WARSAW UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRONICS AND INFORMATION TECHNOLOGY
Institute of Control and Computation Engineering

Ph.D. Thesis

Anthony Chukwuemeka Nwachukwu, M.Sc.

Augmented Lagrangian-Based Algorithms for Separable Non-convex Optimization with Applications in Network Routing and Machine Learning

Supervisor
dr hab. inż. Andrzej Karbowski

WARSAW 2025

POLITECHNIKA WARSZAWSKA

WYDZIAŁ ELEKTRONIKI I TECHNIK INFORMATYCZNYCH
Instytut Automatyki i Informatyki Stosowanej

Rozprawa doktorska

mgr inż. Anthony Chukwuemeka Nwachukwu

Algorytmy oparte na rozszerzonym lagranżianie do rozwiązywania separowalnych zadań optymalizacji niewypukłej z zastosowaniami w routingu sieciowym oraz uczeniu maszynowym

Promotor
dr hab. inż. Andrzej Karbowski

WARSZAWA 2025

Acknowledgements

I owe my deepest gratitude to my supervisor, dr. hab. inż. Andrzej Karbowski. His guidance, insight, and expertise in applied mathematics and optimization have been invaluable throughout this thesis. I am honored to have worked under his mentorship, which has profoundly influenced my development as a researcher.

I am deeply grateful to my family for their unwavering love and support. I owe particular thanks to my father, Pastor Augustine Nwachukwu, and my mother, Mrs. Grace Nwachukwu, whose faith in me and sacrifices have made this journey possible. My siblings have also been a source of inspiration and encouragement; in particular, I thank my elder sister, Isadora Nduka, for her wise counsel and constant support.

I would like to thank my friends for their companionship and support during my doctoral studies. I am especially grateful to my flatmate and friend, Ayomide Fasemire, for providing a supportive environment and camaraderie throughout my study, and to my friends, Ayodeji Ajadi and Adedeji Atiba, for their steadfast friendship and encouragement.

Finally, I extend my sincere gratitude to all those whose support, though not mentioned by name, was invaluable to the completion of this thesis.

Abstract

This dissertation develops and analyzes distributed augmented Lagrangian algorithms for large-scale separable convex and non-convex optimization problems, with generalized network routing and machine learning applications. Such problems involve a global objective subject to coupling constraints over multiple subsystems, and they arise, for example, in energy-aware communication networks, regularized systems of linear equations, and distributed clustering. A comprehensive empirical study of existing decomposition-based Lagrangian methods (classical dual decomposition, Bertsekas, ADMM, Tatjewski method, and SALA) on representative non-convex problems was first conducted. Our experiments indicate that Bertsekas proximal augmented Lagrangian method yields superior solution quality, scalability, and convergence speed performance. Building on this finding, the core of the thesis is the development of a novel asynchronous variant of the Bertsekas decomposition algorithm.

This asynchronous algorithm allows computational nodes to update local variables without global synchronization, using possibly stale information. The update rules were formalized under a bounded-delay computational model and derive a rigorous convergence analysis. Under standard regularity conditions (smoothness, Lipschitz continuity of gradients, and constraint qualifications) and bounded delays, the iterates are shown to converge to a Karush–Kuhn–Tucker point of the original non-convex problem. Also, the method was extended to handle general constraints within the augmented Lagrangian framework, including inequality and mixed-integer constraints.

The algorithms were validated through extensive numerical experiments on synthetic benchmarks and real-world datasets. Application examples include simultaneous routing and bandwidth allocation in energy-aware networks, solution of large, regularized systems of linear equations, and distributed K-means clustering. In each case, the asynchronous Bertsekas method achieves objective values comparable to the synchronous implementation while significantly reducing wall-clock computation time as the degree of parallelism increases.

This dissertation identifies the most effective augmented Lagrangian decomposition scheme for separable non-convex optimization and extends it to an asynchronous parallel framework with provable convergence guarantees. These contributions advance the theory and practice of large-scale distributed optimization, demonstrating how asynchronous Lagrangian methods can exploit parallel computing to solve complex network and machine learning problems more efficiently.

Keywords: Asynchronous Convergence, Bertsekas Augmented Lagrangian Method, Non-convex Optimization, Distributed Computing, Computational Efficiency, Scalability, Algorithmic

Enhancements, Machine Learning, High-dimensional Optimization, Augmented Lagrangian, Optimization, Machine Learning, Alternating Direction Method of Multipliers, Parallel Computing, Convex and Non-convex Optimization, ADMM, Distributed Computing, Clustering, Support Vector Machine, Lasso, Regression, Non-convex optimization, Multiplier Method, Separable problems, Network Optimization, MINLP, NP-hard problems, Lagrangian relaxation

Streszczenie

W niniejszej rozprawie doktorskiej opracowano i przeanalizowano algorytmy rozszerzonego lagranżianu (inaczej algorytmy mnożników Lagrange'a) dla dużych, separowalnych i niewypukłych zadań optymalizacji, z zastosowaniem do uogólnionego routingu sieciowego oraz uczenia maszynowego. Takie zadania obejmują globalny cel podlegający ograniczeniom wiążącym wiele podzadań (podsystemów) i pojawiają się na przykład w energooszczędnych sieciach telekomunikacyjnych, odzyskiwaniu rzadkich sygnałów i obliczeniach rozproszonych związanych z klastrami.

Najpierw przeprowadzono kompleksowe badania empiryczne istniejących metod mnożników Lagrange'a wykorzystujących dekompozycję (klasyczna dekompozycja dualna, ADMM, metody Tatjewskiego, Bertsekasa i SALA) na wybranych zadaniach niewypukłych. Przeprowadzone eksperymenty wskazują, że metoda Bertsekasa rozszerzenia lagranżianu o składnik proksymalny daje najlepszą jakość rozwiązania, skalowalność i szybkość zbieżności. Opierając się na tym odkryciu, sednem rozprawy stało się opracowanie nowego asynchronicznego wariantu algorytmu dekompozycji Bertsekasa.

Ten asynchroniczny algorytm pozwala węzłom obliczeniowym aktualizować zmienne lokalne bez globalnej synchronizacji, wykorzystując potencjalnie nieaktualne informacje. Reguły aktualizacji zostały sformalizowane w ramach modelu obliczeniowego z ograniczonym opóźnieniem, dla którego wyprowadzono rygorystyczną analizę zbieżności. Pod standardowymi warunkami regularności (gładkość, ciągłość gradientów Lipschitza i kwalifikacje ograniczeń) i przy ograniczonych opóźnieniach, iteracje są zbieżne do punktu Karusha-Kuhna-Tuckera oryginalnego zadania niewypukłego. Metoda ta została również rozszerzona w celu obsługi w ramach rozszerzonego lagranżianu ogólnych ograniczeń, w tym nierówności i mieszanych ograniczeń ze zmiennymi całkowitoliczbowymi.

Algorytmy zostały zweryfikowane poprzez obszerne eksperymenty numeryczne na syntetycznych benchmarkach i rzeczywistych zbiorach danych. Przykłady zastosowań obejmują jednoczesny routing i alokację przepustowości w sieciach energooszczędnych, rozwiązywanie dużych, rzadkich układów równań liniowych i rozproszone grupowanie metodą K-środków. W każdym przypadku asynchroniczna metoda Bertsekasa osiąga wartości docelowe porównywalne z implementacją synchroniczną, jednocześnie znacznie skracając czas obliczeń wraz ze wzrostem stopnia zrównoleglenia.

Niniejsza rozprawa doktorska identyfikuje najbardziej efektywny schemat dekompozycji rozszerzonego lagranżianu dla separowalnej optymalizacji niewypukłej i rozszerza go do asynchronicznej struktury równoległej z możliwymi do udowodnienia gwarancjami zbieżności. Praca ta rozwija teorię i praktykę rozproszonej optymalizacji na dużą skalę, pokazując jak asynchroniczne metody mnożników Lagrange'a mogą wykorzystać obliczenia równoległe do

bardziej efektywnego rozwiązywania złożonych zadań sieciowych i uczenia maszynowego.

Słowa kluczowe: Zbieżność asynchroniczna, metoda rozszerzonego lagranżianu Bertsekasa, optymalizacja niewypukła, obliczenia rozproszone, wydajność obliczeniowa, skalowalność, ulepszenia algorytmiczne, uczenie maszynowe, optymalizacja wielowymiarowa, rozszerzony lagranżian, optymalizacja, uczenie maszynowe, metoda mnożników o naprzemiennym kierunku, obliczenia równoległe, optymalizacja wypukła i niewypukła, ADMM, obliczenia rozproszone, klasteryzacja, maszyna wektorów nośnych, lasso, regresja, optymalizacja niewypukła, metoda mnożników, zadania separowalne, optymalizacja sieci, MINLP, zadania NP-trudne, relaksacja Lagrange'a

Contents

List of Figures

List of Tables

1. Introduction

Modern machine learning and networked systems generate massive, high-dimensional data and demand increasingly complex optimization solutions. In such settings, problems are often *large-scale* and *non-convex*, yet exhibit separable structure across distributed components. For example, tasks like simultaneous routing and bandwidth allocation in energy-aware networks or clustering of large datasets naturally decompose across subsystems. **Distributed optimization** has therefore become essential: algorithms must coordinate many nodes to solve a common objective. As Zhang and Kwok observe, "distributed optimization algorithms are highly attractive for solving big data problems" [1]. This appeal is especially strong in applied statistics and machine learning, where global objectives can be formed by coordinating local subproblems.

Among the leading techniques for such problems are **Augmented Lagrangian-based methods**, including the classical method of multipliers and the Alternating Direction Method of Multipliers (ADMM). These methods incorporate constraints via dual variables and quadratic penalties, yielding subproblems that can be solved in parallel. For instance, Boyd *et al.* note that ADMM "is a simple but powerful algorithm that is well suited to distributed convex optimization, and in particular to problems arising in applied statistics and machine learning" [2]. In ADMM, small local subproblems at each node are coordinated to solve a large global problem [2]. In fact, ADMM blends the benefits of classical **dual decomposition** and **augmented Lagrangian** approaches to achieve robust, distributed convergence [2]. In practice, Lagrangian decomposition schemes (e.g. those of Bertsekas, Tatjewski, and others) and ADMM have proven effective for many large-scale problems.

Despite this progress, most implementations of these methods assume *synchronized* updates: all nodes wait for the slowest processor at each iteration before proceeding. This synchronization can incur serious delays in heterogeneous or unreliable networks. A more flexible strategy is to allow **asynchronous** updates, where fast processors proceed with local computations without waiting. As early work by Bertsekas and Tsitsiklis showed, asynchronous iterative algorithms can "alleviate communication overloads" and "are not excessively slowed down by... communication delays" [3]. In the context of ADMM, Zhang and Kwok demonstrate that the synchronous version "suffers from the **straggler problem** as its updates have to be synchronized" [1]. In contrast, their experiments show that a carefully designed asynchronous ADMM can "reduce the time on network waiting, and achieves faster convergence than its synchronous counterpart in terms of wall clock time" [1]. These findings highlight the practical advantages of asynchrony: even if the mathematical guarantees of synchronous and asynchronous methods are similar under mild conditions [3], the ability to proceed without global barriers can yield substantial speedups in wall-clock time.

Research Problem

Motivated by these observations, this dissertation investigates the following question: *Which Lagrangian-based optimization algorithms are most effective for solving large-scale, distributed non-convex problems, and how can we design and rigorously analyze asynchronous versions of these methods?*

The focus is on a general class of constrained nonlinear programs with separable structure. For concreteness, consider the problem:

$$\min_{x \in X} \quad f(x) = \sum_{i=1}^{N} f_i(x_i) \tag{1}$$

$$\text{s.t.} \quad h(x) = \sum_{i=1}^{N} h_i(x_i) = 0 \tag{2}$$

$$\text{s.t.} \quad g_j(x) = \sum_{i=1}^{N} g_{ji}(x_i) \leq 0, \qquad j = 1,2,\ldots,m \tag{3}$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and $h : \mathbb{R}^n \to \mathbb{R}^p$, $g : \mathbb{R}^n \to \mathbb{R}^q$ represent coupling equality and inequality constraints. Such problems arise in many domains (network flows, signal recovery, distributed clustering, etc.) and are challenging due to non-convexity and scale.

Augmented Lagrangian methods handle these difficulties by embedding the constraints into the objective with penalty terms and dual multipliers. When the problem is separable, decomposition schemes (e.g. Bertsekas dual decomposition, Tatjewski's approach, or modern ADMM variants) allow each subsystem to solve a local subproblem in parallel, followed by a coordination step. While classical analyses often assume a synchronized algorithm, our focus is on the *asynchronous* setting: nodes update using possibly stale information and without global coordination. The goal is to develop an asynchronous Lagrangian method with provable convergence guarantees.

To answer the research question, a comparative study of prominent decomposition methods on representative network optimization and machine learning problems was conducted. Our experiments indicate that Bertsekas proximal augmented Lagrangian method (a form of Alternating Direction Method) yields superior performance in terms of solution quality, scalability, and convergence speed. Motivated by this, the core of our thesis is the design and analysis of an **Asynchronous Bertsekas Decomposition Algorithm**. The formalization of its computational model allowed for delayed or out-of-order updates across subsystems. Then, a rigorous proof of convergence under general non-convexity assumptions was provided, extending classical results to this asynchronous context.

Contributions

This dissertation makes the following key contributions:

- **Comprehensive Review:** A comprehensive review of decomposition-based Lagrangian methods (classical dual decomposition, ADMM, Bertsekas, Tatjewski, SALA) applied to challenging non-convex problems.
- **New Asynchronous Algorithm:** The development of a novel asynchronous optimization algorithm – an asynchronous variant of the Bertsekas decomposition scheme – suitable for distributed environments. The update rules are formally specified in the presence of communication delays.
- **Convergence Analysis:** A rigorous convergence theory for the asynchronous algorithm. Under standard regularity conditions (e.g., Lipschitz continuity, sufficiency of constraint qualifications), the proposed method is shown to converge to stationary solutions despite asynchrony.
- **Algorithmic Extensions:** Algorithmic innovations to handle general constraints. In particular, the classical equality-constrained decompositions were adapted to effectively cope with inequality and integer constraints within the asynchronous framework.
- **Extensive Validation:** Extensive numerical validations on both synthetic benchmarks and real-world datasets. Our results demonstrate that the asynchronous Bertsekas method is both practical and robust, often outperforming synchronized counterparts in wall-clock time while maintaining solution accuracy.
- **Insights on Sync vs Async:** New insights into the trade-offs between synchronous and asynchronous execution. The convergence behavior and efficiency of the methods are analyzed, highlighting when and why asynchrony leads to gains in distributed optimization settings.

Thesis Organization

The rest of the dissertation is structured as follows. Chapter 2 provides the mathematical foundation, covering convex analysis, Lagrangian duality and optimality conditions, stability, and convergence theory that underpin our algorithms. Chapter 3 reviews classical decomposition methods, including the method of multipliers, ADMM, Bertsekas scheme, Tatjewski's method, and SALA, highlighting their theory and implementation characteristics. Chapter 4 introduces the asynchronous Bertsekas algorithm, formalizes its parallel updating model, and proves its convergence in the non-convex setting.

Chapter 5 applies several Lagrangian-based decomposition algorithms to the simultaneous routing and bandwidth allocation problem in energy-aware networks. Chapter 6 builds on these results to present and compare synchronous and asynchronous implementations of the

Bertsekas method in that context, demonstrating the computational advantages of asynchrony in practical settings. Chapter 7 formulates a regularized system of linear Equations problem and adapts the same class of decomposition methods to solve it. Chapter 8 mirrors the structure of Chapter 6, evaluating the asynchronous and synchronous Bertsekas algorithms for regularized systems of linear equations and highlighting the generality of the approach. Chapter 9 presents a distributed formulation of the constrained K-Means problem and applies the decomposition algorithms. Chapter 10 further investigates the impact of asynchrony on K-Means clustering through empirical comparisons on large-scale datasets.

Finally, Chapter 11 concludes the dissertation with a summary of findings, reflections on limitations, and proposed directions for future research.

2. Theoretical Foundations

This chapter reviews the mathematical concepts and tools needed for our convergence analysis. Section 2.1 introduces the basic definitions and notation. Section 2.2 covers key **convex analysis** results and first-order optimality conditions, including subgradients and convex conjugates. Section 2.3 develops the foundations of **Lagrangian duality** and optimality theory. The (augmented) Lagrangian, the dual function, and the Karush-Kuhn-Tucker (KKT) conditions for constrained problems are defined. Particular attention is given to the properties of the dual problem, including concavity and conditions for *strong duality*, such as Slater's condition, which guarantees a zero duality gap in convex problems [4]. Next, Section 2.4 recalls key facts about **continuity and stability**, including Lipschitz continuity of gradients and perturbation results that ensure bounded solutions. Section 2.5 develops elements of **convergence theory** and fixed-point analysis, introducing contraction mappings and their role in iterative methods. Finally, Section 2.6 covers **monotone operator theory** and variational properties: the monotonicity for set-valued mappings is defined, and there is a discussion on how primal-dual updates can be viewed as finding zeros of monotone operators. In particular, an operator F is *monotone* if $(F(x)-F(y))^{\mathsf{T}}(x-y) \geq 0$ for all x, y (cf. the definition below) [5]. Strong monotonicity and Lipschitz continuity [5] will play a key role: for a strongly monotone operator with constant $m > 0$ and Lipschitz constant L, one has

$$m\|x-y\|^2 \leq (F(x)-F(y))^{\mathsf{T}}(x-y) \leq L\|x-y\|^2,$$

ensuring convergence properties for certain update schemes [5].

Throughout this chapter, the fundamental results are stated and standard references are cited when appropriate, building a rigorous foundation for the asynchronous convergence proofs to come.

2.1. Preliminaries and Notation

Vectors in $\mathbb{R}^n$ are represented by lowercase letters x, y, and matrices by uppercase letters A, B. For a vector x, $\|x\|$ is the Euclidean norm. For a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, $\nabla f(x)$ is its gradient; more generally, $\partial f(x)$ denotes the subdifferential if f is convex but not smooth. The standard set operations are used: for a matrix A, A^{T} is its transpose and $\|A\|$ its spectral norm; I is the identity matrix. The feasible set for constraints is denoted by X. A sequence $\{x^k\}$ converges to x^* if $\|x^k - x^*\| \to 0$. The "Lipschitz" is used to mean that a function or gradient has a bounded rate of change, and "contractive" to mean a mapping shrinks distances by a constant positive factor less than 1.

