\,(j,j) \;=\; \max\left\{H^{t-T}_d\,(j,j)/3,\; 0.01\right\}$$

Else, we triple the penalty only if the constraint violation is more than a third of its previous value.

$$H^t_d\,(j,j) = \begin{cases} \min\left\{8\;\lambda_{\text{ref}},\; 3\; H^{t-T}_d\,(j,j)\right\} & \text{if } v^t \geq 0.33\; v^{t-T} \\[2ex] H^{t-T}_d\,(j,j) & \textit{otherwise} \end{cases} \tag{206}$$

We perform the first 50 iterations with an over-relaxation factor of 1.5, and we set in MINOS, for the first 100 iterations, a feasibility tolerance of 10^{-3}.

(RP) : In the first 80 iterations an over-relaxation factor of 1.3 is used. For the first 100 iterations a feasibility tolerance of 10^{-3} is used in MINOS. The penalty matrices are updated every $T = 15$ iterations. The updates scheme is similar to the one employed for the H_d^t matrices for (**ARP**): In the first 100 iterations the penalty is slowly increased uniformly, up to a bound.

$$H^t\ (j,j) \ = \ \min\left\{2\ \lambda_{\mathrm{ref}},\ 1.1\ H_d^{t-T}\ (j,j)\right\} \tag{207}$$

Then, for any coupling constraint $j = 1, \ldots, m_0$: if it is satisfied at iteration t, we lower the penalty.

$$H^t\ (j,j) \ = \ \max\left\{H^{t-T}\ (j,j)/3,\ 0.05\right\} \tag{208}$$

Else, we triple the previous penalty only if the decrease in the violation is not satisfactory, as in (206).

(AP) : In the first 80 iterations an over-relaxation factor of 1.3 is used. For the first 100 iterations a feasibility tolerance of 10^{-3} is used in MINOS. The penalty matrices are updated every $T = 10$ iterations. For any primal variable $x_{[i]}\ (k)$ that participates in a coupling constraint, the penalty in the first 100 iterations is slowly increased, as in (207)

$$H_{[i]}^t\ (k,k) \ = \ \min\left\{2\ \lambda_{\mathrm{ref}},\ 1.1\ H_{[i]}^{t-T}\ (k,k)\right\} \tag{209}$$

From there on, the penalty is updated by (208) and (206), for j the index of the coupling constraint in which the variable $x_{[i]}\ (k)$ participates.

In our computational experience, the effect of these strategies has been to fix the penalties soon after the initial iterations of increasing penalties, for all three splittings. The final penalty is the lower bound, for the satisfied constraints, and the upper bound, for the unsatisfied ones. In order to match our assumption of ultimately fixed penalties, a parameter could be set limiting the number of iterations during which changes in the penalties would be allowed (for our problems, a value of 400 iterations would have sufficed).

7.4.4 Termination criteria

We chose to terminate the algorithm at iteration t when the following condition (i) and at least one of conditions (ii) and (iii) are met:

(i) The primal iterates satisfy the block constraints within a tolerance of ϵ_1 and the coupling constraints within ϵ_2.

(ii) The proximal terms for iteration t and $t - T$ differ by less than ϵ_2.

(iii) The primal iterates, the duals from the block subproblems and the p multipliers satisfy the Karush–Kuhn–Tucker optimality conditions for (**MC**) within ϵ_2.

We chose $\epsilon_1 = 10^{-8}$, $\epsilon_2 = 10^{-5}$ and T = 10. The value $\epsilon_1 = 10^{-8}$ was used as the final feasibility tolerance for MINOS.

Our difference function Δ, used in criterion (ii) and in the satisfaction of the coupling constraints in (i), is a combination of absolute difference and component–wise relative difference. For any scalars x and y, we define $\delta(x, y)$, the difference between x and y, as

$$\delta(x,y) := \begin{cases} |x| & \text{if } |y| \le \epsilon_2 \\ \left|1.0 - \frac{x}{y}\right| & \text{otherwise} \end{cases} \tag{210}$$

(The first case is a safeguard against a comparison with a number that has a modulus close to zero.) For any vectors X and $Y \in \mathbb{R}^n$, we define their difference $\Delta(X, Y)$ as

$$\Delta(X,Y) := \max_{1 \le i \le n} \delta(X(i), Y(i)) \tag{211}$$

In case the vectors have components of varying magnitude, the component-wise criterion $\Delta(X,Y) \le \epsilon$ is, in general, stricter than the standard relative difference criterion $\frac{\|X - Y\|}{\|Y\|} \le \epsilon$.

7.5 Results of numerical experiments

7.5.1 The Douglas–Rachford method

7.5.1.1 Relative performance of the three splittings

The relative performance of the three splittings for the Douglas–Rachford method is presented in Tables 2, 3 and 4, and, in a graphical way, in Figures 5, 6 and 7. In all figures each group of lines corresponds to problems with the same number of subnetwork nodes, with the number of commodities (blocks) varying from 4 to 31. The curves present average solution times and iteration counts over the five instances per problem case. In all figures the results for the (**AP**) spitting are shown in solid lines, the results for (**RP**) in thick dashed lines, while thin dashed lines correspond to results for (**ARP**). The timing results clearly demonstrate the general superiority of the (**AP**) splitting.

Figure 5 reports total CPU time in seconds for the three splittings for the linear (**MC**), and for the quadratic (**MC**) with quadratic factor $r = 0.05$ and $r = 0.5$. Timing was done via the CMMD timers, and excludes only the initial reading of block data. The solution time information is also

Problem id	CPU seconds			
	(ARP)	**(RP)**	**(AP)**	**MINOS 5.4**
1	8.42	4.89	5.94	1.96
2	23.79	10.61	11.35	7.11
3	24.39	9.64	12.23	25.84
4	36.14	16.48	18.31	109.87
5	23.78	11.08	9.72	10.44
6	58.25	17.74	18.93	35.83
7	72.55	21.30	20.43	147.62
8	92.71	25.71	22.52	709.53
9	291.83	39.10	32.74	72.13
10	318.82	64.84	57.66	224.82
11	370.01	57.91	63.81	691.15
12	721.55	121.31	109.72	2631.16
13	832.20	88.48	60.87	309.10
14	1001.66	100.27	62.34	702.89
15	1307.87	117.66	70.95	397.22
16	1836.85	129.83	95.31	952.04
17	*	117.49	68.42	397.22
18	*	125.01	83.86	952.04
19	*	132.47	97.66	3809.15
20	*	142.90	106.68	15011.13

*: did not run because of excessive memory and/or time requirements.

TABLE 2: *CPU seconds for the three splittings for the Douglas–Rachford method on the CM-5 and for MINOS 5.4 on a DEC 5000, for the* **linear** *multicommodity network problems.*

presented in Tables 2, 3 and 4. Figure 6 reports the number of *fork-join* iterations for the three splittings for the linear and quadratic (**MC**) problems. Finally the average CPU time per iteration for all cases is reported in Figure 7.

With reference to these tables and figures, we make the following observations:

- The (**AP**) and (**RP**) algorithms perform significantly better than (**ARP**), although the latter takes on the average no more iterations to optimality than either of the other two. However, the subproblem that needs to be solved at each iteration for (**ARP**) is larger than the one for the other two splittings, both in the number of variables and in the number of constraints. This significantly influences the solution time, and is responsible for the high growth rate of CPU time for (**ARP**) as the problem size increases. The (**AP**) algorithm is the best among the three: for problem 20, the largest problem in the suite, having 12,938 rows and 25,792 variables in the equivalent LP formulation, it takes the (**AP**) algorithm 106 seconds to solve

Problem id	CPU seconds			
	(ARP)	**(RP)**	**(AP)**	**MINOS 5.4**
1	7.37	5.01	4.87	2.96
2	13.41	5.85	7.38	13.38
3	17.93	8.69	6.95	41.57
4	20.40	9.16	8.69	200.97
5	22.94	9.70	9.57	13.47
6	37.69	12.55	14.70	56.63
7	53.23	16.38	16.60	271.18
8	74.12	19.52	19.93	980.23
9	185.23	32.05	25.13	108.69
10	245.73	39.00	41.04	375.11
11	285.59	38.77	33.81	1232.24
12	665.36	107.59	68.31	5795.65
13	*	76.89	56.00	619.48
14	*	83.32	52.69	1035.52
15	*	107.51	68.37	4115.52
16	*	132.51	80.93	15625.62
17	*	106.01	70.29	782.94
18	*	125.57	77.14	1470.45
19	*	120.32	86.40	6095.64
20	*	165.71	115.12	42584.73

*: did not run because of excessive memory and/or time requirements.

TABLE 3: *CPU seconds for the three splittings for the Douglas–Rachford method on the CM-5 and for MINOS 5.4 on a DEC 5000, for the* **quadratic** *multicommodity network problems with* $r = 0.05$.

the linear problem, 115 to solve the quadratic problem with factor $r = 0.05$ and 147 seconds for the $r = 0.5$ quadratic problem. The respective times for (**RP**) are 142, 165 and 204 seconds.

- For all three splittings, the number of required iterations decreases, as the quadratic factor r increases. Thus it may be possible to solve linear problems more efficiently by embedding them in a series of strongly convex quadratic ones, obtained by adding a proximal term to the linear objective function.

- As the quadratic factor r increases, the cost per iteration also increases. This can be attributed to the behavior of MINOS, which requires more inner iterations and function evaluations as the subproblem objective function becomes a steeper quadratic. Other solvers may not exhibit such a characteristic.