2.2. Convex Analysis and Optimality Conditions

A set $C \subseteq \mathbb{R}^n$ is **convex** if any line segment between two points in C lies in C. A function $f : C \to \mathbb{R}$ is convex if

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y), \quad \forall x, y \in C,\ \theta \in [0,1].$$

Important consequences of convexity include first-order conditions: if f is differentiable and convex, then its gradient satisfies,

$$f(y) \geq f(x) + \nabla f(x)^{\mathsf{T}}(y-x), \quad \forall x, y \in C.$$

This implies that any stationary point of a convex f (where $\nabla f(x) = 0$) is a global minimum. More generally, for a convex (possibly nondifferentiable) f, x^* is optimal for $\min_{x \in C} f(x)$ if and only if $0 \in \partial f(x^*)$ (the subdifferential) when C is unconstrained. The KKT conditions described later will be required for constrained problems.

Convex conjugates and dual norms also appear: the convex conjugate f^* of f is defined by

$$f^*(u) = \sup_x (u^{\mathsf{T}} x - f(x)).$$

While the duality theory (next section) treats these formally, here convex problems enjoy **strong duality** under mild regularity for linear constraints. In particular, if f is convex and differentiable and the constraints are affine ($Cx = d$), then the duality gap is zero [4]. Slater's condition generalizes this to inequality constraints: if there exists a strictly feasible point for a convex problem, then strong duality holds (i.e., optimal primal and dual values coincide) [4]. These facts ensure that solving the dual problem is often as good as solving the primal, a property that will be exploited in decomposition methods.

While the duality theory (next section) treats these formally, here convex problems enjoy **strong duality** under mild regularity for linear constraints. In particular, if f is convex and differentiable and the constraints are affine ($Cx = d$), then the duality gap is zero [4]. Slater's condition generalizes this to inequality constraints: if there exists a strictly feasible point for a convex problem, then strong duality holds (i.e., optimal primal and dual values coincide) [4]. A **strictly feasible point** is a point that strictly satisfies all inequality constraints; that is, for a problem of the form $g_j(x) \leq 0$, such a point satisfies $g_j(x) < 0$ for all j, while still satisfying any equality constraints exactly. These facts ensure that solving the dual problem is often as good as solving the primal, a property that will be exploited in decomposition methods.

2.3. Lagrangian Duality and Optimality

Consider a general constrained problem,

$$\begin{aligned} &\min_{x \in X} f(x) \\ &\text{s.t. } h(x) = 0, \quad g(x) \leq 0, \end{aligned}$$

where $h : \mathbb{R}^n \to \mathbb{R}^p$ and $g : \mathbb{R}^n \to \mathbb{R}^q$ are constraint functions and X is a simple convex set (e.g., box constraints). The (ordinary) **Lagrangian** is defined by

$$L(x, \lambda, \mu) = f(x) + \lambda^\mathsf{T} h(x) + \mu^\mathsf{T} g(x),$$

where $\lambda \in \mathbb{R}^p$ and $\mu \in \mathbb{R}^q$ are the dual multipliers for the equality and inequality constraints, respectively, with $\mu \geq 0$. The **dual function** is

$$d(\lambda, \mu) = \inf_{x \in X} L(x, \lambda, \mu).$$

Maximizing $d(\lambda, \mu)$ over $\lambda, \mu \geq 0$ defines the *dual problem*. In general, $d(\lambda, \mu)$ is concave in (λ, μ), even if the primal is non-convex, because it is a pointwise infimum of affine functions. *Weak duality* always holds: the optimal dual value is a lower bound on the optimal primal value. *Strong duality* (equality of optimal values) requires convexity or constraint qualifications; as noted, for convex problems under Slater's condition, strong duality holds [4].

The **Karush-Kuhn-Tucker (KKT) conditions** characterize optimality for convex problems: at a primal-dual optimal pair (x^*, λ^*, μ^*), one has:

- *Stationarity:* $\nabla f(x^*) + J_h(x^*)^\mathsf{T} \lambda^* + J_g(x^*)^\mathsf{T} \mu^* = 0$,
- *Primal feasibility:* $h(x^*) = 0, \quad g(x^*) \leq 0$,
- *Dual feasibility:* $\mu^* \geq 0$,
- *Complementary slackness:* $\mu_i^* g_i(x^*) = 0$ for all i.

Here, $h : \mathbb{R}^n \to \mathbb{R}^p$ and $g : \mathbb{R}^n \to \mathbb{R}^q$ are the equality and inequality constraint functions, respectively. The matrices $J_h(x^*) \in \mathbb{R}^{p \times n}$ and $J_g(x^*) \in \mathbb{R}^{q \times n}$ denote their Jacobians, with each row corresponding to the gradient of a constraint function. These conditions will serve as a benchmark for convergence: it will be shown that the asynchronous algorithm drives iterates toward satisfying analogous conditions (typically in a residual sense).

An important variant is the **augmented Lagrangian**, which adds a penalty term to the ordinary Lagrangian to improve numerical stability. For example, for equality constraints

$h(x) = 0$, one defines

$$L_\rho(x, \lambda) = f(x) + \lambda^\top h(x) + \frac{\rho}{2}\|h(x)\|^2,$$

with a penalty parameter $\rho > 0$. This penalized objective leads to algorithms that combine dual ascent with penalty updates. The augmented Lagrangian remains convex in x (for convex f) and underlies methods like ADMM and the method of multipliers. In later chapters, the augmented Lagrangians will be used to derive decomposition updates for distributed subproblems.

2.4. Continuity, Lipschitz Conditions, and Stability

There is an assumption that all functions of interest (objective and constraints) have Lipschitz-continuous gradients. Concretely, f is L_f-*smooth* if

$$\|\nabla f(x) - \nabla f(y)\| \leq L_f \|x - y\|, \quad \forall x, y.$$

Such smoothness implies that f is continuously differentiable and that its Hessian (if it exists) is bounded: $\nabla^2 f(x) \preceq L_f I$. Lipschitz conditions ensure that small changes in the input lead to controlled changes in outputs, which is crucial for stability under asynchronous updates.

It is also required that constraint functions h and g are continuously differentiable and that the feasible set is nonempty. Under these assumptions and mild constraint qualifications (e.g., linear independence of the active constraint gradients), solution sets vary continuously with problem data. In particular, small perturbations in the problem or dual variables lead to small changes in primal solutions, a fact that will be used to bound the error introduced by stale information in asynchronous updates.

2.5. Fixed-Point Iterations and Convergence

Many optimization algorithms can be viewed as iteratively seeking a fixed point of some operator. For a mapping $T : \mathbb{R}^n \to \mathbb{R}^n$, a point x^* satisfying $x^* = T(x^*)$ is a fixed point. Banach's Fixed-Point Theorem states that if T is a contraction (i.e., $\|T(x) - T(y)\| \leq c\|x - y\|$ with $c < 1$), then the iteration $x^{k+1} = T(x^k)$ converges linearly to the unique fixed point. In optimization, gradient descent, proximal-point, and alternating-update schemes often have this structure under certain strong convexity or smoothness conditions.

In the asynchronous setting, fixed-point analysis still applies but must account for delayed or partial information. A common technique is to model the effect of delays as additional contraction or noise and show that the iteration remains convergent if delays are bounded. Such ideas will be leveraged in Chapter 4, using results from the theory of time-varying

and nonexpansive iterations. In particular, when operators are nonexpansive (Lipschitz with constant $L \leq 1$), *averaging* or *relaxation* is often needed to ensure convergence; this is a standard tool in convex splitting methods [2].

2.6. Monotone Operators and Variational Properties

Finally, monotonicity is discussed, an important concept for analyzing primal-dual methods. A (possibly multivalued) operator $F : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is **monotone** if for all $(x,u),(y,v) \in \text{graph}(F)$,

$$(u-v)^{\mathsf{T}}(x-y) \;\geq\; 0.$$

where $\text{graph}(F) := \{(x,u) \in \mathbb{R}^n \times \mathbb{R}^n \mid u \in F(x)\}$ denotes the graph of the operator, that is, the set of all input-output pairs.

Equivalently, in function form this means $(F(x)-F(y))^{\mathsf{T}}(x-y) \geq 0$ for all x,y [5]. Monotonicity is a generalization of subgradients of convex functions. When F is *strongly monotone* with constant $m > 0$, one has

$$(F(x)-F(y))^{\mathsf{T}}(x-y) \;\geq\; m\|x-y\|^2, \quad \forall x,y.$$

as shown in [5]. If, moreover, F is Lipschitz with constant L, it follows that

$$m\|x-y\|^2 \;\leq\; (F(x)-F(y))^{\mathsf{T}}(x-y) \;\leq\; L\|x-y\|^2,$$

providing tight bounds on the operator's behavior [5]. Such operators will be encountered when analyzing the dual gradient or augmented Lagrangian mappings. In particular, dual ascent methods correspond to finding a zero of a monotone operator derived from the Lagrangian, and their convergence can be established using monotonicity and related variational inequalities.

Also note that the inverse of a strongly monotone operator is Lipschitz continuous [5], which will be useful when updates of dual variables will be related back to primal changes. Set-valued mappings (subdifferentials, Lagrangian gradients, etc.) naturally fit into this monotone framework. In summary, monotonicity and its refinements will allow us to apply powerful fixed-point arguments: for example, the primal-dual update can often be seen as a resolvent or averaged operator of a monotone inclusion, guaranteeing convergence when standard contraction conditions may not hold [5].

With these foundational concepts, convexity, duality, Lipschitz stability, fixed-point theory, and monotonicity established, we are now prepared to review classical augmented Lagrangian-based decomposition methods, before presenting and analyzing the asynchronous Bertsekas augmented Lagrangian method in subsequent chapters.

3. Review of Augmented Lagrangian-Based Decomposition Methods

3.1. Classical Separable Problem Formulation

In nonlinear programming, the structure of an optimization problem plays a pivotal role in determining the tractability and the computational efficiency of the solution process. A particularly important structural property is *separability*, which naturally gives rise to a class of strategies known as *decomposition methods*.

Problem (1), (2) is said to be separable if it can be written as

$$\min_{x \in X} \quad \left[f(x) = \sum_{i=1}^{N} f_i(x_i) \right] \tag{4}$$

$$s.t. \quad h(x) = \sum_{i=1}^{N} h_i(x_i) = 0, \tag{5}$$

$$X = X_1 \times X_2 \times \ldots X_N,\ X_i \subseteq \mathbb{R}^{n_i},\ i = 1, \ldots, N \tag{6}$$

where $x_i \in \mathbb{R}^{n_i}$ are subvectors of decision variables, $x = (x_1, ..., x_N) \in \mathbb{R}^n$, $n = \sum_{i=1}^{N} n_i$, $h_i : \mathbb{R}^{n_i} \to \mathbb{R}^m$ are given vector functions that describe constraints, and $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$, $i = 1, ..., N$. The set X is a Cartesian product of the sets X_i in the corresponding subspaces.

This formulation lends itself to *decomposition* techniques, which aim to solve smaller, more manageable subproblems instead of tackling the full-scale problem directly. Decomposition strategies are generally classified into two categories:

- **Parallel decomposition**: The subproblems are independent and can be solved concurrently, that is, $h_i(x_i) = 0, \forall i$, where there are no coupling constraints between the decision variables x_i. This approach is common in block-separable problems, often solved using distributed dual decomposition or parallel subgradient methods.
- **Sequential (or coordinated) decomposition**: The subproblems are coupled and must be solved iteratively, coordinating shared information across iterations, that is, $\sum_{i=1}^{N} h_i(x_i) = 0$, where the constraint couples the decision variables x_i across subproblems. Techniques such as the Augmented Lagrangian methods and the Alternating Direction Method of Multipliers (ADMM) fall into this category.

Sequential decomposition methods naturally lead to the introduction of duality and coordination mechanisms. One common approach involves relaxing the coupling constraints using Lagrange multipliers.

3.2. Ordinary Lagrangian Relaxation Method

In the 1950s to solve the minimization problem (1),(2) in the convex case the Lagrangian relaxation method was proposed [6]–[8], using Lagrangian function $L : \mathbb{R}^{n+m} \to \mathbb{R}$ defined as

$$L(x, \lambda) = f(x) + \lambda^T h(x), \tag{7}$$

where $\lambda \in \mathbb{R}^m$ is the Lagrange multipliers vector. This method used the associated dual functional q for (7) given as

$$q(\lambda) = \min_{x \in X} L(x, \lambda) \tag{8}$$

According to the duality theory, the solution maximizes the dual function with respect to the Lagrange multiplier vector; thus, it is necessary to calculate,

$$\max_{\lambda \in \mathbb{R}^m} q(\lambda) \tag{9}$$

Since problem (1), (2) is separable, (7) can be written as:

$$L(x, \lambda) = \sum_{i=1}^{N} f_i(x_i) + \lambda^T \sum_{i=1}^{N} h_i(x_i) \tag{10}$$

$$= \sum_{i=1}^{N} L_i(x_i, \lambda) \tag{11}$$

This formulation enables us to decompose the original problem into smaller subproblems, each of which solves:

$$\min_{x_i \in X_i} \left[f_i(x_i) + \lambda^T h_i(x_i) \right], \quad \forall i = 1, \ldots, N, \tag{12}$$

while updating λ to enforce global feasibility. This yields the iterative update:

$$x_i^{k+1} = \arg\min_{x_i \in X_i} L_i \left(x_i, \lambda^k \right), \quad i = 1, ..., N \tag{13}$$

$$\lambda^{k+1} = \lambda^k + \alpha\, h \left(x^{k+1} \right) \tag{14}$$

where α is a scalar stepsize coefficient.

The major drawback of the Lagrangian relaxation method is the requirement that the problem should possess a strictly convex structure, which reduces its applicability. Applying it to non-convex problems, including problems with discrete variables, can result in a duality

gap, whereby the maximum of the dual function does not equal the minimum of the original problem.

3.3. Augmented Lagrangian Method

To further improve convergence and robustness of Problem Problem (1), (2), especially in non-convex or ill-conditioned problems, Powell [9], and Hestenes [10] added the squared norm of the equality constraints

$$L_\rho(x,\lambda) = f(x) + \lambda^T h(x) + \frac{\rho}{2}\left|\left|h(x)\right|\right|^2 \tag{15}$$

where $\rho > 0$ is a scalar, penalty parameter. This function is convex in the neighborhood of (x^*,λ^*) when ρ is taken sufficiently large and if at (x^*,λ^*) the second order sufficient optimality condition

$$z^T \nabla^2_{xx} L(x^*,\lambda^*) z > 0, \forall z \in \mathbb{R}^n, z \neq 0 \quad \text{such that} \quad \nabla h(x^*)^T z = 0 \tag{16}$$

is satisfied.

The method of multipliers algorithm for solving problem (1), (2), based on the Augmented Lagrangian, consists in the following iterations:

$$x^{k+1} = \min_{x \in X} L_\rho(x,\lambda^k) \tag{17}$$

$$\lambda^{k+1} = \lambda^k + \rho\, h(x^{k+1}) \tag{18}$$

Equation (15) is a nonseparable Lagrange function due to the last - quadratic - penalty term, hence decomposition algorithms cannot be applied directly to the dual equivalent of the problem (4)-(6). To avoid the nonseparability, some tricks are necessary, which are described in the following subsections. The most important conditions of the convergence of the presented algorithms to the optimal point x^* for sufficiently big ρ are that all functions from the problem statement are of class C^2 in a neighbourhood of the optimal point x^*, gradients of all active constraints are linearly independent at x^*, and that the second order sufficient optimality condition (16) is satisfied (for the ordinary Lagrangian of the problem).

3.4. Bertsekas Decomposition Method

Bertsekas [11] proposed a convexification method that preserves separability of constraints. It belongs to the proximal point algorithms family and may be interpreted as a multipliers method in disguise [12]. The penalty component expresses now not the square of the

constraints function norm, but the square of the distance between the x vector and a parameter vector s from its vicinity. In subsequent iterations this vector approaches x, leading to convergence, while preserving separability. The resulting Lagrangian is

$$L_\rho(x,\lambda,s) = f(x) + \lambda^T h(x) + \frac{\rho}{2}\|s-x\|^2 \tag{19}$$

This function is separable and locally convex for ρ from some interval provided that the above mentioned conditions are satisfied at point x^* [11]. In a decomposed form the augmented Lagrangian (19) can be written as

$$L_\rho(x,\lambda,s) = \sum_{i=1}^{N}\left[f_i(x_i) + \lambda^T h_i(x_i) + \frac{\rho}{2}\|s_i-x_i\|^2\right] \triangleq \sum_{i=1}^{N} L_{\rho_i}(x_i,\lambda,s_i) \tag{20}$$

where

$$L_{\rho_i}(x_i,\lambda,s_i) = f_i(x_i) + \lambda^T h_i(x_i) + \frac{\rho}{2}\|s_i-x_i\|^2 \tag{21}$$

The Bertsekas [11] approach was adapted by us to a MINLP problem. The most efficient proved to be a two-level version with additional scaling coefficient $\beta \ll 1$ in the formula for changing Lagrange multipliers. This algorithm may be described by the following steps:

$$x_i^{k+1} = \arg\min_{x_i \in X_i} L_{\rho_i}\left(x_i, \lambda^k, s_i^k\right), \quad i = 1,...,N \tag{22}$$

$$s_i^{k+1} = \xi\, s_i^k + (1-\xi)\, x_i^{k+1}, \quad i = 1,...,N \tag{23}$$

$$\lambda^{k+1} = \lambda^k + \beta\,\rho\, h\left(x^{k+1}\right) \tag{24}$$

where $\xi \in [0,1)$ is a relaxation parameter.

3.5. Tanikawa-Mukai Decomposition Method

When $X = \mathbb{R}^n$, to reduce the steps of (17),(18) a multiplier estimate function derived from the first order Karush-Kuhn-Tucker optimality condition [6] can be used [13], [14]:

$$\lambda(x) = \arg\min_{\lambda} \|\nabla_x L(x,\lambda)\|^2 = -\left[\nabla h(x)^T \nabla h(x)\right]^{-1} \nabla h(x)^T \nabla f(x), \tag{25}$$

This formula is particularly useful when it is possible to calculate analytically $\lambda(x)$, e.g., when functions $h(x)$ are linear. Then, having $\lambda(x)$, one can replace the two level algorithm (17),(18) with a single level one:

$$x^{k+1} = \min_x L_\rho(x,\lambda(x)) \tag{26}$$

In the case where $X = \mathbb{R}^n$ in the Bertsekas algorithm, the multiplier vector $\lambda(x)$ can be approximated using the direct formula (25). Unfortunately, when $\lambda(x)$ substitutes λ in

(19), the resulting Lagrangian will no longer be separable, because of the term $\lambda(x)^T h(x)$, containing products of functions of different x_i components, $i = 1,\ldots,N$. Hence, separability will be preserved if λ does not depend on x. To address this issue Tanikawa and Mukai [15] replaced $\lambda(x)$ by an approximate $\lambda(s)$ and added a penalty term $\eta h(s)^T M(s) h(x)$ so as to ensure that the minimum point $\hat{s}$ is closer than s to x^*. These improvements resulted in the formula

$$L_{\rho_\eta}(x,s) = f(x) + \frac{\rho}{2}\|x-s\|^2 + \left[\lambda(s)^T + \eta h(s)^T M(s)\right] h(x) \tag{27}$$

where $\eta > 0$ is scalar and $M(s)$ is a symmetric, positive-definite matrix whose elements are of C^1 class. The Lagrangian problem (27) is locally strictly convex in x in the neighborhood of x^* if ρ is taken large enough and it has a unique local minimum point s^* as a function of s.

In a decomposed form (27) can be written as

$$L_{\rho_\eta}(x,s) = \sum_{i=1}^{N} L_{\rho_{\eta_i}}(x_i,s) \tag{28}$$

where

$$L_{\rho_{\eta_i}}(x_i,s) = f_i(x_i) + \frac{\rho}{2}\left|\left|x_i - s_i\right|\right|^2 + \left(\lambda(s)^T + \eta h(s)^T M(s)\right) h_i(x_i) \tag{29}$$

Problem (4)-(6) for $X = \mathbb{R}^n$ can now be solved by a two-level algorithm,

$$x_i^{k+1} = \arg\min_{x_i} L_{\rho_{\eta_i}}(x_i, s^k), \quad i = 1,...,N \tag{30}$$

$$s^{k+1} = \xi_k s^k + (1-\xi_k) x^{k+1},\ \xi_k \in [0,1) \tag{31}$$

Tatjewski and Engelmann [16] extended this approach to a more general case of the problem (4)-(6), involving local sets X_i, given by inequality constraints

$$X_i = \{x_i \in \mathbb{R}^{n_i} | g_i(x_i) \leq 0\} \tag{32}$$

In this case the Lagrange multipliers vector was calculated from the following formula:

$$\lambda(x,\mu) = -\left[\nabla h(x)^T \nabla h(x)\right]^{-1} \nabla h(x)^T \left[\nabla f(x) + \nabla g(x)\mu\right], \tag{33}$$

where μ is the multipliers vector for local inequality constraints

$$g_i(x_i) \leq 0,\ i = 1,\ldots r \tag{34}$$

3.6. Tatjewski Decomposition Method

A different approach to handle the nonseparable term in the function (15) with local sets X_i given by (32), proposed by Tatjewski [17], consists in replacing the function (15) by

$$L_\rho(x,\lambda,s) = \sum_{i=1}^{N} f_i(x_i) + \lambda^T \sum_{i=1}^{N} h_i(x_i) + \frac{\rho}{2} \sum_{i=1}^{N} \left\| \sum_{j=1, j\neq i}^{N} h_j(s_j) + h_i(x_i) \right\|^2 \tag{35}$$

where $s = (s_1, s_2, ..., s_N) \in R^n$ is an approximation point. The function (35) is separable

$$L_\rho(x,\lambda,s) = \sum_{i=1}^{N} L_{\rho_i}(x_i,\lambda,s), \tag{36}$$

where

$$L_{\rho_i}(x_i,\lambda,s) = f_i(x_i) + \lambda^T h_i(x_i) + \frac{\rho}{2} \left\| \sum_{j=1, j\neq i}^{N} h_j(s_j) + h_i(x_i) \right\|^2 \tag{37}$$

The algorithm, adapted to a MINLP problem, retains a two-level structure as in the Bertsekas algorithm and includes an additional scaling factor $\beta \ll 1$ in the formula for updating the Lagrange multipliers:

$$x_i^{k+1} = \arg\min_{x_i \in X_i} L_{\rho_i}\left(x_i, \lambda^k, s^k\right), \quad i = 1, ..., N \tag{38}$$

$$s_i^{k+1} = \xi\, s_i^k + (1-\xi)\, x_i^{k+1}, \quad i = 1, ..., N \tag{39}$$

$$\lambda^{k+1} = \lambda^k + \beta\, \rho\, h\left(x^{k+1}\right) \tag{40}$$

where $\xi \in [0, 1)$ is a relaxation parameter.

3.7. SALA Decomposition Algorithm in ADMM Version

The Alternating Direction Method of Multipliers (ADMM) was proposed by Glowinski and Marrocco [18] and Gabay and Mercier [19]. In the basic version ADMM solves problems dependent on two subvectors of the decision variables vector - x and s, calculating the new estimates of the solutions (x^*, s^*) in the Gauss-Seidel fashion. For our particular, non-convex problem with local constraints, the best choice seems to be the Separated Augmented Lagrangian Algorithm (SALA) proposed by Hamdi et. al. [20]–[22].

In SALA first the problem (4)-(6) is reformulated into

$$\min_{x\in X,\,s} \quad \left[f(x) = \sum_{i=1}^{N} f_i(x_i) \right] \tag{41}$$

$$s.t. \quad h_i(x_i) = s_i, \quad i = 1,\ldots,N \tag{42}$$

$$\sum_{i=1}^{N} s_i = 0, \tag{43}$$

$$X = X_1 \times X_2 \times \ldots X_N,\ X_i \subseteq \mathbb{R}^{n_i}, i = 1,\ldots,N \tag{44}$$

where $s_i \in \mathbb{R}^m, i = 1,\ldots,N$ are additional, artificial variables. To this formulation the multiplier method with partial elimination of constraints [23] is then applied. The eliminated by means of dualization and a penalty are constraints (42), the constraints (43) are retained explicitly.

The Augmented Lagrangian for problem (41)-(44) with elimination of only (42) constraints will have the form (it can be easily proved that $\lambda_1^* = \lambda_2^* = \ldots = \lambda_{|L|}^* \triangleq \lambda^*$):

$$L_\rho(x,s,\lambda) = \sum_{i=1}^{N} f_i(x_i) + \sum_{i=1}^{N} \lambda^T (h_i(x_i) - s_i) + \frac{\rho}{2} \sum_{i=1}^{N} \|h_i(x_i) - s_i\|^2 \triangleq \sum_{i=1}^{N} L_{\rho_i}(x_i, s_i, \lambda) \tag{45}$$

where

$$L_{\rho_i}(x_i, s_i, \lambda) = f_i(x_i) + \lambda^T (h_i(x_i) - s_i) + \frac{\rho}{2} \|h_i(x_i) - s_i\|^2 \tag{46}$$

The SALA method of multipliers can be summed up in the following stages:

$$\left(x^{k+1}, s^{k+1}\right) = \arg \min_{\substack{x\in X \\ \{s|\sum_{i=1}^{N} s_i = 0\}}} L_\rho(x,s,\lambda^k) \tag{47}$$

$$\lambda^{k+1} = \lambda^k + \frac{\rho_k}{N} \sum_{i=1}^{N} h_i(x_i^{k+1}) \tag{48}$$

$$\rho_{k+1} = \alpha \rho_k, \quad \alpha \geq 1 \tag{49}$$

Using the ADMM approach for problem (47), the optimizations separate with respect to x and s. Furthermore, the optimization concerning x can be split into N subproblems, resulting in an ADMM algorithm featuring a Jacobi (i.e., parallelizable) step for the x vector:

$$x_i^{k+1} = \arg\min_{x_i \in X_i} L_{\rho_i}(x_i, s_i^k, \lambda^k), \quad i = 1, \ldots, N \tag{50}$$

$$\text{Calculate} \quad r^{k+1} = \sum_{i=1}^{N} h_i(x_i^{k+1}) \tag{51}$$

$$s^{k+1} = \arg \min_{\{s | \sum_{i=1}^{N} s_i = 0\}} \sum_{i=1}^{N} \left[(\lambda^k)^T \left(h_i(x_i^{k+1}) - s_i \right) + \frac{\rho_k}{2} \left\| h_i(x_i^{k+1}) - s_i \right\|^2 \right] \tag{52}$$

$$\equiv s_i^{k+1} = h_i(x_i^{k+1}) - \frac{1}{N} r^{k+1}, \quad i = 1, \ldots, N$$

$$\lambda^{k+1} = \lambda^k + \frac{\rho_k}{N} r^{k+1} \tag{53}$$

$$\rho_{k+1} = \alpha \rho_k, \quad \alpha \geq 1 \tag{54}$$

4. Formulation and Convergence Analysis of the Asynchronous Bertsekas Augmented Lagrangian Method

Consider this classical separable optimization problem:

$$\min_{x\in\mathbb{R}^n} \quad \left[f(x) = \sum_{i=1}^{N} f_i(x_i) \right] \tag{55}$$

$$s.t. \quad g_j(x) = \sum_{i=1}^{N} g_{ji}(x_i) \leq 0, \qquad j = 1,2,\ldots,m \tag{56}$$

where $x_i \in \mathbb{R}^{n_i}$ are the decision variables in partition i, $x = (x_1, x_2, \ldots, x_N) \in \mathbb{R}^n$, $g_{ji} : \mathbb{R}^{n_i} \to \mathbb{R}$ are the vector functions that describe the j-th constraint having subvectors of i-th partition as arguments, $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ are the objective function components having subvectors of partition i as arguments. The total number of vectors from all partitions sums to dimension $n = \sum_{i=1}^{N} n_i$, Hence, $g = [g_1, \cdots, g_m]^\top \in \mathbb{R}^m$ and $f \in \mathbb{R}$. According to Bertsekas [11], (55)-(56) may be solved by iterative solution of the problem for different s_i^k (57)-(58)

$$\min_{x\in\mathbb{R}} \quad \left[Q_\rho(x,s) = \sum_{i=1}^{N} Q_{\rho_i}(x_i,s_i) = f_i(x_i) + \frac{\rho}{2}\|x_i - s_i^k\|^2 \right] \tag{57}$$

$$s.t. \quad g_j(x) = \sum_{i=1}^{N} g_{ji}(x_i) \leq 0, \qquad j = 1,2,\ldots,m \tag{58}$$

where $s_i^k,\ i = 1, \cdots N$ are additional variables such that $s_i^k \in \mathbb{R}^{n_i}$. In this section, the asynchronous convergence algorithm of the Bertsekas Augmented Lagrangian Method will be developed for solving problems of the form (57) and (56) and will prove its convergence.

4.1. Definitions and Assumptions

The gradient $\nabla g_{ji}(x_i)$ is:

$$\nabla g_{ji}(x_i) = \begin{bmatrix} \frac{\partial g_{ji}}{\partial x_1} \\ \frac{\partial g_{ji}}{\partial x_2} \\ \vdots \\ \frac{\partial g_{ji}}{\partial x_{n_i}} \end{bmatrix} \tag{59}$$

The Hessian matrix $\nabla^2 g_{ji}(x_i)$ is given by:

$$\nabla^2 g_{ji}(x_i) = \begin{bmatrix} \frac{\partial^2 g_{ji}}{\partial x_1^2} & \frac{\partial^2 g_{ji}}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 g_{ji}}{\partial x_1 \partial x_{n_i}} \\ \frac{\partial^2 g_{ji}}{\partial x_2 \partial x_1} & \frac{\partial^2 g_{ji}}{\partial x_2^2} & \cdots & \frac{\partial^2 g_{ji}}{\partial x_2 \partial x_{n_i}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 g_{ji}}{\partial x_{n_i} \partial x_1} & \frac{\partial^2 g_{ji}}{\partial x_{n_i} \partial x_2} & \cdots & \frac{\partial^2 g_{ji}}{\partial x_{n_i}^2} \end{bmatrix} \tag{60}$$

Here, the element in the p-th row and q-th column of the Hessian matrix represents the mixed partial derivative $\frac{\partial^2 g_{ji}}{\partial x_p \partial x_q}$. Since $g_{ji}(x_i)$ is twice continuously differentiable (Assumption 2), the Hessian is symmetric [24], meaning $\frac{\partial^2 g_{ji}}{\partial x_p \partial x_q} = \frac{\partial^2 g_{ji}}{\partial x_q \partial x_p}$.

Consider the following assumptions:

Assumption 1. *The objective function, $f(x)$, is twice continuously differentiable, and there exists $\rho > 0$ such that the cost function $Q(x)$ is strictly convex. Also, $f(x)$ is bounded below, i.e., there exists a constant φ_f such that*

$$\varphi_f \leq f(x) \tag{61}$$

Assumption 2. *For $j = 1, \cdots, m$, $g_j(x)$ are convex and twice continuously differentiable, and there exists some x^* such that*

$$g_j(x^*) \leq 0 \tag{62}$$

Assumption 3. *The gradients $\nabla g_j(x^*)$, $\forall j = 1, \cdots, m$ are linearly independent, and hence there exist unique Lagrange multiplier vectors*

$$\mu^* = (\mu_1^*, \ldots, \mu_m^*)' \in \mathbb{R}_+^m \tag{63}$$

such that the following Kuhn-Tucker conditions hold:

$$\nabla f(x^*) + \sum_{j=1}^{m} \mu_j^* \nabla g_j(x^*) = 0 \tag{64}$$

Assumption 4. *There holds*

$$z_i' \Big[\nabla^2 f_i(x_i^*) + \sum_{j=1}^{m} \mu_j^* \nabla^2 g_{ji}(x_i^*) \Big] z_i > 0 \tag{65}$$

for all $z_i \in \mathbb{R}^{n_i}$ such that $z_i \neq 0$, $\nabla g_{ji}(x_i^*)' z_i = 0$, $\forall i = 1, \cdots, N$, $\forall j = 1, \cdots, m$

Assumption 5. *The gradients $\nabla g_{ji}(x_i)$ are uniformly bounded, i.e., $\forall i = 1, \cdots, N$, there exist constants $\varphi_{g_{ji}}$ such that:*

$$\|\nabla g_{ji}(x_i)\| \leq \varphi_{g_{ji}} \tag{66}$$

where the sum of the bounds on ∇g_{ji} is denoted by $\varphi_g = \sum_i \varphi_{g_i}$ *and* $\varphi_{g_i} = \sum_j \varphi_{g_{ji}}$

Assumption 6. *For all* $i = 1, \cdots, N$,

$$\sigma_{min}^{i} \geq \frac{1}{\varphi_{L_i}} \tag{67}$$

where σ_{min}^{i} is the smallest eigenvalue of $\nabla^2 f_i(x_i) + \rho I + \sum_{j=1}^{m} \mu_j \nabla^2 g_{ji}(x_i)$

4.2. Synchronous Algorithm

The synchronous algorithm is restated here for convenience and to facilitate comparison with the asynchronous version.

$$\min_{x \in \mathbb{R}^n} \quad \sum_{i=1}^{N} f_i(x_i) + \frac{\rho}{2} \|s_i - x_i\|^2 \tag{68}$$

$$s.t. \quad \sum_{i=1}^{N} g_{ji}(x_i) \leq 0, \quad j = 1, 2, \ldots, m \tag{69}$$

where $s_i \in \mathbb{R}$ are additional variables, $s = (s_1, ..., s_N)$. The Lagrangian is given as:

$$L_\rho(x, \mu, s) = \sum_{i=1}^{N} L_{\rho_i}(x_i, \mu, s_i) \tag{70}$$

where

$$L_{\rho_i}(x_i, \mu, s_i) = f_i(x_i) + \frac{\rho}{2} \|s_i - x_i\|^2 + \sum_{j=1}^{m} \mu_j g_{ji}(x_i) \tag{71}$$

The result of Bertsekas [11] approach is in the basic version, the following algorithm: At iteration $k = 1, 2, \ldots$

$$x_i^{k+1} = \arg\min_{x_i \in \mathbb{R}^{n_i}} L_{\rho_i}\left(x_i, \mu^k, s_i^k\right), \quad i = 1, ..., N \tag{72}$$

$$s_i^{k+1} = \zeta s_i^k + (1 - \zeta) x_i^{k+1}, \quad i = 1, ..., N \tag{73}$$

$$\mu_j^{k+1} = \left[\mu_j^k + \gamma \sum_{i=1}^{N} g_{ji}(x_i^{k+1})\right]^+, \quad j = 1, ..., m \tag{74}$$

where

$$\gamma = \beta \rho \tag{75}$$

$$\zeta = [0, 1) \tag{76}$$

and $0 < \beta < 1$ is a step coefficient and $[z]^+ = \max\{z, 0\}$. Generally, the unique minimizers of (72), $\left(x_i(\mu^k, s_i^k),\ i = 1, \ldots, N\right)$ may not be primal optimal. By applying duality theory

[25], there exists a dual optimal $\mu^* \geq 0$ and s^* such that $(x_i(\mu^*, s_i^*),\ i = 1,\ldots,N)$ is indeed primal optimal.

For larger values of ρ, the amount of convexification is higher, the region of convergence is smaller, the rate of convergence is faster, the conditioning of Problem (68)-(69) is better. The problem is thus easier to solve algorithmically.

Algorithm 1 Synchronous Version

Objective i's Algorithm: At iteration, $k = 1,2,...,$ *and* i:

1: receives μ_j^k from all $j = 1,\ldots,m$
2: determines the next x_i^{k+1} and s_i^{k+1} using equations (72) and (73) respectively.
3: communicates x_i^{k+1} to all $j = 1,\ldots,m$

Constraint j's Algorithm: At iteration, $k = 1,2,...,$ *and* j:

1: receives x_i^{k+1} from all $i = 1,\ldots,N$
2: computes the next μ_j^{k+1} using equation (74)
3: communicates μ_j^{k+1} to all $i = 1,\ldots,N$

4.3. Asynchronous Algorithm

This section formulates the asynchronous version of the Bertsekas algorithm. The indices i and j represent processors in a distributed computing system. In each iteration, the ith local optimizer independently solves (72) and communicates its result x_i to all constraint controllers. Each jth constraint controller then updates μ_j according to (74) and sends the updated μ_j to all local optimizers. This cycle then repeats. An additional assumption required to ensure the convergence of the asynchronous method is presented below.