- As expected, for a fixed number of subnetwork nodes, the number of iterations increases, as the

Problem id	CPU seconds			
	(ARP)	**(RP)**	**(AP)**	**MINOS 5.4**
1	6.87	4.58	5.68	5.16
2	11.02	5.40	8.17	23.97
3	23.40	8.99	7.79	51.80
4	22.75	10.11	9.04	172.07
5	29.41	10.91	12.17	24.50
6	38.32	13.05	17.60	56.47
7	69.16	18.38	18.47	213.08
8	106.41	21.83	20.87	733.94
9	217.87	38.62	36.87	110.11
10	250.41	42.31	51.68	291.64
11	352.53	38.84	41.71	975.56
12	936.32	76.85	63.07	3819.22
13	*	77.11	83.18	554.16
14	*	106.21	88.14	565.31
15	*	124.24	103.94	2047.12
16	*	179.46	128.33	6343.09
17	*	123.69	113.14	782.90
18	*	173.31	119.70	907.01
19	*	179.25	143.86	3454.87
20	*	204.27	147.34	31958.88

*: did not run because of excessive memory and/or time requirements.

TABLE 4: *CPU seconds for the three splittings for the Douglas–Rachford method on the CM-5 and for MINOS 5.4 on a DEC 5000, for the* **quadratic** *multicommodity network problems for* $r = 0.5$.

number of blocks increases; this is exhibited quite dramatically in the case of the problems with 200 nodes per subnetwork (problems 9 – 11). It appears that the simple coordination of these splittings, via aggregation of subproblem vectors, becomes less effective as more subnetworks (i.e. blocks) are added. We examine scalability in more detail in section 7.5.1.6.

- For a fixed number of blocks, the number of iterations to optimality is almost independent of the size of the block subproblem.

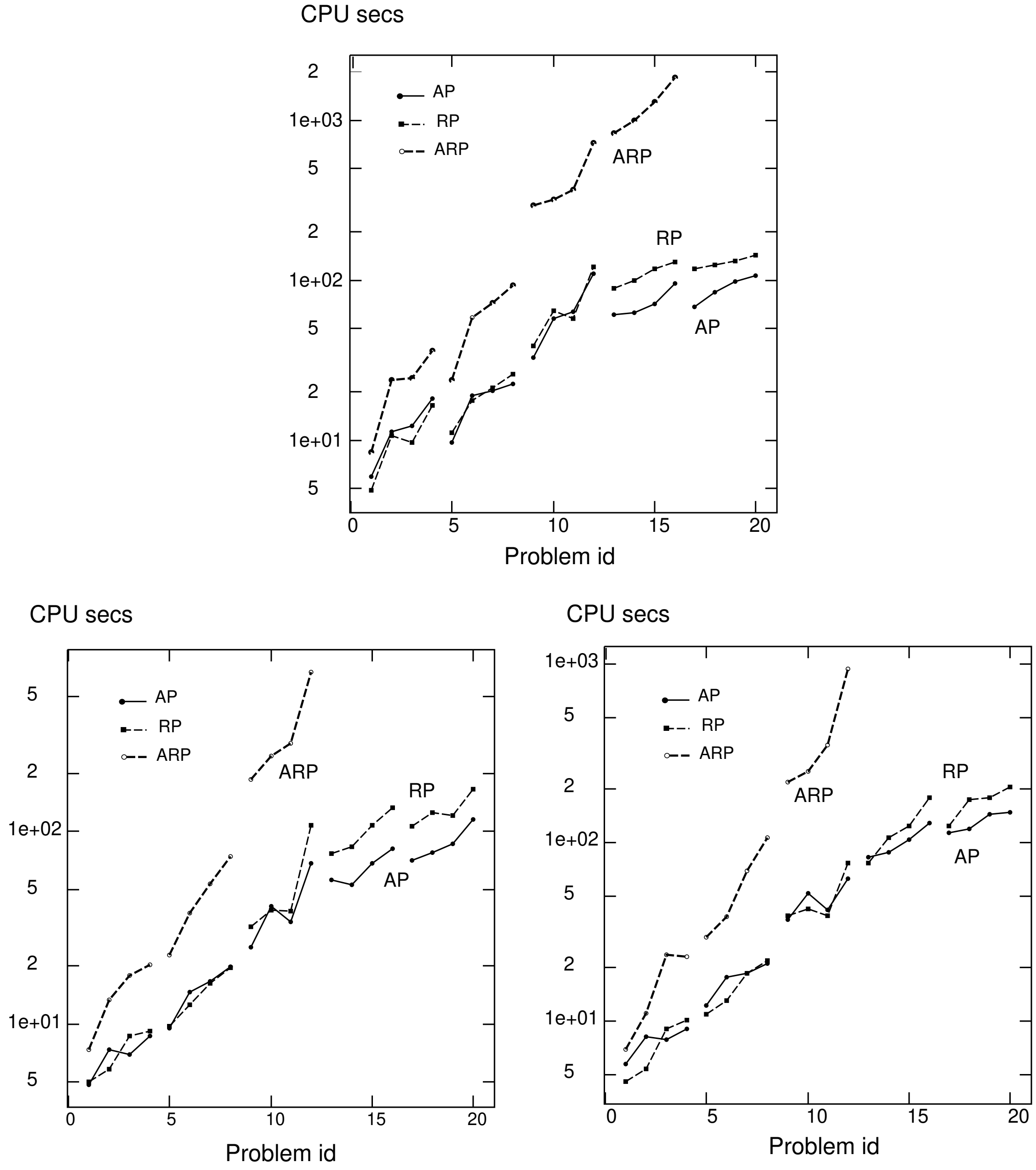

FIGURE 5: *Comparison of CPU seconds for the three splittings for the* **linear** *(top) and* **quadratic** *multicommodity problems with quadratic factors* $r = 0.05$ *(bottom left) and* $r = 0.5$ *(bottom right). (The* (**AP**) *splitting corresponds to solid lines.)*

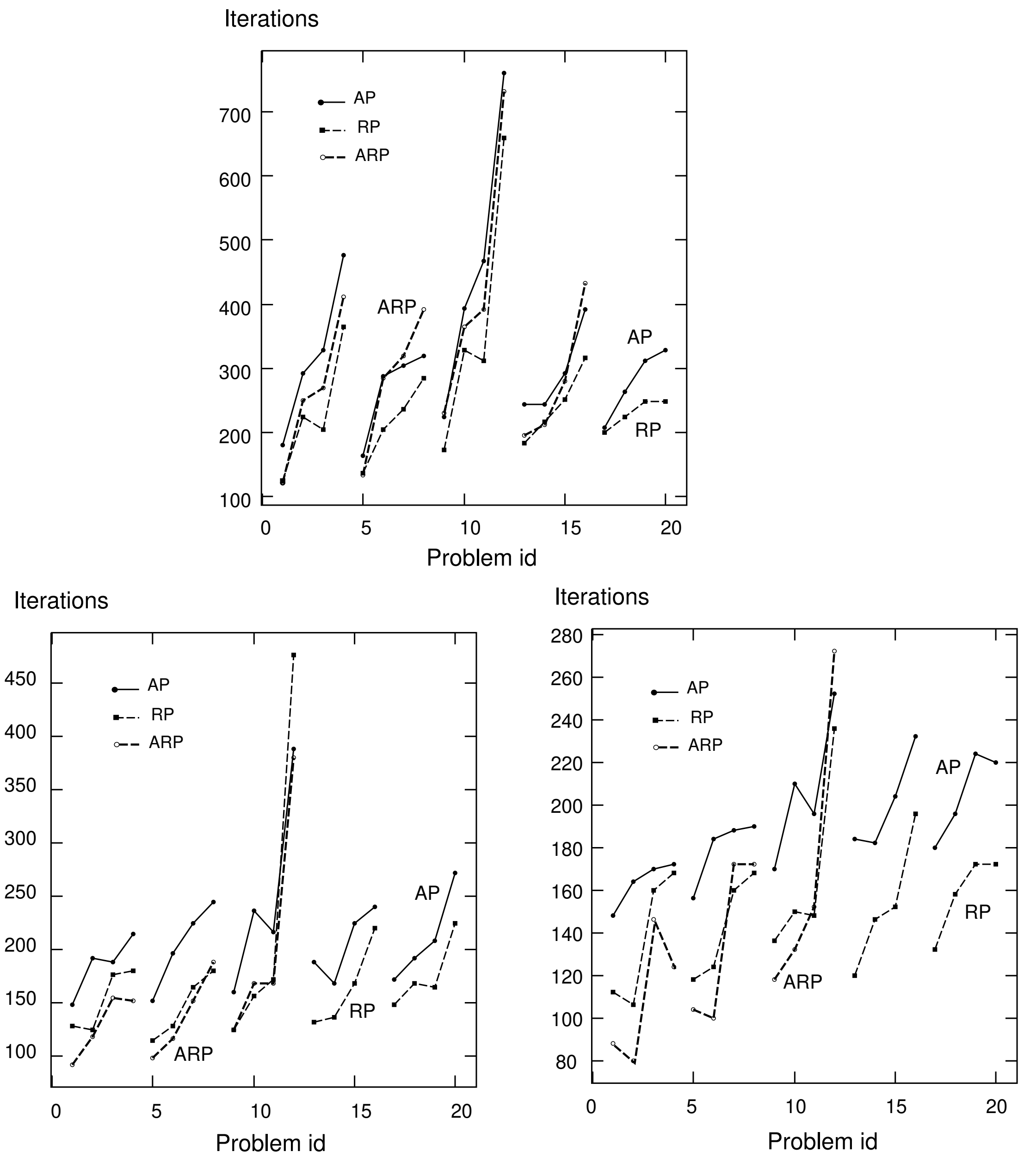

FIGURE 6: *Comparison of number of iterations for the three splittings for the* **linear** *(top) and* **quadratic** *multicommodity problems with quadratic factors* $r = 0.05$ *(bottom left) and* $r = 0.5$ *(bottom right). (The* **(AP)** *splitting corresponds to solid lines.)*