Assumption 7. *Some integer k^0 bounds the delays in receiving updates*

Let $S_i \subseteq \{1,2,\ldots,\}$ be a set of iterations at which i adjusts x_i based on current knowledge of $\mu_j\ \forall j = 1,\ldots,m$ and $T_j \subseteq \{1,2,\ldots,\}$ be a set of iterations at which j adjusts μ_j based on current knowledge of $x_i\ \forall i = 1,\ldots,N$. At iterations $k \in T_j$, j computes an estimate μ_j^k of the derivative of the argument Lagrangian with respect to the j-th multiplier and updates

$$\mu_j^{k+1} = \left[\mu_j^k + \beta\rho\mu_j^k\right]^+ \tag{77}$$

using aggregate past constraint functions,

$$\mu_j^k = \sum_i g_{ji}(\hat{x}_i^k) \tag{78}$$

where

$$\hat{x}_i^k = \sum_{k'=k-k^0}^{k} a_i(k',k) x_i^{k'}, \quad i = 1,\ldots,N \tag{79}$$

$$\sum_{k'=k-k^0}^{k} a_i(k',k) = 1, \quad \forall k, \forall i = 1,\ldots,N \tag{80}$$

and

$$a_i(k',k) \geq 0, \quad \forall k, \forall i = 1,\ldots,N \tag{81}$$

At iterations $k \in S_i$, i-th local optimizer updates x_i^k and s_i^k according to

$$x_i^{k+1} = \arg\min_{x_i} L_{\rho_i}(x_i, s_i^k, \hat{\mu}^k) \tag{82}$$

$$s_i^{k+1} = \zeta s_i^k + (1-\zeta) x_i^{k+1} \tag{83}$$

where

$$\hat{\mu}^k = \left(\hat{\mu}_j^k, j = 1,\ldots,m\right) \tag{84}$$

$$\hat{\mu}_j^k = \sum_{k'=k-k^0}^{k} b_j(k',k) \mu_j^{k'}, \quad j = 1,\ldots,m \tag{85}$$

$$\sum_{k'=k-k^0}^{k} b_j(k',k) = 1, \quad \forall k, \forall j = 1,\ldots,m \tag{86}$$

and

$$b_j(k',k) \geq 0, \quad \forall k, \forall j = 1,\ldots,m \tag{87}$$

As in the previous case, $\hat{\mu}^k$ is obtained by "averaging" over the most recently available values, as defined in (85). Note that (79) and (85) assume a maximum one-way communication delay of k^0 between any pair of agents. The asynchronous version is detailed in Algorithm 2. Unlike Algorithms 1, communication in this case is uncoordinated, and computations are performed using potentially outdated information.

4.4. Convergence Analysis of the Synchronous Algorithm

Theorem 1. *Suppose that Assumptions 1-5 hold. Then starting from any initial primal values x^0 and dual values $\mu^0 \geq 0$, every accumulation point (x^*, s^*, μ^*) of the sequence (x^k, s^k, μ^k) generated by Algorithm 1 is primal-dual optimal.*

Proof (Theorem 1). See the convergence analysis in Bertsekas paper ([11], Section 2). □

Algorithm 2 Asynchronous Version

i's Algorithm: At iterations, $k = 1, 2, ...,$ *and* i:

1: From iteration to iteration, i receives updates μ_j^k from all $j = 1, \ldots, m$ and compute $\hat{\mu}^k$.
2: At each update iteration $k' \in S_i$, i chooses the next x_i^{k+1} and s_i^{k+1}
$x_i^{k+1} = \arg\min_{x_i} L_{\rho_i}\left(x_i, \hat{\mu}^k, s_i^k\right)$
$s_i^{k+1} = \zeta s_i^k + (1-\zeta) x_i^{k+1}$
Transmission is done at this rate until the next update
3: communicates x_i^{k+1} to all constraints controllers.

j's Algorithm: At iterations, $k = 1, 2, ...,$ *and* j:

1: From iteration to iteration, j receives updates x_i^k from all $i = 1, \ldots, N$ and compute $\hat{x}_i^{k+1}$.
2: At each update iteration $k' \in T_j$, j updates μ_j^{k+1} such that
$\mu_j^{k+1} = \left[\mu_j^k + \gamma \sum_{i=1}^N g_{ji}(\hat{x}_i^{k+1})\right]^+$
Transmission is done at this rate until the next update
3: communicates μ_j^{k+1} to all local optimizers.

4.5. Convergence Analysis of the Asynchronous Algorithm

Define $\pi^k = \mu^k - \mu^{k+1}$ and $\hat{\mu}^k = (\hat{\mu}_j^k,\ j = 1, \ldots, m)$ where $\hat{\mu}_j^k$ is defined in (85). Let's denote the primal variable estimate as

$$\begin{aligned} x_i^k &= x_i(\hat{\mu}^k, s_i^k) \\ s_i^{k+1} &= \zeta s_i^k + (1-\zeta) x_i^{k+1} \end{aligned} \tag{88}$$

and the corresponding exact as

$$\begin{aligned} \bar{x}_i^k &= x_i(\mu^k, s_i^k) \\ \bar{s}_i^{k+1} &= \zeta \bar{s}_i^k + (1-\zeta) \bar{x}_i^{k+1} \end{aligned} \tag{89}$$

Theorem 2. *Suppose that Assumptions 1-5 and 7 hold. Provided that the stepsize is sufficiently small, then starting from any initial primal values* x^0 *and dual values* $\mu^0 \geq 0$*, every accumulation point* (x^*, s^*, μ^*) *of the sequence* (x^k, s^k, μ^k) *generated by Algorithm 2 are primal-dual optimal. Moreover, the errors in price estimation* $\|\hat{\mu}^k - \mu^k\|$*, and the gradient estimation* $\|\mu^k - \nabla d_{\rho,s}(\mu^k)\|$ *converges to zero. Also for all* $i = 1, \ldots, N$ *the errors in the primal value calculation* $\|\hat{x}_i^k - \bar{x}_i^k\|$ *and the approximate point value calculation* $\|\hat{s}_i^k - \bar{s}_i^k\|$ *converges to zero.*

4.5.1. Useful Lemmas

Lemma 1. *For all $i = 1, \ldots, N$, for any Lagrangian vector $\mu \in \mathbb{R}^m_+$ and for sufficiently large $\rho > 0$, the matrix*

$$\nabla^2 f_i(x_i(\mu, s_i)) + \rho I + \sum_{j=1}^{m} \mu_j \nabla^2 g_{ji}(x_i(\mu, s_i)) \tag{90}$$

is positive definite. Specifically, there exists a constant $\varphi_{L_i} > 0$ such that

$$\left\| \left[\nabla^2 f_i(x_i(\mu, s_i)) + \rho I + \sum_{j=1}^{m} \mu_j \nabla^2 g_{ji}(x_i(\mu, s_i)) \right]^{-1} \right\| \leq \varphi_{L_i}. \tag{91}$$

Proof. To prove this lemma, we rely on the fact that for sufficiently large ρ, the term ρI dominates the matrix $\nabla^2 f_i(x_i) + \rho I$, ensuring that the entire matrix is positive definite. This is because ρI adds a large positive value to the diagonal elements, making the matrix positive definite regardless of the properties of $\nabla^2 f_i(x_i)$.

Next, consider the term $\sum_{j=1}^{m} \mu_j \nabla^2 g_{ji}(x_i(\mu, s_i))$. Since $\mu_j \geq 0$ and $\nabla^2 g_{ji}(x_i)$ are symmetric nonnegative definite matrices, the weighted sum $\sum_{j=1}^{m} \mu_j \nabla^2 g_{ji}(x_i(\mu, s_i))$ does not reduce the positive definiteness of $\nabla^2 f_i(x_i) + \rho I$. Therefore, the matrix

$$\nabla^2 f_i(x_i(\mu, s_i)) + \rho I + \sum_{j=1}^{m} \mu_j \nabla^2 g_{ji}(x_i(\mu, s_i)) \tag{92}$$

remains positive definite. Given that the matrix is positive definite, its inverse exists and is bounded. To show this, Let $A_i(\rho, \mu, s_i) = \nabla^2 f_i(x_i(\mu, s_i)) + \rho I + \sum_{j=1}^{m} \mu_j \nabla^2 g_{ji}(x_i(\mu, s_i))$ be a positive definite matrix and let it's eigenvalues be $\sigma_1^i(\rho, \mu, s_i), \sigma_2^i(\rho, \mu, s_i), \ldots, \sigma_{n_i}^i(\rho, \mu, s_i)$, where $0 < \sigma_1^i(\rho, \mu, s_i) \leq \sigma_2^i(\rho, \mu, s_i) \leq \cdots \leq \sigma_{n_i}^i(\rho, \mu, s_i)$. It can be shown that the inverse of $A_i(\rho, \mu, s_i)$, denoted as $A_i(\rho, \mu, s_i)^{-1}$, is bounded by φ_{L_i}, assuming $\frac{1}{\varphi_{L_i}}$ is the lower bound on the eigenvalues of $A_i(\rho, \mu, s_i)$. Since $A_i(\rho, \mu, s_i)$ is positive definite, it is also invertible, and its inverse $A_i(\rho, \mu, s_i)^{-1}$ is also symmetric and positive definite. The eigenvalues of $A_i(\rho, \mu, s_i)^{-1}$ are given by $\frac{1}{\sigma_1^i(\rho, \mu, s_i)}, \frac{1}{\sigma_2^i(\rho, \mu, s_i)}, \ldots, \frac{1}{\sigma_{n_i}^i(\rho, \mu, s_i)}$. The spectral norm of $A_i(\rho, \mu, s_i)^{-1}$, $\|A_i(\rho, \mu, s_i)^{-1}\|$, is the largest eigenvalue of $A_i(\rho, \mu, s_i)^{-1}$. Using Assumption 6,

$$\|A_i(\rho, \mu, s_i)^{-1}\| = \max_{1 \leq l \leq n_i} \left| \frac{1}{\sigma_l^i(\rho, \mu, s_i)} \right| = \frac{1}{\sigma_1^i(\rho, \mu, s_i)} \leq \varphi_{L_i} \tag{93}$$

Hence, there exists a constant $\varphi_{L_i} > 0$ such that:

$$\left\| \left[\nabla^2 f_i(x_i(\mu, s_i)) + \rho I + \sum_{j=1}^{m} \mu_j \nabla^2 g_{ji}(x_i(\mu, s_i)) \right]^{-1} \right\| \leq \varphi_{L_i}. \tag{94}$$

□

Lemma 2. *Given the dual objective function as:*

$$d_{\rho,s}(\mu) = \min_x L_\rho(x,\mu,s), \tag{95}$$

under Assumptions 1-2, $d_\rho(\mu,s)$ is concave, upper bounded, and continuously differentiable.

Proof. Given the dual objective function:

$$d_{\rho,s}(\mu) = \min_x L_\rho(x,\mu,s), \tag{96}$$

To show concavity, we start by recalling that the Lagrangian $L_\rho(x,\mu,s)$ is given by:

$$L_\rho(x,\mu,s) = Q_\rho(x,s) + \sum_{j=1}^{m} \mu_j g_j(x), \tag{97}$$

where $Q_\rho(x,s)$ is the objective function given in (57) and $g_j(x) \leq 0$ are the constraints. The function $d_{\rho,s}(\mu)$ is defined as the minimum value of the Lagrangian over x:

$$d_{\rho,s}(\mu) = \min_x \left[Q_\rho(x,s) + \sum_{j=1}^{m} \mu_j g_j(x) \right]. \tag{98}$$

For any two vectors μ^1 and μ^2, and any $\theta \in [0,1]$, consider $\mu^\theta = \theta\mu^1 + (1-\theta)\mu^2$. Using the definition of $d_{\rho,s}(\mu)$:

$$d_{\rho,s}(\mu^\theta) = \min_x \left[Q_\rho(x,s) + \sum_{j=1}^{m} \mu_j^\theta g_j(x) \right]. \tag{99}$$

Since $\mu_j^\theta = \theta\mu_j^1 + (1-\theta)\mu_j^2$, we have:

$$d_{\rho,s}(\mu^\theta) = \min_x \left[Q_\rho(x,s) + \sum_{j=1}^{m} (\theta\mu_j^1 + (1-\theta)\mu_j^2) g_j(x) \right]. \tag{100}$$

Using the properties of the minimum function:

$$d_{\rho,s}(\mu^\theta) \geq \theta \min_x \left[Q_\rho(x,s) + \sum_{j=1}^{m} \mu_j^1 g_j(x) \right] + (1-\theta) \min_x \left[Q_\rho(x,s) + \sum_{j=1}^{m} \mu_j^2 g_j(x) \right]. \tag{101}$$

Thus,

$$d_{\rho,s}(\mu^\theta) \geq \theta d_{\rho,s}(\mu^1) + (1-\theta) d_{\rho,s}(\mu^2). \tag{102}$$

This confirms that $d_{\rho,s}(\mu)$ is concave.

The dual function $d_{\rho,s}(\mu)$ is upper bounded because it represents the minimum value of the Lagrangian over x, given the constraints. Since $\mu \geq 0$ (from the definition of Lagrange multipliers), the terms involving $\mu_j g_j(x)$ are non-positive when $g_j(x) \leq 0$. Therefore, $d_{\rho,s}(\mu)$

is bounded above by the value of the primal objective function at some feasible x, ensuring it does not go to infinity.

To show that $d_{\rho,s}(\mu)$ is continuously differentiable, the KKT conditions and the implicit function theorem are considered. The first-order optimality conditions for the minimization problem

$$d_{\rho,s}(\mu) = \min_x \left[Q_\rho(x,s) + \sum_{j=1}^{m} \mu_j g_j(x) \right] \tag{103}$$

are given by:

$$\nabla_x Q_\rho(x(\mu,s),s) + \sum_{j=1}^{m} \mu_j \nabla g_j(x(\mu,s)) = 0, \tag{104}$$

The Hessian of the Lagrangian with respect to x, denoted by $\nabla^2_{xx} L_\rho(x(\mu,s),\mu,s)$, is assumed to be positive definite due to the strict convexity and twice continuous differentiability of $Q_{\rho_i}(x_i,s_i)$ and $g_{ji}(x_i)$.By the implicit function theorem, the optimal solution $x(\mu,s)$ is a continuously differentiable function of μ. Thus, the dual function $d_{\rho,s}(\mu) = Q_\rho(x(\mu,s),s) + \sum_{j=1}^{m} \mu_j g_j(x(\mu,s))$ is also continuously differentiable with respect to μ and s.

Therefore, under the given assumptions, $d_{\rho,s}(\mu)$ is concave, upper bounded, and continuously differentiable.

□

Lemma 3. *Under Assumptions 1-2 and 4, the Hessian of d_ρ is given by*

$$\nabla^2 d_{\rho,s}(\mu) = -\nabla g(x(\mu,s))^\top \left(\nabla^2 L_\rho(x(\mu,s),\mu,s) \right)^{-1} \nabla g(x(\mu,s)). \tag{105}$$

where it exists.

Proof. The minimizer $x(\mu,s)$ of $L_\rho(x,\mu,s)$ from (70) satisfies the first-order optimality condition:

$$\nabla_x L_\rho(x(\mu,s),\mu,s) = 0. \tag{106}$$

Computing the gradient of $L_\rho(x,\mu,s)$ with respect to x and setting it to zero at $x = x(\mu,s)$ gives:

$$\nabla f(x(\mu,s)) + \rho(x(\mu,s) - s) + \sum_{j=1}^{m} \mu_j \nabla g_j(x(\mu,s)) = 0. \tag{107}$$

For the dual function:

$$d_{\rho,s}(\mu) = L_\rho(x(\mu,s),\mu,s), \tag{108}$$

the gradient is:

$$\nabla d_{\rho,s}(\mu) = \frac{\partial L_\rho(x(\mu,s),\mu,s)}{\partial \mu} = g(x(\mu,s)). \tag{109}$$

To find $\nabla^2 d_{\rho,s}(\mu)$, we differentiate $g(x(\mu,s))$ with respect to μ:

$$\nabla^2 d_{\rho,s}(\mu) = \frac{\partial g(x(\mu,s))}{\partial \mu} = \nabla g(x(\mu,s))^\top \frac{dx(\mu,s)}{d\mu}. \tag{110}$$

Differentiating the first-order optimality condition:

$$\frac{d}{d\mu}\left[\nabla f(x(\mu,s)) + \rho(x(\mu,s)-s) + \sum_{j=1}^{m} \mu_j \nabla g_j(x(\mu,s))\right] = 0, \tag{111}$$

we get:

$$\left(\nabla^2 f(x(\mu,s)) + \rho I + \sum_{j=1}^{m} \mu_j \nabla^2 g_j(x(\mu,s))\right) \frac{dx(\mu,s)}{d\mu} + \nabla g(x(\mu,s)) = 0. \tag{112}$$

Solving for $\frac{dx(\mu,s)}{d\mu}$ in (112):

$$\frac{dx(\mu,s)}{d\mu} = -\left(\nabla^2 f(x(\mu,s)) + \rho I + \sum_{j=1}^{m} \mu_j \nabla^2 g_j(x(\mu,s))\right)^{-1} \nabla g(x(\mu,s)). \tag{113}$$

Substituting into the expression (110) for $\nabla^2 d_{\rho,s}(\mu)$:

$$\begin{aligned}\nabla^2 d_{\rho,s}(\mu) &= -\nabla g(x(\mu,s))^\top \left(\nabla^2 f(x(\mu,s)) + \rho I + \sum_{j=1}^{m} \mu_j \nabla^2 g_j(x(\mu,s))\right)^{-1} \nabla g(x(\mu,s)) \\ &= -\nabla g(x(\mu,s))^\top \left(\nabla^2 L_\rho(x(\mu,s),\mu,s)\right)^{-1} \nabla g(x(\mu,s)).\end{aligned} \tag{114}$$

□

Lemma 4. *Under Assumptions 1-2 with Lemma 1, ∇d_ρ is Lipschitz with*

$$\|\nabla d_{\rho,s}(q) - \nabla d_{\rho,s}(\mu)\| \le \varphi_L \varphi_g \|q-\mu\|. \tag{115}$$

for all $q, \mu \ge 0$

Proof. Given any $q \ge 0, \mu \ge 0$, s and for some $w = t\mu + (1-t)q \ge 0,\ t \in [0,1]$, applying the mean value theorem, Lemma 1 and Assumption 5,

$$
\begin{aligned}
\left\|\nabla d_{\rho,s}(q)-\nabla d_{\rho,s}(\mu)\right\|_1 &= \sum_{j=1}^{m}\left|\left[\nabla d_{\rho,s}(q)\right]_j-\left[\nabla d_{\rho,s}(\mu)\right]_j\right| \\
&= \sum_{j=1}^{m}\left|g_j(x(q,s))-g_j(x(\mu,s))\right| \\
&\leq \sum_{j=1}^{m}\left|\frac{\partial g_j(x(w,s))}{\partial w_j}(q_j-\mu_j)\right| \\
&= \sum_{j=1}^{m}\left|\left[\nabla^2 d_\rho(w,s)\right]_j\right|\left|q_j-\mu_j\right| \\
&= \left\|\nabla^2 d_\rho(w,s)\right\|_1\|q-\mu\|_1 \\
&= \left\|-\nabla g\left(x(w,s)\right)^\top\left[\nabla^2 L_\rho(x(w,s))\right]^{-1}\nabla g\left(x(w,s)\right)\right\|_1\|q-\mu\|_1 \\
&\leq \|\nabla g\left(x(w,s)\right)\|_1\left\|\left[\nabla^2 L_\rho(x(w,s))\right]^{-1}\right\|_1\|\nabla g\left(x(w,s)\right)\|_1\|q-\mu\|_1 \\
&\leq \varphi_L\varphi_g\|q-\mu\|_1
\end{aligned}
\tag{116}
$$

□

Lemma 5.

1. *For all k,* $-{\mu^k}^T\pi^k \geq \frac{1}{\gamma}\|\pi^k\|_2^2$
2. *There exists a constant* $A_1>0$ *such that, for all* $\mu\geq 0$ *and all q, we have* $q^T\nabla^2 d_{\rho,s}(\mu)q \geq -2A_1\|q\|^2$
3. *For any* $k'=k-k^0,\ldots,k;\quad$ *for all* $l=1,\ldots,n_i$*;* $0\leq\left|\frac{\partial x_i(\hat{\mu}^k,s_i^k)}{\partial b_j(k',k)}\right|_l\leq\varphi_{L_i}\varphi_{g_{ji}}\mu_j^{k'}$ *where it exists.*
4. *For all k,* $\left\|x_i(\hat{\mu}^k,s_i^k)-x_i(\mu^k,s_i^k)\right\|_1\leq n_i\varphi_{L_i}\varphi_{g_i}\sum_{k'=k-k^0}^{k-1}\sum_{j=1}^{m}\left|\pi_j^{k'}\right|$
5. *For all k,* $\left\|x_i(\mu^k,s_i^k)-x_i(\mu^\tau,s_i^\tau)\right\|_1\leq n_i\varphi_{L_i}\varphi_{g_i}\sum_{k'=\tau}^{k-1}\sum_{j=1}^{m}|\pi_j^{k'}|$

Proof (Lemma 5.1). For $k\in S$, applying the projection theorem ([26], Proposition 2.1.3) to the scalar $\mu_j^{k+1}=\left[\mu_j^k+\gamma\mu_j^k\right]^+$, we have

$$
\begin{aligned}
\left(\mu_j^k-\mu_j^{k+1}\right)\left(\mu_j^k+\gamma\mu_j^k-\mu_j^{k+1}\right)&\leq 0 \\
\pi_j^k\left(\pi_j^k+\gamma\mu_j^k\right)&\leq 0 \\
\mu_j^k\pi_j^k&\leq\frac{-1}{\gamma}\left(\pi_j^k\right)^2
\end{aligned}
\tag{117}
$$

Summing over j we have

$$
-{\mu^k}^T\pi^k\geq\frac{1}{\gamma}\|\pi^k\|_2^2 \tag{118}
$$

□

Proof (Lemma 5.2). From Lemma 3, $\nabla^2 d_{\rho,s}(\mu) = -v^\top H v$, where $v = \nabla g(x(\mu,s))$ and $H = \left[\nabla^2 L_\rho(x(\mu,s),\mu,s)\right]^{-1}$. Let $C = -\nabla^2 d_{\rho,s}(\mu) = v^\top H v$. By sub-multiplicative properties of matrix norms,

$$\|C\| = \|v^\top H v\| \leq \|v\|\|H\|\|v\| = \|v\|^2\|H\|. \tag{119}$$

Since C is symmetric, by ([26], Appendix A, Proposition A.18) we have:

$$q^T C q \leq \sigma_{\max}(C)\,\|q\|^2 = \|C\|\|q\|^2 \tag{120}$$

where $\sigma_{\max}(C)$ is the largest eigenvalue of the matrix C and $q \in R^m$. Therefore,

$$\begin{aligned} q^\top C q &\leq \|\nabla g(x(\mu,s))\|^2 \left\| \left[\nabla^2 L_\rho(x(\mu,s),\mu,s)\right]^{-1} \right\| \|q\|^2 \\ &\leq \varphi_L \varphi_g^2 \|q\|^2 \end{aligned} \tag{121}$$

Since $C = -\nabla^2 d_{\rho,s}(\mu)$,

$$-q^\top \nabla^2 d_{\rho,s}(\mu) q \leq \varphi_L \varphi_g^2 \|q\|^2 \tag{122}$$

For $A_1 = \varphi_L \varphi_g^2 > 0$

$$q^\top \nabla^2 d_{\rho,s}(\mu) q \geq -A_1\|q\|^2 \geq -2A_1\|q\|^2 \tag{123}$$

□

Proof (Lemma 5.3). Applying the chain rule, we have

$$\frac{\partial x_i(\hat{\mu}^k, s_i^k)}{\partial b_j(k',k)} = \frac{\partial x_i(\hat{\mu}^k, s_i^k)}{\partial \hat{\mu}_j^k} \cdot \mu_j^{k'} \tag{124}$$

Similar to (113) we have,

$$\frac{\partial x_i(\hat{\mu}^k, s_i^k)}{\partial \hat{\mu}_j^k} = -\left[\nabla^2 f_i(x_i(\hat{\mu}^k, s_i^k)) + \rho I + \sum_{j'=1}^{m} \hat{\mu}_{j'}^k \nabla^2 g_{j'i}(x_i(\hat{\mu}^k, s_i^k))\right]^{-1} \nabla g_{ji}(x_i(\hat{\mu}^k, s_i^k)). \tag{125}$$

From Lemma 1, we know that the inverse of the Hessian is bounded, specifically:

$$\left\| \left[\nabla^2 f_i(x_i(\hat{\mu}^k, s_i^k)) + \rho I + \sum_{j'=1}^{m} \hat{\mu}_{j'}^k \nabla^2 g_{j'i}(x_i(\hat{\mu}^k, s_i^k))\right]^{-1} \right\| \leq \varphi_{L_i} \tag{126}$$

And from Assumption 5, the gradient $\nabla g_{ji}(x_i(\hat{\mu}^k, s_i^k))$ is bounded by $\varphi_{g_{ji}}$. Combining these and since $\mu > 0$, we get:

$$\begin{aligned}\left\|\frac{\partial x_i(\hat{\mu}^k, s_i^k)}{\partial b_j(k',k)}\right\| &= \left\|\frac{\partial x_i(\hat{\mu}^k, s_i^k)}{\partial \hat{\mu}_j^k} \cdot \mu_j^{k'}\right\| \\ &\leq \left\|\left[\nabla^2 f_i(x_i(\hat{\mu}^k, s_i^k)) + \rho I + \sum_{j'=1}^{m} \hat{\mu}_{j'}^k \nabla^2 g_{j'i}(x_i(\hat{\mu}^k, s_i^k))\right]^{-1}\right\| \left\|\nabla g_{ji}(x_i(\hat{\mu}^k, s_i^k))\right\| \mu_j^{k'} \\ &\leq \varphi_{L_i} \varphi_{g_{ji}} \mu_j^{k'}\end{aligned} \tag{127}$$

For any component l, we can bound the magnitude by the norm of the vector:

$$\left[\frac{\partial x_i(\hat{\mu}^k, s_i^k)}{\partial b_j(k',k)}\right]_l \leq \left|\left[\frac{\partial x_i(\hat{\mu}^k, s_i^k)}{\partial b_j(k',k)}\right]_l\right| \leq \left\|\frac{\partial x_i(\hat{\mu}^k, s_i^k)}{\partial b_j(k',k)}\right\| \leq \varphi_{L_i} \varphi_{g_{ji}} \mu_j^{k'} \tag{128}$$

□

Proof (Lemma 5.4). For $\hat{\mu}_j^k$ from (85), and $\forall l = 1, \ldots, n_i$ define

$$\begin{aligned} u_{il}(\varepsilon^k; \mu^k) &= \left[x_i(\hat{\mu}^k, s_i^k)\right]_l \\ u_{il}(\mathbf{1}^k; \mu^k) &= \left[x_i(\mu^k, s_i^k)\right]_l \end{aligned} \tag{129}$$

where $\mathbf{1}^k \in \mathbb{R}_+^{m(k^0+1)}$ is defined by

$$\mathbf{1}_{jk'}^k = \begin{cases} 1 & \text{if } k' = k, \\ 0 & \text{otherwise.} \end{cases} \tag{130}$$

and $\varepsilon^k \in \mathbb{R}_+^{m(k^0+1)}$ is defined by

$$\varepsilon_{jk'}^k = b_j(k',k) \tag{131}$$

Using the definitions (129) and by the mean value theorem, for some $\tilde{\varepsilon}_l$ between ε^k and $\mathbf{1}^k$, we have:

$$\begin{aligned}\left\|x_i(\hat{\mu}^k, s_i^k) - x_i(\mu^k, s_i^k)\right\|_1 &= \sum_l \left|\left[x_i(\hat{\mu}^k, s_i^k)\right]_l - \left[x_i(\mu^k, s_i^k)\right]_l\right| \\ &= \sum_l \left|u_{il}(\varepsilon^k; \mu^k) - u_{il}(\mathbf{1}^k; \mu^k)\right| \\ &= \sum_l \left|\sum_{(j,k')} \frac{\partial u_{il}}{\partial \varepsilon_{jk'}}\left(\tilde{\varepsilon}_l; \mu^k\right)\left(\varepsilon_{jk'}^k - \mathbf{1}_{jk'}^k\right)\right| \\ &= \sum_l \left|\sum_{(j,k')} \left[\frac{\partial x_i(\hat{\mu}^k, s_i^k)}{\partial b_j(k',k)}\right]_l \left[b_j(k',k) - \mathbf{1}_{jk'}^k\right]\right|\end{aligned} \tag{132}$$

Using the bound on $\frac{\partial u_{il}}{\partial \varepsilon_{jk'}}$ from Lemma 5.3 we have:

$$\left[\frac{\partial x_i(\hat{\mu}^k, s_i^k)}{\partial b_j(k',k)}\right]_l \leq \varphi_{L_i}\varphi_{g_{ji}}\mu_j^{k'} \tag{133}$$

Applying the above bound and the fact that $\varphi_{L_i}\varphi_{g_{ji}} \geq 0$, we can write:

$$\begin{aligned}\left\|x_i(\hat{\mu}^k, s_i^k) - x_i(\mu^k, s_i^k)\right\|_1 &\leq \sum_l \left|\sum_{(j,k')} \varphi_{L_i}\varphi_{g_{ji}}\mu_j^{k'}\left[b_j(k',k) - \mathbf{1}_{jk'}^k\right]\right| \\ &= n_i\varphi_{L_i}\varphi_{g_i}\left|\sum_{(j,k')} \mu_j^{k'}\left[b_j(k',k) - \mathbf{1}_{jk'}^k\right]\right|\end{aligned} \tag{134}$$

Using (130) and the definition of π we have:

$$\begin{aligned}\left|\sum_{(j,k')} \mu_j^{k'}\left[b_j(k',k) - \mathbf{1}_{jk'}^k\right]\right| &\leq \sum_{j=1}^m \left|\sum_{k'=k-k^0}^{k} \mu_j^{k'} b_j(k',k) - \mu_j^k\right| \\ &\leq \sum_{j=1}^m \max_{k-k^0\leq k'\leq k}\left|\mu_j^{k'} - \mu_j^k\right| \\ &\leq \sum_{j=1}^m \max_{k-k^0\leq k'\leq k}\sum_{\tau=k'}^{k-1}\left|\pi_j^{\tau}\right| \\ &\leq \sum_{k'=k-k^0}^{k-1}\sum_{j=1}^m \left|\pi_j^{k'}\right|\end{aligned} \tag{135}$$

Therefore,

$$\left\|x_i(\hat{\mu}^k, s_i^k) - x_i(\mu^k, s_i^k)\right\|_1 \leq n_i\varphi_{L_i}\varphi_{g_i}\sum_{k'=k-k^0}^{k-1}\sum_{j=1}^m \left|\pi_j^{k'}\right| \tag{136}$$

This bound ensures that the difference between $x_i(\hat{\mu}^k, s_i^k)$ and $x_i(\mu^k, s_i^k)$ remains controlled, facilitating stability and convergence analysis of the algorithm.

□

Proof (Lemma 5.5). Define:

$$u_{il}(\mu^k) = \left[x_i(\mu^k, s_i^k)\right]_l \tag{137}$$

Using the definitions (129) and by the mean value theorem for some $\tilde{\mu}_l$ between μ^k and μ^τ:

$$\begin{aligned}\left\|x_i(\mu^k,s_i^k)-x_i(\mu^\tau,s_i^\tau)\right\|_1 &= \sum_l \left|\left[x_i(\mu^k,s_i^k)\right]_l - [x_i(\mu^\tau,s_i^\tau)]_l\right| \\ &= \sum_l \left|u_{il}(\mu^k)-u_{il}(\mu^\tau)\right| \\ &= \sum_l \left|\sum_{j=1}^m \frac{\partial u_{il}}{\partial \mu_j}(\tilde{\mu}_l)\cdot(\mu_j^k-\mu_j^\tau)\right| \\ &= \sum_l \left|\sum_{j=1}^m \left[\frac{\partial x_i(\tilde{\mu}_l,\tilde{s}_i)}{\partial \mu_j}\right]_l \left[\mu_j^k-\mu_j^\tau\right]\right|\end{aligned} \tag{138}$$

Applying (113), then the bounds from Assumption 5, and

(91):

$$\left[\frac{\partial x_i(\tilde{\mu}_l,\tilde{s}_i)}{\partial \mu_j}\right]_l \leq \varphi_{L_i}\varphi_{g_i} \tag{139}$$

Therefore,

$$\begin{aligned}\left\|x_i(\mu^k,s_i^k)-x_i(\mu^\tau,s_i^\tau)\right\|_1 &\leq \sum_l \left|\varphi_{L_i}\varphi_{g_i}\sum_{j=1}^m \left[\mu_j^k-\mu_j^\tau\right]\right| \\ &= n_i\varphi_{L_i}\varphi_{g_i}\sum_{j=1}^m \left|\mu_j^k-\mu_j^\tau\right|\end{aligned} \tag{140}$$

But by expanding the difference, we have:

$$\begin{aligned}\mu_j^k-\mu_j^\tau &= (\mu_j^k-\mu_j^{k-1})+(\mu_j^{k-1}-\mu_j^{k-2})+\cdots+(\mu_j^{\tau+1}-\mu_j^\tau) \\ &= -\pi_j^{k-1}-\pi_j^{k-2}-\cdots-\pi_j^\tau \\ &= -\sum_{k'=\tau}^{k-1}\pi_j^{k'}\end{aligned} \tag{141}$$

Hence,

$$\begin{aligned}\left\|x_i(\mu^k,s_i^k)-x_i(\mu^\tau,s_i^\tau)\right\|_1 &\leq n_i\varphi_{L_i}\varphi_{g_i}\sum_{j=1}^m \left|\sum_{k'=\tau}^{k-1}(-\pi_j^{k'})\right| \\ &\leq n_i\varphi_{L_i}\varphi_{g_i}\sum_{k'=\tau}^{k-1}\sum_{j=1}^m \left|\pi_j^{k'}\right|\end{aligned} \tag{142}$$

□

Lemma 6. *There exists a constant $A_2 > 0$ such that*

$$\|\nabla d_{\rho,s^k}(\mu^k) - \mu^k\|_1 \leq A_2 \sum_{k'=k-2k^0}^{k-1} \|\pi^{k'}\| \tag{143}$$

Proof. Using the definitions of $\nabla d_{\rho,s^k}(\mu^k)$ and μ_j^k from (109) and (78) respectively, then applying the mean-value theorem, we have:

$$\begin{aligned}
\left\|\nabla d_{\rho,s^k}(\mu^k) - \mu^k\right\|_1 &= \sum_{j=1}^{m} \left| \sum_{i=1}^{N} \left[g_{ji}(x_i(\mu^k, s_i^k)) - g_{ji}(\hat{x}_i^k) \right] \right| \\
&\leq \sum_{j=1}^{m} \sum_{i=1}^{N} \left| g_{ji}(x_i(\mu^k, s_i^k)) - g_{ji}(\hat{x}_i^k) \right| \\
&= \sum_{j=1}^{m} \sum_{i=1}^{N} \left\| \nabla g_{ji}(\bar{c}) \right\| \left\| x_i(\mu^k, s_i^k) - \hat{x}_i^k \right\|
\end{aligned} \tag{144}$$

Applying the bounds from Assumption 5:

$$\left\| \nabla g_{ji}(\bar{c}) \right\| \leq \varphi_{g_{ji}} \tag{145}$$

Hence,

$$\begin{aligned}
\left\|\nabla d_{\rho,s^k}(\mu^k) - \mu^k\right\|_1 &\leq \sum_{i=1}^{N} \left[\left\| x_i(\mu^k, s_i^k) - \hat{x}_i^k \right\| \sum_{j=1}^{m} \varphi_{g_{ji}} \right] \\
&= \varphi_g \sum_{i=1}^{N} \left\| x_i(\mu^k, s_i^k) - \hat{x}_i^k \right\| \\
&\leq \varphi_g \sum_{i=1}^{N} \left\| x_i(\mu^k, s_i^k) - \hat{x}_i^k \right\|_1
\end{aligned} \tag{146}$$

Using the definitions of $\hat{x}_i^k$ and $x_i^{k'}$ from (79) and (88), we get:

$$\begin{aligned}
\left\|\nabla d_{\rho,s^k}(\mu^k) - \mu^k\right\|_1 &\leq \varphi_g \sum_{i=1}^{N} \left\| x_i(\mu^k, s_i^k) - \sum_{k'=k-k^0}^{k} a_i(k',k) x_i^{k'} \right\|_1 \\
&\leq \varphi_g \sum_{i=1}^{N} \max_{k-k^0 \leq k' \leq k} \left\| x_i^{k'} - x_i(\mu^k, s_i^k) \right\|_1 \\
&= \varphi_g \sum_{i=1}^{N} \max_{k-k^0 \leq k' \leq k} \left\| x_i(\hat{\mu}^{k'}, s_i^{k'}) - x_i(\mu^k, s_i^k) \right\|_1
\end{aligned} \tag{147}$$