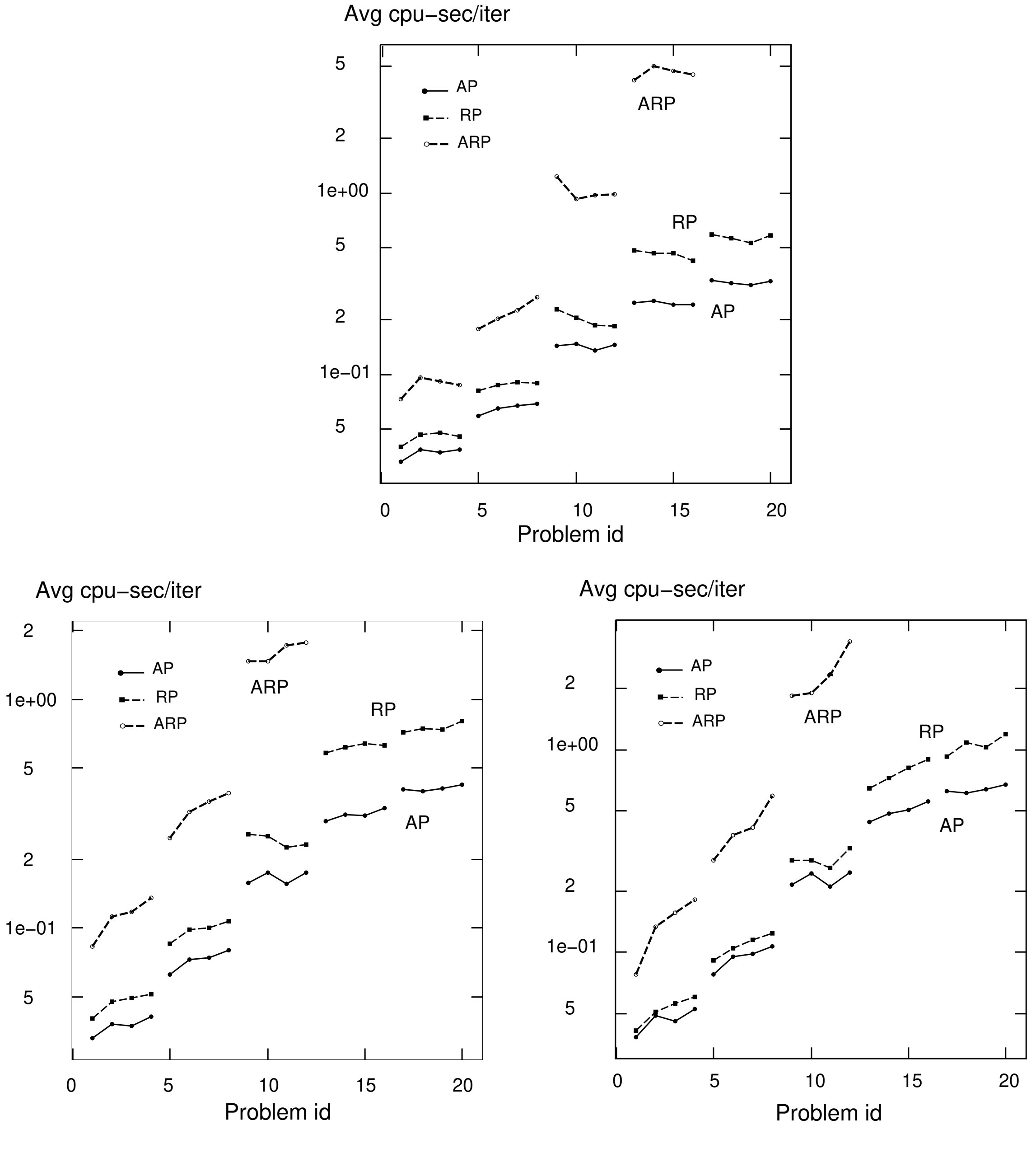

FIGURE 7: *Average CPU time spent per iteration per splitting for the* **linear** *(top) and* **quadratic** *multicommodity problems with quadratic factors $r = 0.05$ (bottom left) and $r = 0.5$ (bottom right). (The* (**AP**) *splitting corresponds to solid lines.)*

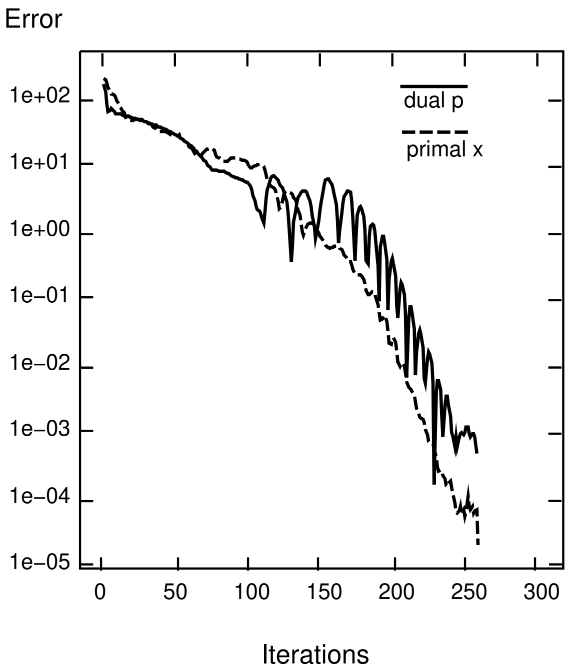

FIGURE 8: *Error magnitude in the two norm for the primal* $\{x^t\}$ *and dual* $\{p^t\}$ *iterates of the (***AP***) Douglas–Rachford algorithm for a linear instance of multicommodity problem 9.*

7.5.1.2 Convergence profile

We are also interested in the behavior of the absolute error in the primal and dual variables, $\|x^t - x^*\|_2$ and $\|p^t - p^*\|_2$. In our computational experiments we found out that these errors do not, in general, decrease monotonically, but rather display an oscillatory behavior. In Figure 8 we have plotted the magnitude of the errors for a typical case, a linear instance of (**MC**) problem 9. (In place of the optimal values x^* and p^* we used in the error calculation the values of the variables at the final iteration.) The oscillation is more pronounced in the error curve for the p multipliers and exhibits an almost regular pattern. Eckstein [25], [26] has also observed oscillations when applying the Douglas–Rachford method to monotropic network optimization.

7.5.1.3 Estimates of parallel efficiency and relative speedup

Another topic of interest is processor utilization, i.e. the fraction of total CPU time that the processors spend on "useful" work, as opposed to busy-waiting for synchronization. In order to estimate this fraction, we activated two timers per processor i. The "outer" timer measured $t_{[i]}$, the total CPU time until termination. The "inner" timer measured $t'_{[i]}$, the cumulative time spent on solving the first subproblem and on performing the updates that involve data local to the processor. The slack $t_{[i]} - t'_{[i]} \geq 0$ is the time spent in wait for synchronization and in computing the global aggregates via the `CMMD_reduce` function. The quantity

$$\widehat{s_K} := \frac{\sum_{i=1}^{K} t'_{[i]}}{\max_{i=1,\ldots,K} t_{[i]}}$$

is an (under)estimate of parallel speedup, since a single-processor implementation, sequentially simulating the computations of each of the K processors, would require a CPU time of $\sum_{i=1}^{K} t'_{[i]}$, plus the time to compute the global sums. Then, an estimate $\widehat{\mu_K}$ of parallel efficiency is

$$\widehat{\mu_K} := \widehat{s_K}/K$$

Problem id	Blocks	Efficiency	Speedup
1	4	0.791	3.164
2	8	0.680	5.440
3	16	0.664	10.624
4	31	0.574	17.794
5	4	0.737	2.948
6	8	0.730	5.840
7	16	0.663	10.608
8	31	0.541	16.771
9	4	0.699	2.796
10	8	0.660	5.280
11	16	0.624	9.984
12	31	0.524	16.244
13	4	0.681	2.724
14	8	0.615	4.920
15	16	0.594	9.504
16	31	0.518	16.058
17	4	0.652	2.608
18	8	0.577	4.616
19	16	0.588	9.408
20	31	0.489	15.159

TABLE 5: *Estimates of parallel efficiency and relative speedup for the* (**AP**) *splitting on the CM-5 applied to the linear multicommodity problems.*

In Table 5 we display these estimates of efficiency and speedup for the (**AP**) splitting applied to the linear (**MC**) problems. The table indicates that, for a fixed number of constraints in each block, the parallel efficiency decreases, as the number of blocks increases. This can be attributed to the fact that, as more subproblems are solved per iteration, the variability in the solution times increases, and thus the average wait for synchronization also increases. Also, for a fixed number of blocks, the parallel efficiency decreases, as the size of the block increases: as subproblems become larger, the solution time variability again increases.

7.5.1.4 Comparison to other decomposition methods

Deleone presents in [22] an implementation of the Dantzig-Wolfe decomposition algorithm on the CM-5. In each iteration, a designated node solves a linear *master* subproblem to determine new prices, and then the remaining *slave* processors solve in parallel linear block-subproblems to determine new columns with negative reduced costs. Only the current basis and the new columns generated by the subproblems are retained in the master. MINOS 5.4 is used to solve both master and slave subproblems.

Problem id	CPU secs
5	3.4
6	4.7
7	7.3
9	11.2
10	39.4
11	67.4
13	378.5
14	298.3
15	340.5

TABLE 6: *Timing results for the modified Dantzig-Wolfe decomposition method on the CM-5.*

Table 6 presents timing results on a subset of our linear (**MC**) test problems. Comparison with Table 2 shows that, for the three largest of these test problems, both the (**AP**) and (**RP**) splittings are faster than Dantzig-Wolfe by a factor of 2.6 to 4.5.