We then use the triangle inequality to separate the terms:

$$\left\|\nabla d_{\rho,s^k}(\mu^k) - \mu^k\right\|_1 \leq \varphi_g \sum_{i=1}^{N} \max_{k-k^0 \leq k' \leq k} \left\{ \left\| x_i(\hat{\mu}^{k'}, s_i^{k'}) - x_i(\mu^{k'}, s_i^{k'}) \right\|_1 + \left\| x_i(\mu^{k'}, s_i^{k'}) - x_i(\mu^k, s_i^k) \right\|_1 \right\} \tag{148}$$

Using Lemmas 5.4 and 5.5:

$$\begin{aligned}\left\|x_i(\hat{\mu}^{k'}, s_i^{k'}) - x_i(\mu^{k'}, s_i^{k'})\right\|_1 &\leq n_i \varphi_{L_i} \varphi_{g_i} \sum_{k''=k'-k^0}^{k'-1} \sum_{j=1}^{m} \left|\pi_j^{k''}\right| \\ \left\|x_i(\mu^{k'}, s_i^{k'}) - x_i(\mu^{k}, s_i^{k})\right\|_1 &\leq n_i \varphi_{L_i} \varphi_{g_i} \sum_{k''=k'}^{k-1} \sum_{j=1}^{m} \left|\pi_j^{k''}\right|\end{aligned} \tag{149}$$

Combining these bounds:

$$\left\|\nabla d_{\rho,s^k}(\mu^k) - \mu^k\right\|_1 \leq \varphi_g \sum_{i=1}^{N} \max_{k-k^0 \leq k' \leq k} \left\{ n_i \varphi_{L_i} \varphi_{g_i} \left(\sum_{k''=k'-k^0}^{k'-1} \sum_{j=1}^{m} \left|\pi_j^{k''}\right| + \sum_{k''=k'}^{k-1} \sum_{j=1}^{m} \left|\pi_j^{k''}\right| \right) \right\} \tag{150}$$

Simplifying further:

$$\begin{aligned}\left\|\nabla d_{\rho,s^k}(\mu^k) - \mu^k\right\|_1 &\leq n\varphi_g^2 \sum_{i=1}^{N} \varphi_{L_i} \max_{k-k^0 \leq k' \leq k} \left\{ \sum_{k''=k'-k^0}^{k-1} \sum_{j=1}^{m} \left|\pi_j^{k''}\right| \right\} \\ &\leq n\varphi_g^2 \sum_{i=1}^{N} \varphi_{L_i} \sum_{k''=k-2k^0}^{k-1} \sum_{j=1}^{m} \left|\pi_j^{k''}\right|\end{aligned} \tag{151}$$

Let $A_2 = n\varphi_g^2 \sum_{i=1}^{N} \varphi_{L_i}$:

$$\left\|\nabla d_{\rho,s^k}(\mu^k) - \mu^k\right\|_1 \leq A_2 \sum_{k''=k-2k^0}^{k-1} \left\|\pi^{k''}\right\|_1 \tag{152}$$

□

Lemma 7. *Provided γ is sufficiently small, we have for all k,*

$$\nabla d_{\rho,s^{k+1}}(\mu^{k+1}) \geq \nabla d_{\rho,s^0}(\mu^0) + \left(\frac{1}{\gamma} - A_1 - 2A_2 k^0\right) \sum_{\tau=0}^{k} \|\pi^\tau\|^2 \tag{153}$$

where A_1 and A_2 are Lemma 5 and 6 constants, respectively. Hence $\|\pi^k\| \longrightarrow 0$ as $k \longrightarrow \infty$

Proof. From the second-order Taylor expansion, we have:

$$\begin{aligned}d_{\rho,s^{k+1}}(\mu^{k+1}) &= d_{\rho,s^k}(\mu^k) + \left[\nabla d_{\rho,s^k}(\mu^k)\right]^\top (\mu^{k+1} - \mu^k) \\ &\quad + \frac{1}{2}(\mu^{k+1} - \mu^k)^\top \nabla^2 d_{\rho,s^k}(\mu^k)(\mu^{k+1} - \mu^k) \\ &= d_{\rho,s^k}(\mu^k) - \left[\nabla d_{\rho,s^k}(\mu^k)\right]^\top \pi^k + \frac{1}{2}\pi^{k^\top} \nabla^2 d_{\rho,s^k}(\mu^k)\pi^k\end{aligned} \tag{154}$$

We will apply Lemma 5.2, then 5.1 to the last equality from (154):

$$\begin{aligned} d_{\rho,s^{k+1}}(\mu^{k+1}) &\geq d_{\rho,s^k}(\mu^k) - \left[\nabla d_{\rho,s^k}(\mu^k)\right]^\top \pi^k - A_1\|\pi^k\|^2 \\ &= d_{\rho,s^k}(\mu^k) - \left[\nabla d_{\rho,s^k}(\mu^k) - \mu^k\right]^\top \pi^k - \mu^{k\top}\pi^k - A_1\|\pi^k\|^2 \\ &\geq d_{\rho,s^k}(\mu^k) - \left[\nabla d_{\rho,s^k}(\mu^k) - \mu^k\right]^\top \pi^k + \frac{1}{\gamma}\|\pi^k\|^2 - A_1\|\pi^k\|^2 \end{aligned} \tag{155}$$

From Lemma 6, the second inequality below holds:

$$\begin{aligned} \left[\nabla d_{\rho,s^k}(\mu^k) - \mu^k\right]^\top \pi^k &\leq \|\nabla d_{\rho,s^k}(\mu^k) - \mu^k\| \cdot \|\pi^k\| \\ &\leq A_2 \sum_{k'=k-2k^0}^{k-1} \|\pi^{k'}\| \cdot \|\pi^k\| \end{aligned} \tag{156}$$

That is,

$$-\left[\nabla d_{\rho,s^k}(\mu^k) - \mu^k\right]^\top \pi^k \geq -A_2 \sum_{k'=k-2k^0}^{k-1} \|\pi^{k'}\| \cdot \|\pi^k\| \tag{157}$$

Substituting (157) into (155)

$$\begin{aligned} d_{\rho,s^{k+1}}(\mu^{k+1}) &\geq d_{\rho,s^k}(\mu^k) - A_2 \sum_{k'=k-2k^0}^{k-1} \|\pi^{k'}\| \cdot \|\pi^k\| - A_1\|\pi^k\|^2 + \frac{1}{\gamma}\|\pi^k\|^2 \\ &= d_{\rho,s^k}(\mu^k) + \left(\frac{1}{\gamma} - A_1\right)\|\pi^k\|^2 - A_2 \sum_{k'=k-2k^0}^{k-1} \|\pi^{k'}\| \cdot \|\pi^k\| \end{aligned} \tag{158}$$

For the last component of (158) from [[27], Eq.(10)] we have:

$$-\sum_{k'=k-2k^0}^{k-1} \|\pi^{k'}\| \cdot \|\pi^k\| \geq \left(\frac{1}{2} - k^0\right)\|\pi^k\|^2 - \frac{1}{2}\sum_{k'=k-2k^0}^{k} \|\pi^{k'}\|^2 \tag{159}$$

Substituting (159) into (158), we have

$$d_{\rho,s^{k+1}}(\mu^{k+1}) \geq d_{\rho,s^k}(\mu^k) + \left[\frac{1}{\gamma} - A_1 + A_2\left(\frac{1}{2} - k^0\right)\right]\|\pi^k\|^2 - \frac{A_2}{2}\sum_{k'=k-2k^0}^{k} \|\pi^{k'}\|^2 \tag{160}$$

Applying the inequality recursively to all $d_\rho(\mu^\tau),\ \tau = k, k-1, \ldots, 1$, taking $\pi^k = 0$ for $k < 0$, we have

$$d_{\rho,s^{k+1}}(\mu^{k+1}) \geq d_{\rho,s^0}(\mu^0) + \left[\frac{1}{\gamma} - A_1 + A_2\left(\frac{1}{2} - k^0\right)\right]\sum_{\tau=0}^{k}\|\pi^\tau\|^2 - \frac{A_2}{2}\sum_{\tau=0}^{k}\sum_{k'=\tau-2k^0}^{\tau}\|\pi^{k'}\|^2 \tag{161}$$

Observe that under the assumption that $\pi^k = 0$ for $k < 0$

$$\begin{aligned}\sum_{\tau=0}^{k}\sum_{k'=\tau-2k^0}^{\tau}\|\pi^{k'}\|^2 &= \sum_{\tau=0}^{k}\sum_{k''=0}^{2k^0}\|\pi^{k''+\tau-2k^0}\|^2 \\ &= \sum_{k''=0}^{2k^0}\sum_{\tau=0}^{k}\|\pi^{\tau+k''-2k^0}\|^2 \\ &\leq \sum_{k''=0}^{2k^0}\sum_{\tau=0}^{k}\|\pi^{\tau}\|^2 \\ &= (2k^0+1)\sum_{\tau=0}^{k}\|\pi^{\tau}\|^2\end{aligned} \tag{162}$$

Multiplying the final result of (162) by -1 we have

$$-\sum_{\tau=0}^{k}\sum_{k'=\tau-2k^0}^{\tau}\|\pi^{k'}\|^2 \geq -(2k^0+1)\sum_{\tau=0}^{k}\|\pi^{\tau}\|^2 \tag{163}$$

Substituting (163) into (161), we have

$$\begin{aligned}d_{\rho,s^{k+1}}(\mu^{k+1}) &\geq d_{\rho,s^0}(\mu^0) + \left[\frac{1}{\gamma} - A_1 + A_2\left(\frac{1}{2}-k^0\right)\right]\sum_{\tau=0}^{k}\|\pi^{\tau}\|^2 - \frac{A_2}{2}(2k^0+1)\sum_{\tau=0}^{k}\|\pi^{\tau}\|^2 \\ &\geq d_{\rho,s^0}(\mu^0) + \left(\frac{1}{\gamma} - A_1 - 2A_2k^0\right)\sum_{\tau=0}^{k}\|\pi^{\tau}\|^2\end{aligned} \tag{164}$$

Since the inequality (164) holds for all $\gamma > 0$, we can choose γ sufficiently small such that

$$\frac{1}{\gamma} - A_1 - 2A_2k^0 > 0 \tag{165}$$

Since $d_{\rho,s^k}(\mu^k)$ is upper bounded (Lemma 2), letting $k \longrightarrow \infty$, we must have $\sum_{k=0}^{\infty}\|\pi^k\|^2 < \infty$, and hence

$$\|\pi^k\| \longrightarrow 0 \; as \; k \longrightarrow \infty \tag{166}$$

□

4.5.2. Proof of Theorem 2

First, let's show that the various errors introduced by asynchronism converge to zero. For all $i \in N$, it follows from (85) that

$$\begin{aligned}\|\hat{\mu}^k - \mu^k\| &\leq \sum_{j=1}^{m} \max_{k-k^0 \leq k' \leq k} |\mu_j^{k'} - \mu_j^k| \\ &\leq \sum_{k'=k-k^0}^{k} \sum_{j=1}^{m} |\pi_j^{k'}| \\ &\leq \sum_{k'=k-k^0}^{k} \|\pi^{k'}\|_1\end{aligned} \tag{167}$$

which by (166) converges to zero as $k \to \infty$. From (88) and (89) and Lemma 5.4 we have

$$\begin{aligned}\|x_i^k - \bar{x}_i^k\|_1 &= \|x_i(\hat{\mu}^k, s_i^k) - \bar{x}_i(\mu^k, s_i^k)\|_1 \\ &\leq n_i \varphi_{L_i} \varphi_{g_i} \sum_{k'=k-k^0}^{k-1} \|\pi^{k'}\|_1\end{aligned} \tag{168}$$

Hence by (166), $|x_i^k - \bar{x}_i^k| \to 0$ for all $i \in N$.

Using (168) and the triangular inequality property:

$$\begin{aligned}|s_i^{k+1} - \bar{s}_i^{k+1}| &= |\zeta s_i^k + (1-\zeta)x_i^{k+1} - [\zeta \bar{s}_i^k + (1-\zeta)\bar{x}_i^{k+1}]| \\ &= |\zeta(s_i^k - \bar{s}_i^k) + (1-\zeta)(x_i^{k+1} - \bar{x}_i^{k+1})| \\ &\leq \zeta |s_i^k - \bar{s}_i^k| + (1-\zeta)|x_i^{k+1} - \bar{x}_i^{k+1}| \\ &\leq \zeta |s_i^k - \bar{s}_i^k| + \varphi_{L_i}\varphi_{g_i}(1-\zeta) \sum_{k'=k-k^0}^{k-1} |\pi^{k'}| \\ &= \zeta |s_i^k - \bar{s}_i^k| + \varepsilon_k \\ &\leq \zeta \left(\zeta |s_i^{k-1} - \bar{s}_i^{k-1}| + \varepsilon_{k-1} \right) + \varepsilon_k \\ &= \zeta^2 |s_i^{k-1} - \bar{s}_i^{k-1}| + \zeta \varepsilon_{k-1} + \varepsilon_k \\ &\leq \zeta^{k+1-k_0} |s_i^{k_0} - \bar{s}_i^{k_0}| + \sum_{j=k_0}^{k} \zeta^{k-j} \varepsilon_j\end{aligned} \tag{169}$$

Where $\varepsilon_k = \varphi_{L_i}\varphi_{g_i}(1-\zeta)\sum_{k'=k-k^0}^{k-1} |\pi^{k'}|$. Since $|\pi^k| \to 0$ as $k \to \infty$, $\varepsilon_k \to 0$ as well. Given that $\zeta \in [0,1)$, we know that $\zeta^n \to 0$ as $n \to \infty$. Thus, the term $\zeta^{k+1-k_0}|s_i^{k_0} - \bar{s}_i^{k_0}|$ will diminish over time. From Lemma 6 and (166), the error $\|\nabla d_\rho(\mu^k) - \mu^k\|$ in gradient estimation converges to zero.

Now, it will be shown that every accumulation point of the sequence $\{\mu^k\}$ generated by Algorithm 2 maximizes the dual problem. Let μ^* be an accumulation point of $\{\mu^k\}$. At least one exists since the level set $\{\mu \geq 0 | d_{\rho,s}(\mu) \geq d_{\rho,s^0}(\mu^0)\}$ of $d_{\rho,s}$ is compact and the sequence $\{d_{\rho,s^k}(\mu^k)\}$ is in the level set provided γ is sufficiently small [see (164)]. Moreover, since the interval between consecutive updates is bounded (Assumption 7), μ^* is an accumulation

point of $\{\mu^k, k \in \cap_j T_j\}$. Let $\{k_t\} \subseteq \cap_j T_j$ be a sequence such that $\{\mu(k_t)\}$ converges to μ^*. Since $\nabla d_{\rho,s}$ is continuous (Lemma 2) and $\|\mu^k - \nabla d_{\rho,s^k}(\mu^k)\| \longrightarrow 0$ (Lemma 6), we have

$$\lim_t \mu^{k_t} = \lim_t d_{\rho,s^{k_t}}(\mu^{k_t}) = d_{\rho,s^*}(\mu^*) \tag{170}$$

Hence

$$\begin{aligned} \mu^* - \left[\mu^* + \gamma d_{\rho,s^*}(\mu^*)\right]^+ &= \mu^{k_t} - \lim_t \left[\mu^{k_t} + \gamma d_{\rho,s^{k_t}}(\mu^{k_t})\right]^+ \\ &= \lim_t \pi^{k_t} = 0 \end{aligned} \tag{171}$$

where the last equality follows from (166). Then, the projection theorem [4, Proposition 2.1.3]) implies

$$\gamma \left[d_{\rho,s^*}(\mu^*)\right]^\top (\mu - \mu^*) \leq 0, \quad \forall\, \mu \geq 0 \tag{172}$$

which, due to the concavity of $d_{\rho,s}$ implies that μ^* maximizes $d_{\rho,s}$ over $\mu \geq 0$. By duality $x^* = x(\mu^*, s^*)$ and s^* are the unique primal optimal rates. It will be shown that they are limit points of $\{x^k\}$ and $\{s^{k+1}\}$ generated by Algorithm 2. Consider the subsequence $\{x^{k_t}\}$ and $\{s^{k_t}\}$. Since they are in the compact set X_i, there exists a subsequence $\{k_n\} \subseteq \{k_t\} \cap \cap_i S_i$, such that $x_i^{k_n}$ and $s_i^{k_n}$ converge. Since $\|x_i^k - \bar{x}_i^k\| \to 0$ and $\|s_i^{k+1} - \bar{s}_i^{k+1}\| \to 0$, we have

$$\lim_n x_i^{k_n} = \lim_n \bar{x}_i^{k_n} = \lim_n x_i(\mu^{k_n}, s^{k_n}) = x_i(\mu^*, s^*) \tag{173}$$

$$\begin{aligned} \lim_n s^{k_n+1} &= \lim_n \bar{s}^{k_n+1} \\ &= \zeta \lim_n s_i^{k_n} + (1-\zeta) \lim_n x_i^{k_n+1} \\ &= \zeta s_i^* + (1-\zeta) x_i^* \\ &= s^* \end{aligned} \tag{174}$$

This proves that the errors in price estimation, gradient estimation, and the primal variable calculation all converge to zero as $k \to \infty$.

Therefore, the proof of Theorem 2 is now complete.

5. Solution of the Simultaneous Routing and Bandwidth Allocation Problem in Energy-Aware Networks Using Augmented Lagrangian Based Algorithms and Decomposition

In 2022, data transmission networks consumed 260-360 TWh or 1-1.5% of the global electricity use [28]. A strong growth in demand for data centre and network services is expected to continue at least until the end of this decade [29], especially because of video streaming and gaming. On the other hand, a global energy crisis causes an increase in prices of electricity and energy commodities, and the threat of their shortages. All this makes it more important than ever to use the network infrastructure efficiently to guarantee the highest possible quality of service with the lowest possible energy consumption.

In the standard approach, network congestion control along with Active queue management algorithms try to minimize an aggregated cost with respect to source rates, assuming that routing is given and constant over the time scale of interest. However, it seems that it would be more effective if transport and network layers were treated together and minimized cross-layer costs on the time scale of routing changes, taking into account the energy component.

In the paper [30], an approach to solve such a simultaneous routing and flow rate optimization problem in energy-aware computer networks has been presented. The formulation considered two criteria. The first criterion was the cost of a bad quality of service, and the second was the energy consumption. It was earlier shown [31] that such problems, where routing variables are binary, even in the simplified version, without energy component, are NP-hard, which means that for bigger networks it is impossible to get the optimal solution in a reasonable time. The ordinary Lagrangian relaxation cannot help to get a good assessment of the solution and, as it is in most of the integer linear programming (ILP) and mixed-integer nonlinear programming (MINLP) problems [32], [33], duality gap can be observed [34], that is a nonzero difference between the minimum of the original problem and the maximum of the dual function and the violation of many constraints.