Medhi in [68] uses a complex coordinator based on the bundle method to solve (**CBA**) problems. It is difficult to compare our results to his, since Medhi's test problems are randomly generated and have few (10 – 50) coupling constraints. On his problems, his method takes 14 to 64 outer iterations, but those problems are very different in structure from the multicommodity problems used in our tests.

Zenios has implemented the row-action algorithm on the CM-2 for strictly convex quadratic (**MC**) problems. An 8-commodity problem with 32 nodes and 256 arcs per network was solved in 25 seconds on a 4K CM-2 [106, Table 1]. For comparison, it took 6.95 seconds on the CM-5 for the (**AP**) splitting to solve a problem with comparable size, the quadratic (**MC**) problem 3.

7.5.1.5 Comparison to MINOS 5.4

We also employed MINOS 5.4 on a 25-MHz DEC 5000 workstation to solve most of both the linear and quadratic (**MC**) problems. For linearly constrained problems, MINOS 5.4 employs the revised simplex method in the case of linear objective, and the reduced-gradient method otherwise. We set all MINOS tolerances to their default values; in particular, the default for both the feasibility and the optimality tolerance is 10^{-6}. We used a "cold start" in all cases, and treated all matrices as sparse.

The last column of Tables 2, 3 and 4 records the CPU solution times for this serial MINOS. In Figure 9 we compare CPU solution times for MINOS on the DEC 5000 versus the (**AP**) splitting on the CM-5. Objective function values at termination matched to 5–7 significant digits. These figures demonstrate that, for the larger problems, the parallel (**AP**) method is better than the serial MINOS 5.4 by one to two orders of magnitude. For instance, for problem 16, with 9,736 rows and 21,142 variables in the equivalent LP formulation, the (**AP**) algorithm requires 95 seconds to solve the linear problem, 81 seconds for the quadratic problem with $r = 0.05$ and 128 seconds for the $r = 0.5$ quadratic problem. The respective times for MINOS 5.4 on the DEC 5000 are 9,867, 15,625 and 6,343 seconds.

This gap in performance can be attributed to the fact that solution times for MINOS 5.4 demonstrate, as expected, an almost quadratic growth as a function of the size of the problem. The ADI splitting methods, on the other hand, decompose the problem into subproblems whose size may remain fixed as the overall problem size grows, and therefore solution time increases slowly as a function of problem size.

7.5.1.6 Scalability

In order to see how the algorithm scales with respect to the number of blocks in the constraint matrix, we used MNETGEN to generate sixty additional linear and quadratic test problems with 48 and 64 blocks, for subnetworks with 300 and 400 nodes. The characteristics of each class of problems are displayed in table 7. Again, five instances were randomly generated for each case; the

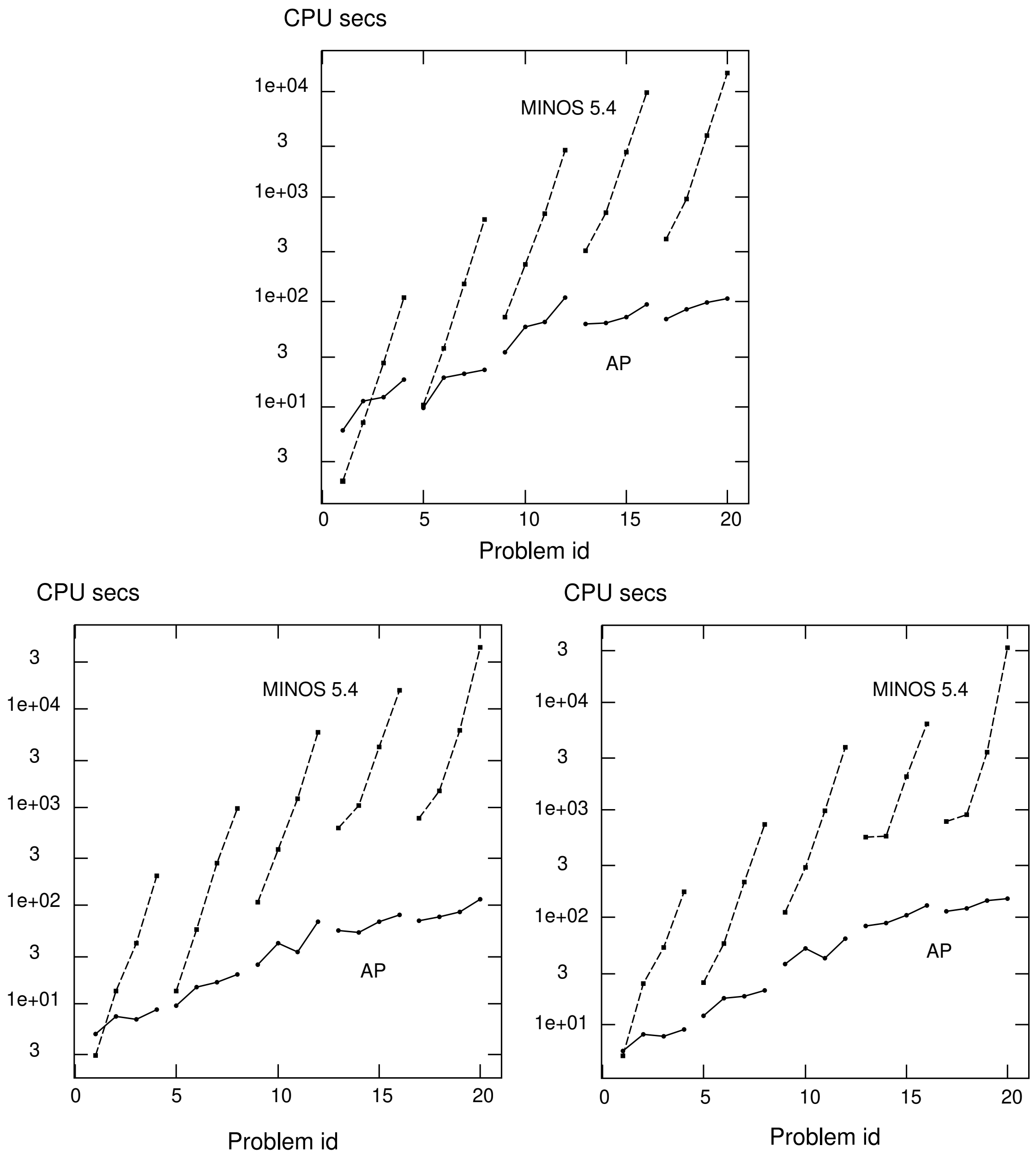

FIGURE 9: *Comparison of CPU seconds for the* (**AP**) *splitting on the CM-5 versus MINOS 5.4 on a DEC 5000 workstation, for the* **linear** *(top), and* **quadratic** *multicommodity problems with quadratic factors* $r = 0.05$ *(bottom left) and* $r = 0.5$ *(bottom right). (The* (**AP**) *splitting corresponds to solid lines.)*

table reports average characteristics. Each block was assigned to a distinct PE of the CM-5; the same heuristics and termination criteria were used in the code.

The computational results for the (**AP**) splitting on these problems are shown in Table 8. The performance of the (**AP**) splitting for all problems with 300 and 400 nodes, with blocks varying from 4 to 64, is displayed in Figures 10 (CPU solution time) and 11 (number of iterations.)

Multicommodity network problems						
Problem id	Blocks	Subproblem size		Coupling	Equivalent LP size	
		Nodes	Arcs	constraints	Constraints	Variables
21	48	300	688	432	14832	33024
22	64	300	687	449	19649	43968
23	48	400	843	551	19751	40464
24	64	400	843	554	26154	53952

TABLE 7: *Characteristics of additional test problems.*

	linear (**MC**)		$r = 0.05$		$r = 0.5$	
Problem id	CPU secs	iters	CPU secs	iters	CPU secs	iters
21	96.80	392	93.00	276	137.85	248
22	78.74	392	68.37	208	112.38	212
23	106.68	444	154.29	360	195.54	266
24	137.87	336	112.55	256	195.73	269

TABLE 8: *Computational performance of the Douglas–Rachford (**AP**) splitting on the CM-5 for the additional larger multicommodity network problems.*

With reference to these figures, we observe that the computational effort increases slowly as a function of problem size, and therefore the (**AP**) splitting has good scalability with respect to the number of blocks. Of interest is the fact that the MNETGEN problems with 64 blocks were, on the average, easier to solve than those with 48 blocks. A similar case is reported by Zakarian [105] for a nonlinear Jacobi method, and may be attributed to the fact that the solutions found for the 64-block problems had, on average, fewer positive p multipliers than the ones for the 48-block problems (22 vs. 27); this may be a characteristic of the MNETGEN generator.

7.5.2 The Peaceman–Rachford method

We also ran the Peaceman–Rachford algorithm for the three splittings on the two hundred

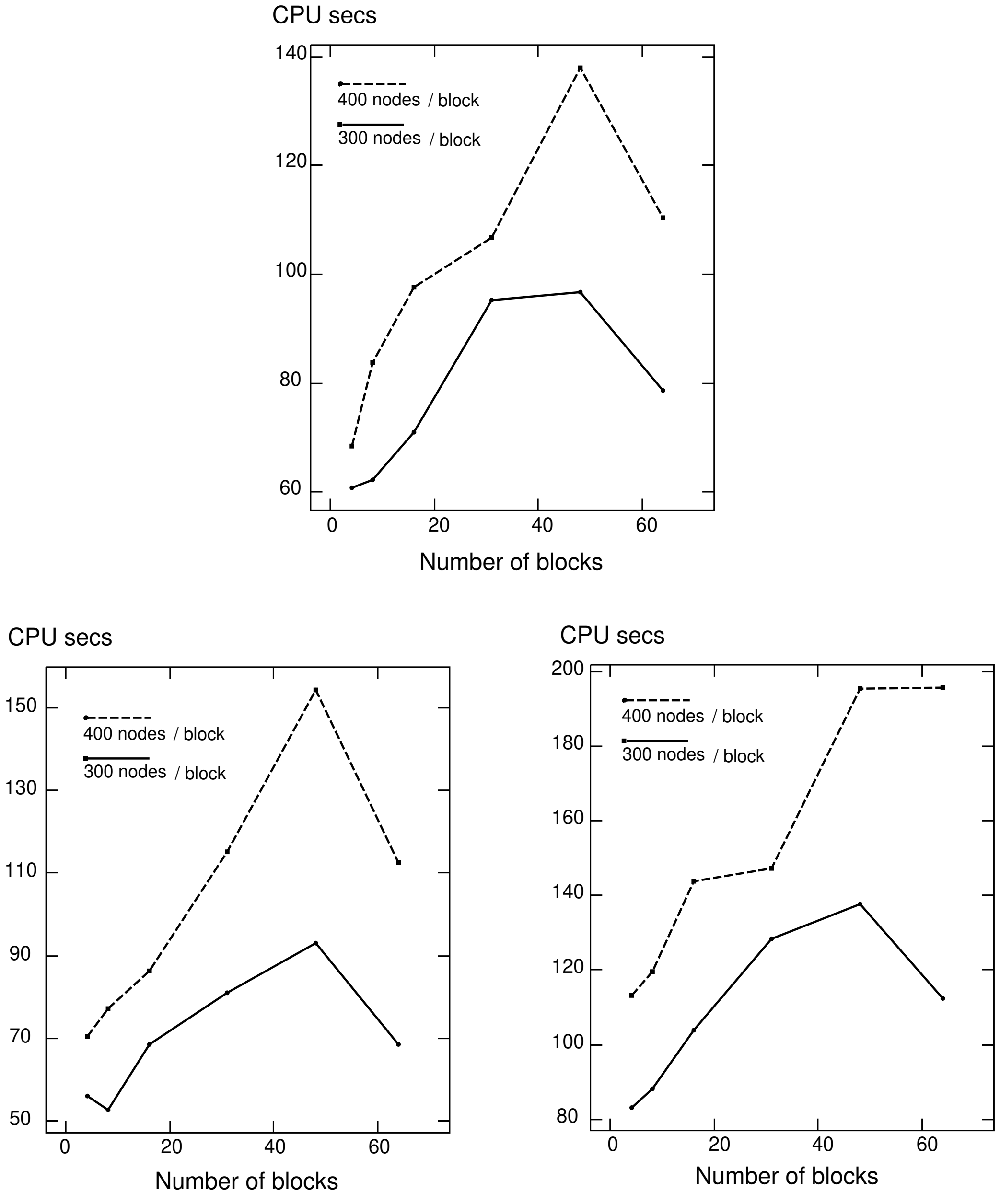

FIGURE 10: *Scalability of the (***AP***) splitting: CPU seconds versus number of blocks, for the* **linear** *(top) and* **quadratic** *multicommodity problems with quadratic factors $r = 0.05$ (bottom left) and $r = 0.5$ (bottom right).*

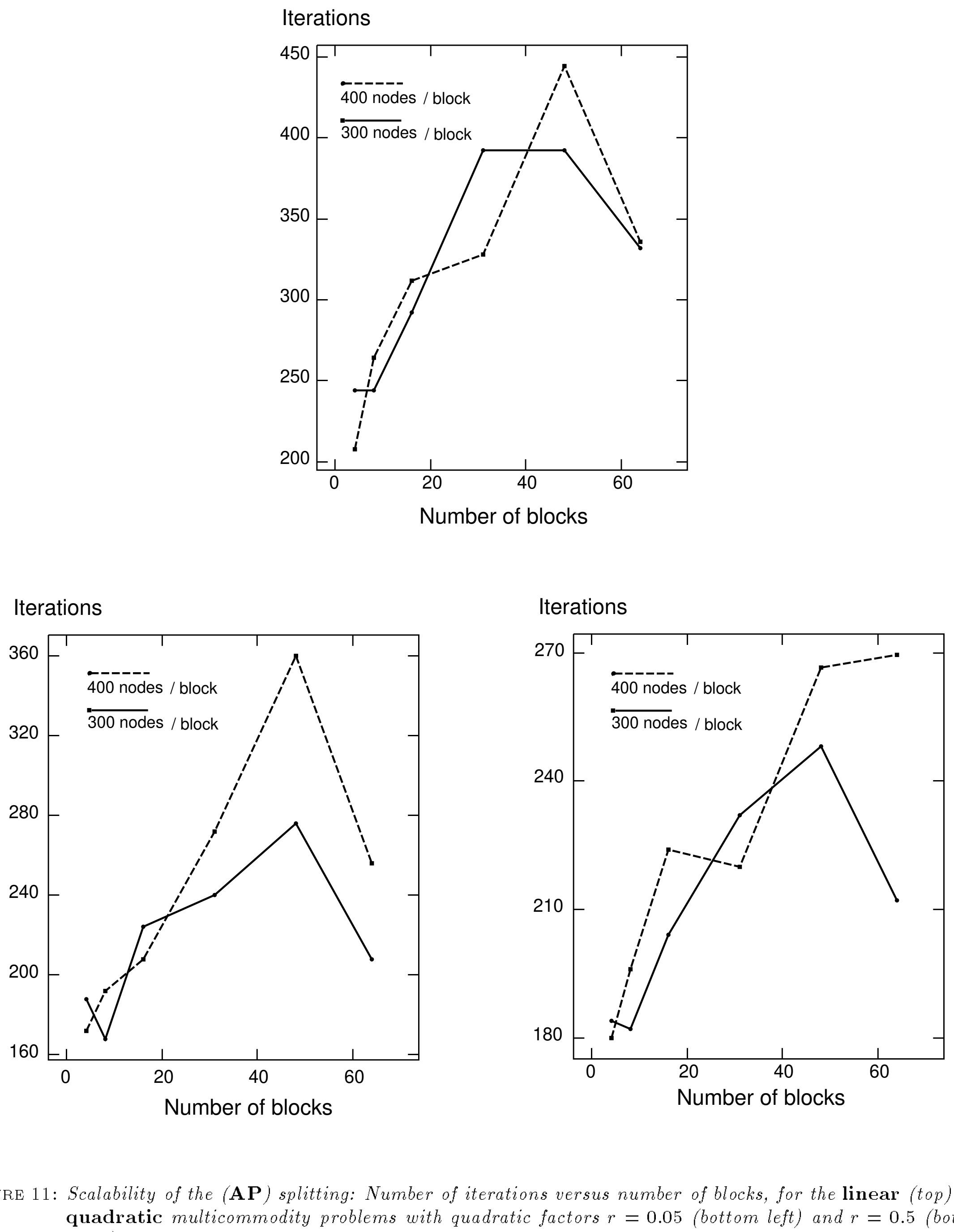

FIGURE 11: *Scalability of the (***AP***) splitting: Number of iterations versus number of blocks, for the* **linear** *(top) and* **quadratic** *multicommodity problems with quadratic factors $r = 0.05$ (bottom left) and $r = 0.5$ (bottom right).*

Problem id	CPU seconds					
	$r = 0.05$			$r = 0.5$		
	(**ARP**)	(**RP**)	(**AP**)	(**ARP**)	(**RP**)	(**AP**)
1	8.03	5.99	5.37	5.55	4.71	5.07
2	12.63	5.84	7.19	12.71	6.24	7.15
3	21.03	11.06	9.00	20.96	8.05	6.95
4	19.23	9.80	7.33	18.64	8.57	8.11
5	23.94	13.28	11.20	24.08	13.03	10.91
6	38.14	16.28	12.99	37.26	15.90	13.41
7	52.70	17.33	16.33	59.90	19.96	15.33
8	62.80	18.73	25.02	86.98	21.79	22.20
9	225.39	44.98	25.64	212.29	52.85	31.55
10	243.98	50.98	59.11	235.63	53.81	41.05
11	301.53	44.03	30.01	328.73	48.10	35.78
12	519.26	78.71	62.96	665.22	64.50	44.60
13	*	110.31	59.27	*	169.77	61.00
14	*	119.64	56.35	*	213.59	82.26
15	*	120.68	91.98	*	151.70	101.23
16	*	141.02	81.94	*	172.12	112.35
17	*	183.73	96.02	*	208.13	97.22
18	*	169.11	98.95	*	207.85	107.45
19	*	135.59	90.84	*	285.35	114.00
20	*	187.62	105.30	*	216.49	120.21

*: did not run because of excessive memory and/or time requirements.

TABLE 9: *CPU seconds for the three splittings for the Peaceman–Rachford method on the CM-5 for the* **quadratic** *multicommodity network problems for $r = 0.05$ and $r = 0.5$.*

quadratic (**MC**) problems 1–20. (Strict convexity of the objective function is required by the convergence theorem.) We implemented the algorithm as primal over-relaxation with factor 2, rather than as twice updating the dual multipliers per iteration. In both implementations, however, the update work is comparable to that for the Douglas–Rachford scheme. We also employed the same initialization and update heuristics, and the same termination criteria.

Timing results are shown in Table 9 and Figure 12. Figure 13 displays the number of *fork-join* iterations, while Figure 14 displays the average CPU time per iteration.