In the case of non-convex problems, the most popular remedy to obtain strong duality, that is, the zero gap, and to ensure that the constraints are met, is to use augmented Lagrangian [35]–[41], and the multiplier or, in other words, shifted penalty function method, based on it [23], [42], [43]. Since the focus is on a network MINLP problem, only formulations involving non-convex constraints or discrete sets are of interest. As it is stated in [40], continuous optimization over concave constraints can be used to represent the integrality condition. The simplest way of conversion of a discrete set $\tilde{X}$ to an equality constraint with a continuous function for any $x \in X \subseteq \mathbb{R}^n$, where $\tilde{X} \subseteq X$ and X compact, is to use a

distance function $d(x,\tilde{X}) = \min_{z\in\tilde{X}} \|x-z\|, \forall x \in X$ and the constraint equation $d(x,\tilde{X}) = 0$ [40]. Other methods of continuous reformulations of mixed-integer optimization problems that may be effective in some problems are presented in [44].

Unfortunately, although MILP and MINLP optimization problems have been present from the very beginning in the most important works on the augmented Lagrangian approach [23] and they are very important in practical problems [45]–[48], most of the works dealing with non-convex problems consider only non-convexity of the objective functions [49], [50]. There are relatively few proposals of methods and algorithms considering also non-convexity of constraints [11], [15]–[17], [20], [47], and even in a recent paper, the following statement can be found: "Despite the recent development of distributed algorithms for non-convex programs, highly complicated constraints still pose a significant challenge in theory and practice" [51]. The proposed algorithms are rather complicated, involve many levels and loops, corrections, linearizations, etc. [47], [51]–[53].

This chapter addresses the two-layer network optimization problem using a specially adapted classical method proposed in the literature for problems with non-convex constraints, which also offers potential for parallelization. Because of that, very intensively developed in recent years are methods from the ADMM family (Alternative Directions Method of Multipliers), which are of a Gauss-Seidel type, are sequential, and as such, are treated on an equal footing with others. When necessary, such as when an algorithm is originally formulated for a different type of constraints, the method is adapted to suit the specific requirements of the problem.

5.1. Network Optimization Problem of Simultaneous Routing and Bandwidth Allocation in Energy-Aware Networks

The problem of optimizing routing and flow rates simultaneously is described as identifying flow rates and routes (single paths) that satisfy network constraints for all possible source-destination pairs at the minimal cost. The variables and parameters used in the formulation are:

$N, i \in N$	-	set of all network nodes and a single node, respectively,
$A, (i,j) \in A$	-	set of all network arcs and a single arc, respectively,
$L, l \in L$	-	set of all labeled links and a single labeled link, respectively,
$e : A \to L$	-	one-to-one mapping from arcs to links labeled by a single natural number,
$W, w \in W$	-	set of all demands and a single demand, respectively,
$S(w), D(w)$	-	source and destination node for the specific demand w, respectively,
x_w	-	flow rate for the specific demand $w, x_w \in \mathbb{R}_+ \cup \{0\}$,
$\underline{x}_w, \bar{x}_w$	-	lower and upper bound on the flow rate for the demand w, it is assumed that $0 < \underline{x}_w < \bar{x}_w$,
c_l	-	capacity of the link l, $c_l > 0$,
b_{wl}	-	binary routing decision variable, whether the link l is used by the demand w,
γ	-	positive parameter – the weight of the QoS part of the objective function,
δ	-	positive parameter – the weight of the energy usage part of the objective function.

With this notation, the problem can be formulated in the following way [34]:

$$\min_{x,b,y} \quad \sum_{w \in W} \left[f_w(x_w, b_w) = \gamma \left(\bar{x}_w - x_w \right)^2 + \delta \sum_{l \in L} b_{wl} \right] \tag{175}$$

subject to (s.t.)

$$\sum_{\{i \in N | (i,j) \in A\}} y_{w,e(i,j)} - \sum_{\{k \in N | (j,k) \in A\}} y_{w,e(j,k)} = \begin{cases} -x_w & j = S(w) \\ x_w & j = D(w) \\ 0 & \text{otherwise} \end{cases} \quad \forall w \in W, \forall j \in N \tag{176}$$

$$\underline{x}_w \leq x_w \leq \bar{x}_w, \qquad \forall w \in W \tag{177}$$

$$\sum_{w \in W} y_{wl} \leq c_l, \qquad \forall l \in L \tag{178}$$

$$y_{wl} \geq 0, \qquad \forall w \in W, \forall l \in L \tag{179}$$

$$y_{wl} \leq b_{wl} \bar{x}_w, \qquad \forall w \in W, \forall l \in L \tag{180}$$

$$\sum_{\{j \in N | (i,j) \in A\}} b_{w,e(i,j)} \leq 1, \qquad \forall w \in W, \forall i \in N \tag{181}$$

$$b_{wl} \in \{0,1\}, \qquad \forall w \in W, \forall l \in L \tag{182}$$

where the vector of binary variables $b_w = \{b_{wl}, l = 1,2,\ldots,|L|\}$ defines a single path for the demand $w \in W$.

This is the third formulation from the paper [34], denoted there by P_{alt}. Objective function (175) is quadratic and convex. The first component of the function $f_w(x_w, b_w)$ expresses a cost for not delivering the full possible bandwidth for a connection $w \in W$. The second component expresses the total cost of energy used in the network by the connection $w \in W$. Flow conservation law equations are formulated with auxiliary real variables y_{wl} and binary variables b_{wl}, the equality (176), and three inequalities (181), (179), (180). Constraints (181) force single path routing, constraints (180) keep relation between auxiliary variables and binary variables.

5.2. Decomposition of the network problem

Unfortunately, not all the methods presented in Section 5 can be applied to solve our network problem (175)-(182). Tanikawa-Mukai and Tatjewski-Engelmann methods from Section 3.5 are not suitable, because this is a mixed-integer problem, where Karush-Kuhn-Tucker optimality conditions, which are the basis of the explicit formulas for calculation of the Lagrange multipliers, are not well defined. Fortunately, all the remaining methods can be applied.

We will try to solve the problem (175)-(182) decomposing it with respect to flows $w \in W$. Hence, it will be convenient to introduce for every flow w the admissible set XBY_w resulting from the equations (176), (177), (179)-(182). This set for the flow w will be defined as

$$XBY_w = \left\{ \begin{array}{l} x_w, y_{wl} \in \mathbb{R}, b_{wl} \in \{0,1\}, l \in L: \\ \sum_{\{i \in N | (i,j) \in A\}} y_{w,e(i,j)} - \sum_{\{k \in N | (j,k) \in A\}} y_{w,e(j,k)} \\ \qquad = \begin{cases} -x_w & j = S(w) \\ x_w & j = D(w) \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in N, \\ \underline{x}_w \le x_w \le \bar{x}_w, \\ y_{wl} \ge 0, \quad \forall l \in L, \\ y_{wl} \le b_{wl} \bar{x}_w, \quad \forall l \in L \\ \sum_{\{j \in N | (i,j) \in A\}} b_{w,e(i,j)} \le 1, \quad \forall i \in N \end{array} \right\} \tag{183}$$

5.2.1. The Standard Multiplier Method Without Decomposition

The Augmented Lagrangian for (175)-(182) is

$$\begin{aligned} L_\rho(x,b,y,\lambda,z) = \sum_{w \in W} f_w(x_w,b_w) + \sum_{l \in L} \lambda_l \Big(\sum_{w \in W} y_{wl} - c_l + z_l \Big) \\ + \frac{\rho}{2} \sum_{l \in L} \Big(\sum_{w \in W} y_{wl} - c_l + z_l \Big)^2 \end{aligned} \tag{184}$$

where $z_l \ge 0, l \in L$ are slack variables. To find for every $l \in L$ the value of z_l^* that minimizes the Augmented Lagrangian (184) in current conditions (that is, for given $y_{wl}, w \in W, l \in L$) we write

$$\min_{z_l \ge 0} \left[\lambda_l z_l + \frac{\rho}{2} \Big(\sum_{w \in W} y_{wl} - c_l + z_l \Big)^2 \right] \tag{185}$$

The unconstrained minimum of the expression in square brackets in (185) is the scalar $\hat{z}_l$ at which the derivative is 0

$$\lambda_l + \rho \Big(\sum_{w \in W} y_{wl} - c_l + \hat{z}_l \Big) = 0 \tag{186}$$

that is,

$$\hat{z}_l = -\frac{\lambda_l}{\rho} - \sum_{w \in W} y_{wl} + c_l \tag{187}$$

Hence the solution of problem (185) is

$$z_l^* = \max \Big(0, \; -\frac{\lambda_l}{\rho} - \sum_{w \in W} y_{wl} + c_l \Big), \qquad \forall l \in L \tag{188}$$

and

$$L_\rho(x,b,y,\lambda,z^*) \triangleq L_\rho(x,b,y,\lambda) = \sum_{w\in W} f_w(x_w,b_w) + \sum_{l\in L}\lambda_l \max\Big(-\frac{\lambda_l}{\rho}, \sum_{w\in W} y_{wl} - c_l\Big) + \frac{\rho}{2}\sum_{l\in L}\Big[\max\Big(-\frac{\lambda_l}{\rho}, \sum_{w\in W} y_{wl} - c_l\Big)\Big]^2 \tag{189}$$

It is known that at iteration $k+1$,

$$\lambda_l^{k+1} = \lambda_l^k + \rho_k\Big(\sum_{w\in W} y_{wl}^{k+1} - c_l + z_l^k\Big), \qquad \forall l \in L \tag{190}$$

Substituting

$$z_l^k = \max\Big(0,\ -\frac{\lambda_l}{\rho} - \sum_{w\in W} y_{wl}^{k+1} + c_l\Big), \qquad \forall l \in L \tag{191}$$

into (190) we have

$$\lambda_l^{k+1} = \lambda_l^k + \rho_k \max\Big(-\frac{\lambda_l^k}{\rho}, \sum_{w\in W} y_{wl}^{k+1} - c_l\Big), \qquad \forall l \in L \tag{192}$$

Taking into account (189), we can solve (175)-(182) using the algorithm (17) - (18), which will take the form

$$\begin{aligned} (x^{k+1},b^{k+1},y^{k+1}) &= \arg\min_{x,b,y} L_\rho(x,b,y,\lambda^k) \\ s.t &\quad (176),(177),(179)-(182) \qquad (193)\\ \lambda_l^{k+1} &= \lambda_l^k + \rho_k \max\Big(-\tfrac{\lambda_l^k}{\rho}, \textstyle\sum_{w\in W} y_{wl}^{k+1} - c_l\Big), \qquad \forall l \in L \qquad (194) \end{aligned}$$

with $\rho > 0$.

5.2.2. The Bertsekas Method

The augmented Lagrangian in Bertsekas method for the problem (175)-(182) will have the form

$$L_\rho(x,b,y,z,x^s,b^s,y^s,z^s,\lambda) = \sum_{w\in W} f_w(x_w,b_w) + \sum_{l\in L}\lambda_l\left(\sum_{w\in W} y_{wl} - c_l + z_l\right) + \frac{\rho}{2}\left\{\sum_{w\in W}(x_w - x_w^s)^2 + \sum_{w\in W}\sum_{l\in L}\left[(b_{wl}-b_{wl}^s)^2 + (y_{wl}-y_{wl}^s)^2\right] + \sum_{l\in L}(z_l - z_l^s)^2\right\} \tag{195}$$

where $z_l \geq 0$ are slack variables and $\lambda_l \in \mathbb{R}$ for $l \in L$. Formula (195) is equivalent to

$$\begin{aligned} L_\rho(x,b,y,z,x^s,b^s,y^s,z^s,\lambda) = \sum_{w\in W} \Bigg\{ & f_w(x_w,b_w) + \sum_{l\in L} \lambda_l y_{wl} \\ & + \frac{\rho}{2}\left[(x_w - x_w^s)^2 + \sum_{l\in L} \left((b_{wl} - b_{wl}^s)^2 + (y_{wl} - y_{wl}^s)^2 \right) \right] \Bigg\} \\ & + \sum_{l\in L} \left\{ \lambda_l (z_l - c_l) + \frac{\rho}{2} (z_l - z_l^s)^2 \right\} \\ & \triangleq \sum_{w\in W} L_{\rho 1_w}(x_w,b_w,y_w,x_w^s,b_w^s,y_w^s,\lambda) + \sum_{l\in L} L_{\rho 2_l}(z_l,z_l^s,\lambda_l) \end{aligned} \tag{196}$$

where

$$\begin{aligned} L_{\rho 1_w}(x_w,b_w,y_w,x_w^s,b_w^s,y_w^s,\lambda) = & f_w(x_w,b_w) + \sum_{l\in L} \lambda_l y_{wl} \\ & + \frac{\rho}{2}\left[(x_w - x_w^s)^2 + \sum_{l\in L} \left((b_{wl} - b_{wl}^s)^2 + (y_{wl} - y_{wl}^s)^2 \right) \right] \end{aligned} \tag{197}$$

and

$$L_{\rho 2_l}(z_l,z_l^s,\lambda_l) = \lambda_l (z_l - c_l) + \frac{\rho}{2} (z_l - z_l^s)^2 \tag{198}$$

To find for every $l \in L$ the value of z_l^* that minimizes the Augmented Lagrangian (195) we will solve independently the problems:

$$\min_{z_l \geq 0} \left[L_{\rho 2_l}(z_l,z_l^s,\lambda_l) = \lambda_l (z_l - c_l) + \frac{\rho}{2} (z_l - z_l^s)^2 \right] \tag{199}$$

The unconstrained minimum of the expression in square brackets in (199) is the scalar $\hat{z}_l$ at which the derivative is 0, that is

$$\lambda_l + \rho\left(\hat{z}_l - z_l^s\right) = 0 \tag{200}$$

Hence

$$\hat{z}_l = z_l^s - \frac{\lambda_l}{\rho} \tag{201}$$

and the solution of problem (199) is

$$z_l^* = \max\left(0,\ z_l^s - \frac{\lambda_l}{\rho}\right), \qquad \forall l \in L \tag{202}$$

Now, subsequent iterations of the algorithm (22)-(24) can be performed.

5.2.3. The Tatjewski Method

The Tatjewski Lagrangian of (175)-(182) in a decomposed form can be written as

$$\begin{aligned} L_\rho(x,b,y,z,y^s,z^s,\lambda) = \sum_{w\in W} f_w(x_w,b_w) + \sum_{l\in L}\lambda_l\Big(\sum_{w\in W} y_{wl} - c_l + z_l\Big) \\ + \frac{\rho}{2}\sum_{l\in L}\sum_{w\in W}\Big(y_{wl} + \sum_{q\in W, q\neq w} y^s_{ql} + z^s_l - c_l\Big)^2 \\ + \frac{\rho}{2}\sum_{l\in L}\Big(z_l - c_l + \sum_{w\in W} y^s_{wl}\Big)^2 \end{aligned} \tag{203}$$

where $z_l \geq 0, l \in L$ are slack variables. Formula (203) may be rewritten in the following way:

$$\begin{aligned} L_\rho(x,b,y,z,y^s,z^s,\lambda) &= \sum_{w\in W}\left[f_w(x_w,b_w) + \sum_{l\in L}\lambda_l y_{wl} + \frac{\rho}{2}\sum_{l\in L}\Big(y_{wl} + \sum_{q\in W,q\neq w} y^s_{ql} + z^s_l - c_l\Big)^2\right] \\ &+ \sum_{l\in L}\left[\lambda_l\Big(z_l - c_l\Big) + \frac{\rho}{2}\Big(z_l - c_l + \sum_{w\in W} y^s_{wl}\Big)^2\right] \\ &\triangleq \sum_{w\in W} L_{\rho 1_w}(x_w,b_w,y_w,y^s,z^s,\lambda) + \sum_{l\in L} L_{\rho 2_l}(z_l,y^s_l,\lambda_l) \end{aligned} \tag{204}$$

where

$$L_{\rho 1_w}(x_w,b_w,y_w,y^s,z^s,\lambda) = f_w(x_w,b_w) + \sum_{l\in L}\lambda_l y_{wl} + \frac{\rho}{2}\sum_{l\in L}\Big(y_{wl} + \sum_{q\in W,q\neq w} y^s_{ql} + z^s_l - c_l\Big)^2 \tag{205}$$

and

$$L_{\rho 2_l}(z_l,y^s_l,\lambda_l) = \lambda_l\Big(z_l - c_l\Big) + \frac{\rho}{2}\Big(z_l - c_l + \sum_{w\in W} y^s_{wl}\Big)^2 \tag{206}$$

To find for every $l \in L$ the value of z^*_l that minimizes the Augmented Lagrangian (203), the problems will be solved independently

$$\min_{z_l\geq 0}\left[L_{\rho 2_l}(z_l,y^s_l,\lambda) = \lambda_l\Big(z_l - c_l\Big) + \frac{\rho}{2}\Big(z_l - c_l + \sum_{w\in W} y^s_{wl}\Big)^2\right] \tag{207}$$

The unconstrained minimum of the expression enclosed in square brackets in (207) is for the scalar $\hat{z}_{wl}$ at which the derivative is 0, that is

$$\lambda_l + \rho\Big(z_l - c_l + \sum_{w\in W} y^s_{wl}\Big) = 0 \tag{208}$$

Hence

$$\hat{z}_l = -\frac{\lambda_l}{\rho} + c_l - \sum_{w \in W} y^s_{wl}, \qquad \forall l \in L \tag{209}$$

The solution of problem (207) is

$$z^*_l = \max\left[0, -\frac{\lambda_l}{\rho} + c_l - \sum_{w \in W} y^s_{wl}\right], \qquad \forall l \in L \tag{210}$$

Now the iterations of Tatjewski algorithm (38)-(40) can be executed.