The conclusions we can derive from these results on the relative performance of the three splittings are similar to those for the Douglas–Rachford scheme in section 7.5.1.1. Again, the (**AP**) splitting is the faster algorithm of the three, with (**RP**) a relatively close second and the (**ARP**) not competitive for the larger problems. We also observe that the Peaceman–Rachford algorithm requires, in general, slightly fewer iterations to converge than the Douglas–Rachford algorithm, yet the aggregate solution

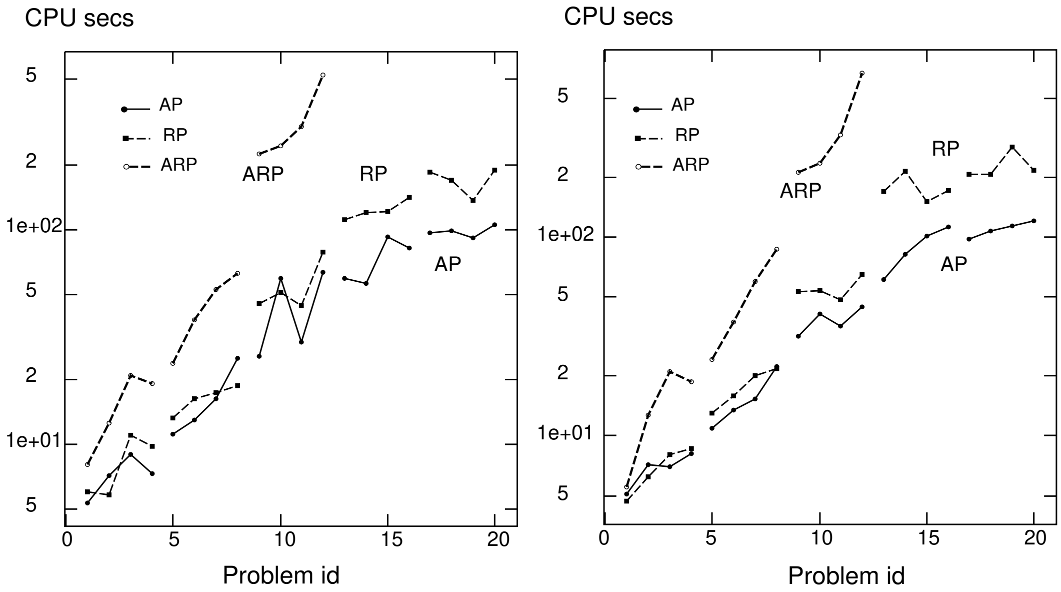

FIGURE 12: *Comparison of CPU seconds for the three splittings for Peaceman–Rachford for the* **quadratic** *multi-commodity problems with quadratic factors* $r = 0.05$ *(left) and* $r = 0.5$ *(right). (The* (**AP**) *splitting corresponds to solid lines.)*

time is similar for both schemes. Eckstein and Ferris in [29] report that the Douglas–Rachford and Peaceman–Rachford schemes require roughly the same time to converge when applied to solving monotone linear complementarity problems. There is, however an important difference between the computational performance of their algorithms and ours: their Peaceman–Rachford scheme required about twice the amount of work per iteration of their Douglas–Rachford scheme, but converged in about half the number of iterations; in our case, both the work per iteration and the number of iterations of the two splittings are comparable.

In certain problem runs we observed a numerical instability of the Peaceman–Rachford method: oscillations of small, constant magnitude in some components of the iterates. These were remedied by enforcing a Douglas–Rachford update, as a corrective step, every twenty iterations. This modified algorithm converged (i.e. termination criteria were satisfied) even in the case of the (**RP**) splitting, in which the splitting matrix has not full column rank, in general. We also ran the modified algorithm on a subset of the linear problems; it converged for some, but not for the majority of the cases.

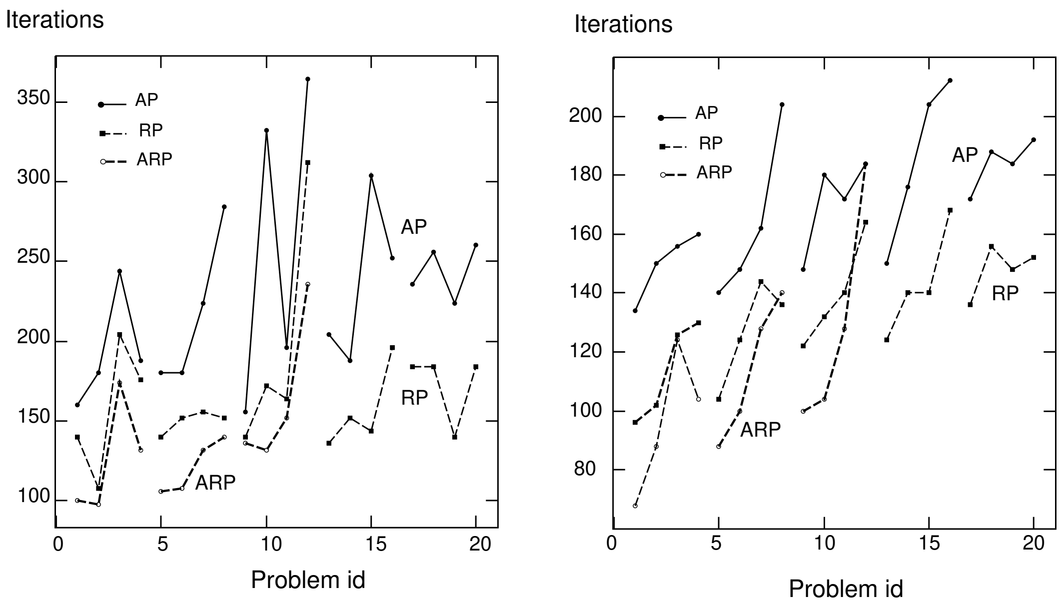

FIGURE 13: *Comparison of number of iterations for the three splittings for Peaceman–Rachford for the* **quadratic** *multicommodity problems with quadratic factors $r = 0.05$ (left) and $r = 0.5$ (right). (The* (**AP**) *splitting corresponds to solid lines.)*

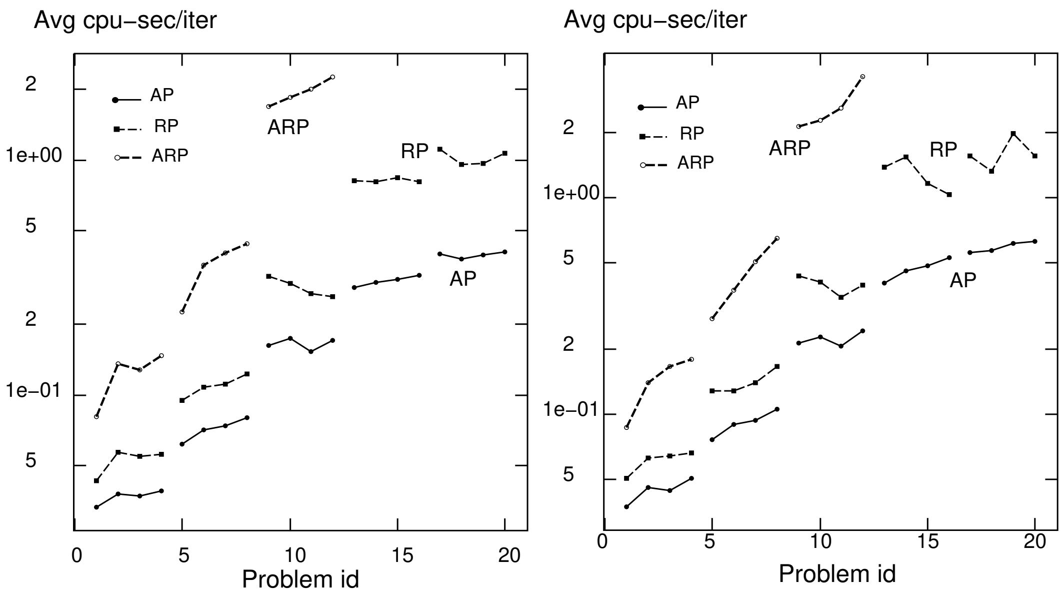

FIGURE 14: *Average CPU time spent per iteration per splitting for Peaceman–Rachford for the* **quadratic** *multicommodity problems with quadratic factors $r = 0.05$ (left) and $r = 0.5$ (right). (The* (**AP**) *splitting corresponds to solid lines.)*

Chapter 8

Conclusions

8.1 Summary of this work

The goal of this thesis was the development of computationally efficient algorithms for the convex block-angular minimization problem (**CBA**). From the method of Alternating Directions we have derived three decomposition algorithms and have investigated their theoretical aspects and practical performance. These methods, which can be viewed as block Gauss-Seidel modifications of the Augmented Lagrangian algorithm, take full advantage of the block structure of (**CBA**). We have implemented the algorithms on the Thinking Machines CM-5 parallel supercomputer. We have used the machine in a Multiple Instruction Multiple Data fashion under the *fork-join* coordination protocol: each block of (**CBA**) is assigned to a processor; in the *fork* phase each processor solves a minimization subproblem involving constraints and variables pertaining to its assigned block, and a modified objective that consists of a block of the original separable objective plus a Lagrangian linear term and a quadratic proximal term. The *join* phase is very simple: processors combine their solution information in order to form, as aggregates, new values for the subproblem proximal terms and Lagrangian multipliers. The *join* step can be performed efficiently on the CM-5, because of its tree topology and the availability of fast, `vector_reduce` global communication functions.

In chapter 2 we prove convergence, under mild assumptions, of an extended version of the method of Alternating Directions (ADI). Our method uses variable penalty matrices for the proximal term, which ultimately become fixed; this enables the development of heuristics for penalty update, that can reduce the number of iterations to convergence, by taking further advantage of the problem structure of the application at hand, and by adaptively tuning the penalty using information on the violation of the coupling constraints.

The three highly-parallel ADI algorithms for (**CBA**), the (**ARP**), (**RP**) and (**AP**) splittings, are derived in chapters 4, 5 and 6, respectively. For each one, we simplify the general method of chapter 2, describe the properties of the iterates, present a streamlined algorithm and prove its convergence.

Computational experience on randomly generated linear and quadratic multicommodity network problems on the CM-5 is reported in chapter 7. Our results indicate that, because of the reduced size of the subproblems solved at each iteration, the (**AP**) and (**RP**) algorithms, based on proximizing either resources or activities, perform significantly better than the (**ARP**) algorithm, which proximizes both resources and activities. On large-scale problems, this parallel implementation of the (**AP**) splitting is 2 to 4 times faster than the Dantzig-Wolfe decomposition on the CM-5, and one to two orders of magnitude faster than the serial package MINOS 5.4 on a DEC 5000 workstation.

8.2 Directions for future research

Future work on ADI methods may proceed along two complementary directions: one of generalization and one of specialization.

There are several ways in which problem (9) can be generalized and the method be extended; it is of interest to determine whether the existing convergence theory of chapter 2 can cover these modifications, as well. We can generalize problem (9) by considering an objective function that is the sum of three or more convex terms, and some of the constraints are non-linear. We can extend the ADI method by considering:

- A modified augmented Lagrangian method with a general strictly convex penalty function, not necessarily a quadratic.
- Updating the multipliers and the proximal terms modulo a fixed number of iterations, or using asynchronous updates.
- A sophisticated multiplier update that uses second-order Hessian information.
- Incorporating active-set techniques, for adding/dropping constraints in the course of the iterations.

In general, we are interested in variants that may offer accelerated convergence and can be implemented under the *fork-join* protocol.

In the area of specialization, the focus is on faster algorithms for specific instances of practical interest of (**CBA**). For linear-quadratic (**CBA**), the block subproblem for each iteration is a least-squares linear-quadratic problem with linear constraints, usually sparse. For these problems, an efficient sparse solver may be developed by employing algorithms such as those described in Björck [13, sections 25–27]. (Fast sparse least-squares solvers are needed as building blocks in other mathematical programming algorithms, as well; for instance, for calculating a Newton direction in logarithmic barrier methods for linear programs, see Murray [75, section 7].) For the multicommodity network problem, in particular, specialized solvers that also take into account the network structure can be developed; flow push heuristics can be used to determine a good, feasible initial point.

For the scenario analysis problem, it might be possible to design additional splittings that exploit its staircase structure. Also, the simple form of the non-anticipativity constraints can be exploited in designing an efficient inter-PE communication pattern for the computation of the updates as aggregates. Then, the relative performance of the three splittings on scenario problems has to be investigated, as well as their comparison with other specialized algorithms, such as the Diagonal Quadratic Approximation (Mulvey and Ruszczyński [71]) and Progressive Hedging (Rockafellar and Wets [88, 102], Mulvey and Vladimirou [72, 73, 74]).

Another area of future research is the design of parallel ADI methods for other classes of important optimization problems such as the Linear Complementarity Problem; such methods may be similar to the successive over-relaxation schemes that have been successfully implemented in a parallel environment (De Leone and Mangasarian [21]).

Bibliography

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The design and analysis of computer algorithms.* Addison–Wesley, 1974.

[2] R. Ahuja, T.L Magnanti, and J.B. Orlin. *Network flows: theory, algorithms, and applications.* Prentice–Hall Inc., 1993.

[3] S.G. Akl. *The design and analysis of parallel algorithms.* Prentice–Hall Inc., 1989.

[4] A.I. Ali and J.L. Kennington. MNETGEN program documentation. Technical Report IEOR 77003, Dept. of Industrial Engineering and Operations Research, Southern Methodist University, Dallas, TX, 1977.

[5] G. Amdahl. The validity of the single processor approach to achieving large-scale computing capabilities. In *Proceedings of the AFIPS Spring Joint Computer Conference, Atlantic City, NJ*, volume 30, pages 483–485. AFIPS Press, Reston, VA, April 1967.

[6] A.A. Assad. Multicommodity network flows: a survey. *Networks*, 8:37–91, 1978.

[7] J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[8] R.E. Benner, G.R. Montry, and J.L. Gustafson. A structural analysis algorithm for massively parallel computers. In G.F. Carey, editor, *Parallel Supercomputing: Methods, Algorithms and Applications*, pages 207–222. John Wiley & Sons, 1989.

[9] D.P. Bertsekas. Multiplier methods: a survey. *Automatica*, 12:133–145, 1976.

[10] D.P. Bertsekas. *Constrained optimization and Lagrange multiplier methods.* Academic Press, New York, 1982.

[11] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and distributed computation: numerical methods.* Prentice–Hall Inc., 1989.

[12] M.L. Best, A. Greenberg, C. Stanfill, and L.W. Tucker. CMMD I/O: a parallel unix I/O. In *Proceedings of the 7th International Parallel Processing Symposium, Newport Beach, CA*, pages 489–495. IEEE Computer Society Press, April 1993.

[13] Aake Björck. Least squares methods. Technical report, Dept. of Mathematics, Linköping University, Linköping, Sweden, May 23, 1991.

[14] H. Brézis. *Opérateurs maximaux monotones et semi-groupes de contractions dans les espaces de Hilbert.* North–Holland, 1973.

[15] J. Céa. *Lectures on optimization – theory and algorithms.* Tata Institute, Bombay, 1978.

[16] Y. Censor and A. Lent. An iterative row-action method for interval convex programming. *Journal of Optimization Theory and Applications*, 34:321–353, 1981.

[17] G. Cheng and M. Teboulle. A proximal-based decomposition method for convex minimization problems. *Mathematical Programming, Series A*, 64(1):81–110, 1994.

[18] R.W. Cottle, J.-S. Pang, and R.E. Stone. *The linear complementarity problem.* Academic Press, 1992.

[19] G.B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[20] R. De Leone, M. Gaudioso, and M.-F. Monaco. Nonsmooth optimization methods for parallel decomposition of multicommodity flow problems. Technical Report 1080, Computer Sciences Department, University of Wisconsin, 1992.

[21] R. De Leone and O.L. Mangasarian. Asynchronous parallel successive overrelaxation for the symmetric linear complementarity problem. *Mathematical Programming, Series B*, 42:347–362, 1988.

[22] R. De Leone, R.R. Meyer, S. Kontogiorgis, A. Zakarian, and G. Zakeri. Coordination methods in coarse-grained decomposition. *SIAM Journal on Optimization*, 4(4):777–793, 1994.

[23] J.J. Dongarra and W. Gentzsch, editors. *Computer benchmarks: advances in parallel computing vol. 8.* Elsevier, 1993.

[24] J. Douglas and H.H. Rachford Jr. On the numerical solution of heat conduction problems in two- and three–space variables. *Transactions of the American Mathematical Society*, 82:421–439, 1956.

[25] J. Eckstein. *Splitting methods for monotone operators with applications to parallel optimization.* PhD thesis, Massachusetts Institute of Technology, Department of Civil Engineering, 1989.

[26] J. Eckstein. The alternating step method for monotropic programming on the Connection Machine CM-2. *ORSA Journal on Computing*, 5(1):293–318, 1993.

[27] J. Eckstein. Large-scale parallel computing, optimization, and operations research: a survey. *ORSA Computer Science Technical Section Newsletter*, 14(2):1–27, 1993.

[28] J. Eckstein and D.P. Bertsekas. On the Douglas–Rachford splitting method and the proximal point method for maximal monotone operators. *Mathematical Programming, Series A*, 55(3):293–318, 1992.

[29] J. Eckstein and M. Ferris. Operator splitting methods for monotone linear complementarity problems. Technical Report TMC-239, Thinking Machines Corporation, 1992.

[30] M.C. Ferris. Parallel constraint distribution for convex quadratic programs. Technical Report 1009, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1991. To appear in Mathematics of Operations Research.

[31] M.C. Ferris and J.D. Horn. Partitioning mathematical problems for parallel solution. Technical Report 1232, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1994.

[32] M.C. Ferris and O.L. Mangasarian. Parallel constraint distribution. *SIAM Journal on Optimization*, 1(4):487–500, 1991.

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

[34] M. Fortin and R. Glowinski, editors. *Augmented Lagrangian methods: applications to the numerical solution of boundary-valued problems.* North–Holland, 1983.

[35] M. Frank and F. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[36] M. Fukushima. Application of the alternating direction method of multipliers to separable convex programming problems. *Computational Optimization and Applications*, 1:93–111, 1992.

[37] D. Gabay. Applications of the method of multipliers to variational inequalities. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian methods: applications to the numerical solution of boundary-valued problems.* North–Holland, 1983.

[38] M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-Completeness.* W.H. Freeman, 1979.

[39] A.M. Geoffrion. Primal resource-directive approaches for optimizing nonlinear decomposable systems. *Operations Research*, pages 375–403, May-June 1970.

[40] P.E. Gill, S.J. Hammarling, W. Murray, M.A. Saunders, and M.H Wright. User's guide for LSSOL: a FORTRAN package for constrained linear least-squares and convex quadratic programming. Technical Report SOL 86-1R, Stanford University, 1987.

[41] R. Glowinski and P. Le Tallec. *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics.* Society for Industrial and Applied Mathematics, 1989.

[42] R. Glowinski and A. Marrocco. Sur l' approximation par éléments finis d' ordre un, et la résolution par pénalisation-dualité d' une classe de problémes de Dirichlet nonlinéaires. *Revue Française d' Automatique, Informatique et Recherche Opérationelle*, 2:41–76, 1975.

[43] S.K. Gnanendran and J.K. Ho. Load balancing in the parallel optimization of block-angular linear programs. *Mathematical Programming*, 62:41–67, 1993.

[44] Ye. G. Golshtein. The block method of convex programming. *Soviet Mathematics Doklady*, 33:584–587, 1986.

[45] Ye. G. Golshtein. A general approach to decomposition of optimization systems. *Soviet Journal of Computer and Systems Sciences*, 25(3):105–114, 1987.

[46] J.L. Gustafson. Reevaluating Amdahl's law. *Communications of the ACM*, 31(5):532–533, 1988.

[47] S.P Han and G. Lou. A parallel algorithms for a class of convex problems. *SIAM Journal on Control and Optimization*, 26:345–355, 1988.

[48] J.L. Hennessy and D.A. Patterson. *Computer architecture: a quantitative approach*. Morgan Kaufmann, 1990.

[49] W.D. Hillis and L.W. Tucker. The CM-5 Connection Machine: a scalable supercomputer. *Communications of the ACM*, 36(11):31–40, 1993.

[50] J.K. Ho, T.C. Lee, and R.P. Sundarraj. Decomposition of linear programs using parallel computation. *Mathematical Programming, Series B*, 42:391–406, 1988.

[51] K. Hwang. *Advanced computer architecture with parallel programming*. McGraw-Hill, 1993.

[52] S.L. Johnsson, Y. Saad, and M. Schultz. Alternating direction methods on multiprocessors. *SIAM journal of Scientific and Statistical Computing*, 8(5):686–700, 1987.

[53] P. Kall. *Stochastic linear programming*. Springer Verlag, 1976.

[54] P.V. Kamesam and R.R. Meyer. Multipoint methods for separable nonlinear networks. *Mathematical Programming Study*, 22:185–205, 1984.

[55] J.L. Kennington. A survey of linear cost multicommodity network flows. *Operations Research*, 26:209–236, 1978.

[56] J.L. Kennington and R.V. Helgason. *Algorithms for network programming*. John Wiley & Sons, New York, 1980.

[57] D. Klingman, A Napier, and J. Stutz. NETGEN: a program for generating large scale capacitated assignment, transportation and minimum cost network problems. *Management Science*, 20(8):814–821, 1974.

[58] J.S. Kowalik and S.P. Kumar. Parallel algorithms for recurrence and tridiagonal equations. In J.S. Kowalik, editor, *Parallel MIMD computation: the HEP supercomputer and its applications*, pages 295–307. MIT Press, Cambridge, MA, 1985.

[59] L.S Lasdon. *Optimization theory for large systems*. MacMillan, 1970.

[60] H.W. Lawson. *Parallel processing in industrial real-time applications*. Prentice–Hall Inc., 1992.

[61] C.E. Leiserson et al. The network architecture of the Connection Machine CM-5. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*. ACM, 1992.

[62] P.L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

[63] O.L. Mangasarian. *Nonlinear programming*. McGraw-Hill, 1969.

[64] O.L. Mangasarian. Solution of symmetric linear complementarity problems by iterative methods. *Journal of Optimization Theory and Applications*, 22(4):465–485, 1977.

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

[66] G.I. Marchuk. Splitting and alternating direction methods. In P.G. Ciarlet and J.L. Lions, editors, *Handbook of numerical analysis*. North–Holland, Berlin, 1990.

[67] E.W. Mayr. Basic parallel algorithms in graph theory. In G. Tinhofer, E.W. Mayr, H. Noltenmeier, and M.M Syslo, editors, *Computational graph theory*, pages 69–91. Springer Verlag, Vienna, 1990.

[68] D. Medhi. Parallel bundle-based decomposition for large-scale structured mathematical programming problems. *Annals of Operation Research*, 22:101–127, 1990.

[69] A. Miele, P.E. Moseley, A.V. Levy, and G.M. Coggins. On the method of multipliers for mathematical programming problems. *Journal of Optimization Theory and Applications*, 10(1):1–33, 1972.

[70] J.M. Mulvey and A. Ruszczyński. A new scenario decomposition method for large-scale stochastic optimization. Technical Report SOR 91-19, Princeton University, 1991.

[71] J.M. Mulvey and A. Ruszczyński. A diagonal quadratic approximation method for large scale linear programs. *Operations Research Letters*, 12:205–215, 1992.

[72] J.M. Mulvey and H. Vladimirou. Evaluation of a parallel hedging algorithm for stochastic network programming. In R. Sharda et al., editors, *Impacts of recent computer advances on operations research*. North–Holland, 1989.

[73] J.M. Mulvey and H. Vladimirou. Solving multistage stochastic networks: an application of scenario aggregation. *Networks*, 21(6):619–643, 1990.

[74] J.M. Mulvey and H. Vladimirou. Stochastic network programming for financial planning problems. *Management Science*, 38(11):1642–1664, 1992.

[75] W. Murray. Methods for large-scale linear programming. In *Algorithms and model formulations in mathematical programming*, NATO ASI Series F vol. 51, pages 115–137. Springer Verlag, 1989.

[76] B.A. Murtagh and M.A. Saunders. MINOS 5.1 user's guide. Technical Report SOL 83.20R, Stanford University, 1987.

[77] B.A. Murtagh and M.A. Saunders. MINOS 5.4 release notes, appendix to MINOS 5.1 user's guide. Technical report, Stanford University, 1992.

[78] R.J. O' Doherty and B.L. Pierson. A numerical study of augmented penalty function algorithms for terminally constrained optimal control problems. *Journal of Optimization Theory and Applications*, 14(4):393–403, 1974.

[79] P.C. Patton. Performance limits for parallel processors. In G.F. Carey, editor, *Parallel supercomputing: methods, algorithms and applications*, pages 1–16. John Wiley & Sons, 1989.

[80] D.W. Peaceman and H.H. Rachford Jr. The numerical solution of parabolic and elliptic differential equations. *SIAM Journal*, 3:28–42, 1955.

[81] M.Ç. Pinar and S.A. Zenios. Parallel decomposition of multicommodity network flows using a linear-quadratic penalty algorithm. *ORSA Journal on Computing*, 4(3):235–248, 1992.

[82] B.T. Polyak. *Introduction to optimization.* Optimization Software, Inc., Publications Division, 1987.

[83] S.M. Robinson. *Lectures on convex analysis.* Madison, Wisconsin, 1978.

[84] S.M. Robinson. *Lecture notes on maximal monotone operators, IE 823.* Madison, Wisconsin, 1990.

[85] R.T. Rockafellar. *Convex analysis.* Princeton University Press, 1970.

[86] R.T. Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1:97–116, 1976.

[87] R.T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14:887–898, 1976.

[88] R.T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16:119–147, 1991.

[89] A. Ruszczyński. Augmented lagrangian decomposition for sparse convex optimization. Working Paper WP 92-75, IIASA, April 1993.

[90] R.A. Saleh et al. Parallel circuit simulation on supercomputers. In *Proceedings of the IEEE, special issue on supercomputer technology*, pages 1983–1991. IEEE Press, December 1989.

[91] G.L. Schultz. *Barrier decomposition for the parallel optimization of block-angular programs.* PhD thesis, University of Wisconsin-Madison, 1991.

[92] G.L. Schultz and R.R. Meyer. An interior point method for block angular optimization. *SIAM Journal on Optimization*, 1(4):583–602, 1991.

[93] J.E. Spingarn. Partial inverse of a monotone operator. *Applied Mathematics and Optimization*, 10:247–265, 1983.

[94] J.E. Spingarn. Applications of the method of partial inverses to convex programming: decomposition. *Mathematical Programming*, 32:199–223, 1985.

[95] G. Stephanopoulos and A.W. Westerberg. The use of Hestenes' method of multipliers to resolve dual gaps in engineering system optimization. *Journal of Optimization Theory and Applications*, 15:285–309, 1975.

[96] Thinking Machines Corporation, Cambridge, MA. *The Connection Machine CM–5 technical summary*, 1991.

[97] Thinking Machines Corporation, Cambridge, MA. *CMMD reference manual, version 3.0*, 1993.

[98] C.D. Thomborson. Does your workstation computation belong on a vector supercomputer ? *Communications of the ACM*, 36(11):41–49, 1993.

[99] P.L. Toint and D. Tuyttens. LSSNO, a FORTRAN subroutine for solving large scale nonlinear network optimization problems. Technical Report 90/11, Dept. of Mathematics, Facultés Universitaires de la Paix, Namur, Belgium, 1990.

[100] P.L. Toint and D. Tuyttens. On large scale nonlinear network optimization. *Mathematical Programming*, 48:125–159, 1990.

[101] H. Uzawa. Iterative methods for concave programming. In K.J. Arrow, L. Hurwicz, and H. Uzawa, editors, *Studies in linear and nonlinear programming*, pages 154–165. Stanford University Press, Stanford, CA, 1958.

[102] R. J.-B. Wets. The aggregation principle in scenario analysis and stochastic optimization. In *Algorithms and model formulations in mathematical programming*, NATO ASI Series F vol. 51, pages 91–114. Springer Verlag, 1989.

[103] J. Worlton. Towards a science of parallel computation. In *Computational mechanics – advances and trends, AMD vol. 75*, pages 23–35. ASME, New York, 1987.

[104] M-Y. Wu and D.D. Gajski. Computer-aided programming for message-passing systems: problems and a solution. In *Proceedings of the IEEE, special issue on supercomputer technology*, pages 1983–1991. IEEE Press, December 1989.

[105] A. Zakarian. Parallel solution of multicommodity network flow programs via nonlinear Jacobi algorithms. Unpublished manuscript, University of Wisconsin–Madison, Computer Sciences Dept., 1994.

[106] S.A. Zenios. On the fine-grain decomposition of multicommodity transportation problem. *SIAM Journal on Optimization*, 1(4):643–669, 1991.

[107] S.A. Zenios, M.Ç. Pinar, and R.S. Dembo. A smooth penalty function algorithm for network-structured problems. Technical Report 90-12-05, Decision Sciences Department, The Wharton School, University of Pennsylvania, January 1991.