5.2.4. SALA ADMM algorithm

In the network problem (175)-(182) constraints binding flows (178) are of the inequality type, hence in our version of problem formulation, to be solved by SALA algorithm (41)-(44) slack variables $z_{wl} \geq 0$ are introduced, such that

$$\sum_{w \in W} y_{wl} - c_l + \sum_{w \in W} z_{wl} = \sum_{w \in W} s_{wl}, \quad \forall l \in L \tag{211}$$

where $\forall l \in L$

$$\sum_{w \in W} s_{wl} = 0 \tag{212}$$

The equation (211) may be written as:

$$\sum_{w \in W} \left(y_{wl} - \frac{c_l}{|W|} + z_{wl} - s_{wl} \right) = 0 \tag{213}$$

The equality constraint for the flow $w \in W$ in link $l \in L$ will be as follows:

$$y_{wl} - \frac{c_l}{|W|} + z_{wl} - s_{wl} = 0 \tag{214}$$

The Augmented Lagrangian is

$$
\begin{aligned}
L_\rho(x,b,y,z,s,\lambda) = \sum_{w\in W} \Bigg[& f_w(x_w,b_w) \\
& + \sum_{l\in L} \lambda_l \left(y_{wl} - \frac{c_l}{|W|} + z_{wl} - s_{wl} \right) + \frac{\rho}{2} \sum_{l\in L} \left(y_{wl} - \frac{c_l}{|W|} + z_{wl} - s_{wl} \right)^2 \Bigg] \\
& = \sum_{w\in W} \left[f_w(x_w,b_w) + L_{pen,\rho_w}(y_w,s_w,z,\lambda) \right] \\
& \triangleq \sum_{w\in W} L_{\rho_w}(x_w,b_w,y_w,s_w,z,\lambda)
\end{aligned} \tag{215}
$$

where

$$
L_{pen,\rho_w}(y_w,s_w,z,\lambda) = \sum_{l\in L} \left[\lambda_l \left(y_{wl} - \frac{c_l}{|W|} + z_{wl} - s_{wl} \right) + \frac{\rho}{2} \left(y_{wl} - \frac{c_l}{|W|} + z_{wl} - s_{wl} \right)^2 \right] \tag{216}
$$

The algorithm (50) -(54) adapted to this problem will take the form

$$
(x_w^{k+1}, b_w^{k+1}, y_w^{k+1}, z_w^{k+1}) = \arg \min_{\substack{(x_w,b_w,y_w)\in XBY_w \\ z\geq 0}} L_{\rho_w}(x_w,b_w,y_w,z,s_w^k,\lambda^k),\ w\in W \tag{217}
$$

$$
\text{Calculate } r_l^{k+1} = \sum_{w\in W} \left(y_{wl}^{k+1} + z_{wl}^{k+1} \right) - c_l, \quad l\in L \tag{218}
$$

$$
s^{k+1} = y_{wl} - \frac{c_l}{|W|} + z_{wl} - \frac{r_l}{|W|} \tag{219}
$$

$$
\lambda_l^{k+1} = \lambda_l^k + \frac{\rho_k}{|W|} r_l^{k+1},\ w\in W, l\in L \tag{220}
$$

$$
\rho_{k+1} = \alpha\,\rho_k, \quad \alpha \geq 1 \tag{221}
$$

5.3. Numerical Tests

5.3.1. Dataset Description

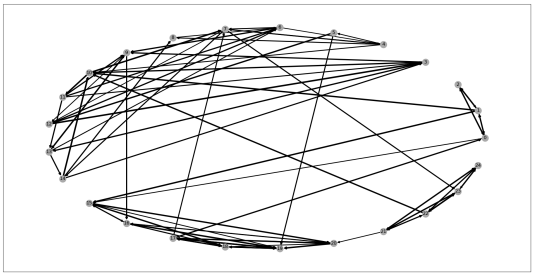
The algorithms described above were implemented and tested on ten network problems of different sizes. The tested networks consisted of loosely connected node clusters with strongly connected nodes. The problems were first solved without decomposition and later with decomposition, allowing for future parallelization (however, our tests were performed in a sequential way).

Five distinct network topologies were created for the study. The first two topologies, the

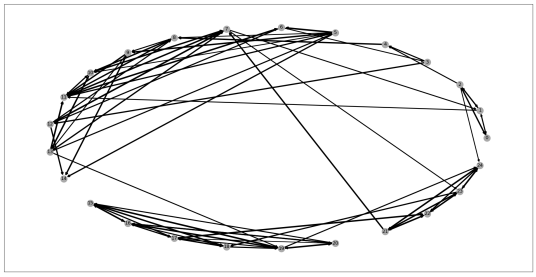
medium network, had fewer variables and can be observed in Table 5.1 and Fig. 5.1. The second two topologies, the large network, had more variables and can be seen in Table 5.1 and Fig. 5.2. The last topology, the extra-large network, had many more variables and can be seen in Table 5.1 and Fig. 5.3. The source and destination nodes for each problem instance were generated randomly, along with the capacities of links. Instances of the problem were created for all network topologies.

Table 5.1. Problem instances

	Nodes Number	Arcs Number	Demands Number	Flow Rates Bounds	Capacities Bound
Medium Problem	25	84	12	[0.001,3]	[0.3,1]
Large Problem	49	143	32	[0.001,3]	[0.3,1]
Extra Large Problem	77	227	64	[0.001,3]	[0.3,1]

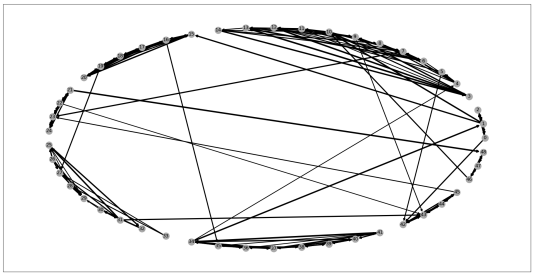


(a) For $\gamma = 1,2;\ \delta = 1$

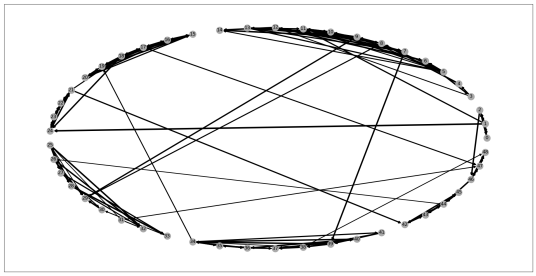


(b) For $\gamma = 1,2;\ \delta = 2$

Figure 5.1. Medium Network Topology



(a) For $\gamma = 1,2;\ \delta = 1$



(b) For $\gamma = 1,2;\ \delta = 2$

Figure 5.2. Large Network Topology

5.3.2. Implementation Details

The Python 3.7.7 programming language was used for all implementations and numerical experiments. To construct and evaluate optimization problems, the Pyomo software package was employed, the Gurobi solver was used for the optimization, and the NetworkX software package [54] was applied to create and display networks related to optimization problems.

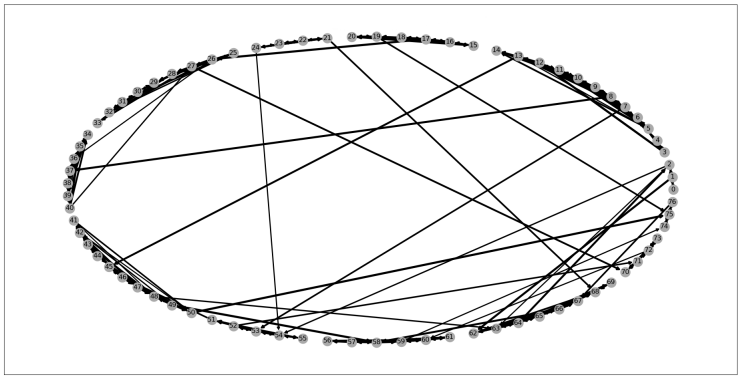


Figure 5.3. Extra-Large Network Topology for $\gamma = 1,2;\ \delta = 2$

The tests were performed on the machine with AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz processor, 32 GB RAM, 64-bit operating system, x64-based processor, 512 GB SSD storage, under Windows 10 Pro operating system.

5.3.3. Results and Discussion

Figures 5.4–5.13 show plots of values of the objective function (175) and the norm of constraints residuals in subsequent iterations of all optimization algorithms considered in Section 5.2 for large problems.

In most cases all algorithms based on the Augmented Lagrangian method dealt very well with duality gap (the biggest value of the gap was less than 5%; in most cases it was below 0.5%), which is a serious issue when Lagrangian Relaxation is applied to MINLP problems, such as simultaneous optimization of routing and bandwidth allocation [30], [31], [34].

Among the algorithms tested, the standard Augmented Lagrangian method achieved the objective value with the least relative error for medium and large-sized problems. In most tests Tatjewski and ADMM SALA algorithms also produced competitive results, though slightly higher, Bertsekas algorithm was a little worse but not always. In the case of very large problems the Tatjewski algorithm proved to be the most precise, ADMM SALA and Bertsekas algorithms were a little worse, but they all delivered approximation of solutions with a very small relative error less than 0.6%. The ordinary Augmented Lagrangian algorithm did not converge within 8 hours.

Regarding constraints violation, the algorithms are comparable - they all deliver (of course if they converge) admissible solution with the assumed accuracy 10^{-6}. As expected, none of the algorithms incurred any routing violations, indicating their ability to generate feasible routing solutions that satisfy network flow conservation equations.

The runtime analysis in Tables 5.2–5.4 reveals that while the medium problems are solved

very fast by one of the best commercial solvers (Gurobi), in the case of large and very large problems AL based decomposed algorithms are competitive. The run times of delivering of a good approximation of the optimal solution were 2–3 times shorter for the standard AL algorithm and 4–10 (for one test example even 30) times for the algorithms with decomposition. The standard Augmented Lagrangian method required significantly more time to solve both medium and large-sized problems and failed in the case of very large problems.

The results highlight the trade-offs between various Lagrangian-based optimization algorithms. Standard Augmented Lagrangian excels in finding optimal routing solutions in not too big problems, but at the cost of increased computational time. Bertsekas algorithm offers a balanced approach, achieving competitive results with shorter runtimes. Tatjewski's algorithm maintains high accuracy of solutions for big problems, making it suitable for scenarios with stringent demands. ADMM SALA is a little worse in terms of the objective function, but relatively fast.

It should be mentioned, that the above AL based methods will work also for more complicated two-layer network problems, e.g., with logarithmic objective functions [30].

Table 5.2. Medium Problems: Objective values, relative errors (in %) with respect to the centralized ("Global") Gurobi solution (in brackets) for the tested AL algorithms with, respectively: $\gamma = 1$ and $\delta = 1$, $\gamma = 2$ and $\delta = 1$, $\gamma = 1$ and $\delta = 2$, $\gamma = 2$ and $\delta = 2$.

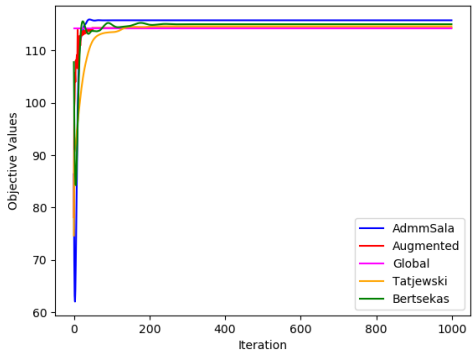
	Objective	Time(secs)	Status
Global	114.31 (0.0)	0	Optimal
Augmented	114.31 (0.0)	504	Optimal
Bertsekas	115.03 (0.63)	149	Optimal
Tatjewski	114.64 (0.29)	171	Optimal
AdmmSala	115.78 (1.29)	87	Optimal
Global	191.61 (0.0)	1	Optimal
Augmented	191.61 (0.0)	976	Optimal
Bertsekas	195.45 (2.0)	162	Optimal
Tatjewski	191.63 (0.01)	368	Optimal
AdmmSala	196.55 (2.58)	85	Optimal
Global	163.26 (0.0)	0	Optimal
Augmented	163.26 (0.0)	93	Optimal
Bertsekas	163.42 (0.1)	44	Optimal
Tatjewski	170.75 (4.59)	216	Optimal
AdmmSala	163.26 (0.0)	84	Optimal
Global	246.52 (0.0)	2	Optimal
Augmented	246.52 (0.0)	137	Optimal
Bertsekas	247.74 (0.49)	95	Optimal
Tatjewski	246.5 (0.01)	222	Optimal
AdmmSala	246.55 (0.01)	85	Optimal

Table 5.3. Large Problems: Objective values, relative errors (in %) with respect to the centralized ("Global") Gurobi solution (in brackets) for the tested AL algorithms with, respectively: $\gamma = 1$ and $\delta = 1$, $\gamma = 2$ and $\delta = 1$, $\gamma = 1$ and $\delta = 2$, $\gamma = 2$ and $\delta = 2$.

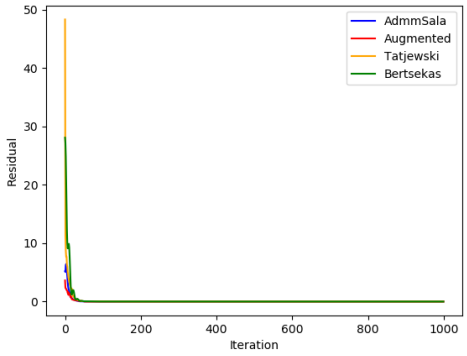
	Objective	Time(secs)	Status
Global	384.96 (0.0)	4005	Timeout
Augmented	384.96 (0.0)	1723	Optimal
Bertsekas	385.09 (0.03)	379	Optimal
Tatjewski	384.98 (0.0)	800	Optimal
AdmmSala	385.32 (0.09)	383	Optimal
Global	632.92 (0.0)	4003	Timeout
Augmented	632.92 (0.0)	2026	Optimal
Bertsekas	635.18 (0.36)	369	Optimal
Tatjewski	632.95 (0.0)	920	Optimal
AdmmSala	633.67 (0.12)	396	Optimal
Global	570.92 (0.0)	4007	Timeout
Augmented	570.92 (0.0)	1482	Optimal
Bertsekas	571.49 (0.1)	379	Optimal
Tatjewski	570.91 (0.0)	718	Optimal
AdmmSala	571.06 (0.03)	395	Optimal
Global	819.84 (0.0)	4006	Timeout
Augmented	819.84 (0.0)	1965	Optimal
Bertsekas	820.57 (0.09)	127	Optimal
Tatjewski	820.36 (0.06)	280	Optimal
AdmmSala	820.16 (0.04)	416	Optimal

Table 5.4. Extra-Large Problems: Objective values, relative errors (in %) with respect to the centralized ("Global") Gurobi solution (in brackets) for the tested AL algorithms with, respectively: $\gamma = 1$ and $\delta = 2$, $\gamma = 2$ and $\delta = 2$.

	Objective	Time(secs)	Status
Global	1123.49 (0.0)	6013	Timeout
Augmented	1153.63 (2.68)	30938	MaxIter
Bertsekas	1124.75 (0.11)	796	Optimal
Tatjewski	1124.0 (0.05)	3692	Optimal
AdmmSala	1125.1 (0.14)	1268	Optimal
Global	1627.31 (0.0)	30042	Timeout
Augmented	1682.76 (3.41)	31040	MaxIter
Bertsekas	1635.12 (0.48)	1147	Optimal
Tatjewski	1632.81 (0.34)	3917	Optimal
AdmmSala	1636.19 (0.55)	1302	Optimal

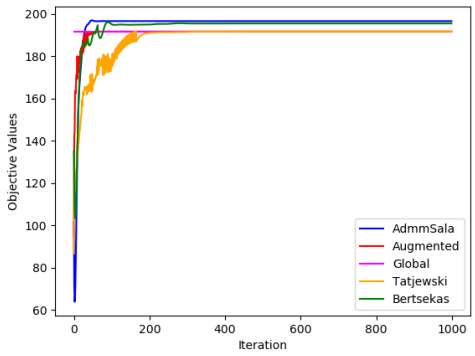


(a) Objective function values

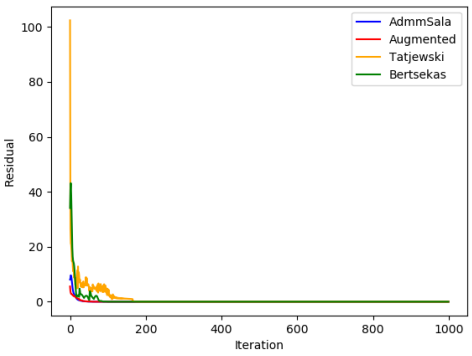


(b) Norm of residual values

Figure 5.4. Medium Problem ($\gamma = 1$ and $\delta = 1$)

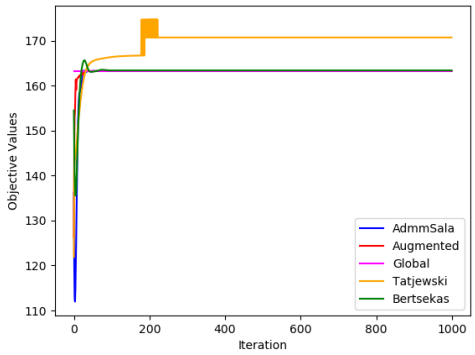


(a) Objective function values

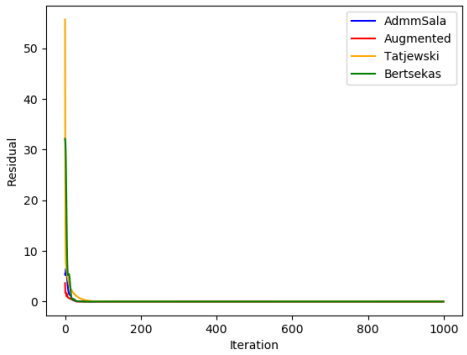


(b) Norm of residual values

Figure 5.5. Medium Problem ($\gamma = 2$ and $\delta = 1$)

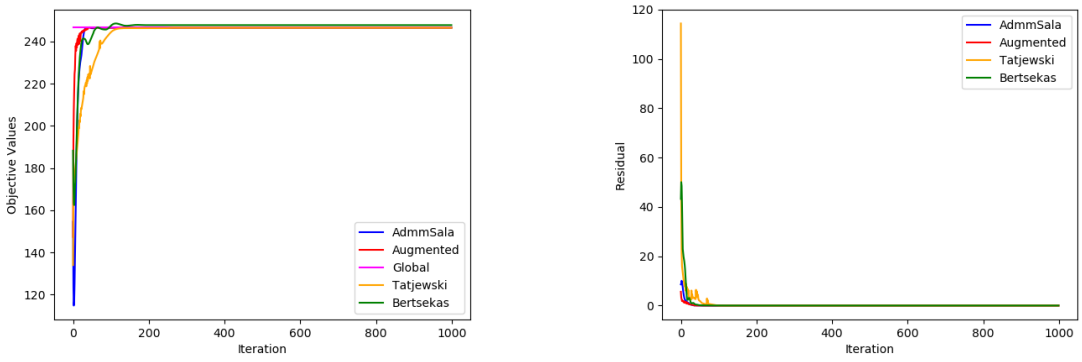


(a) Objective function values



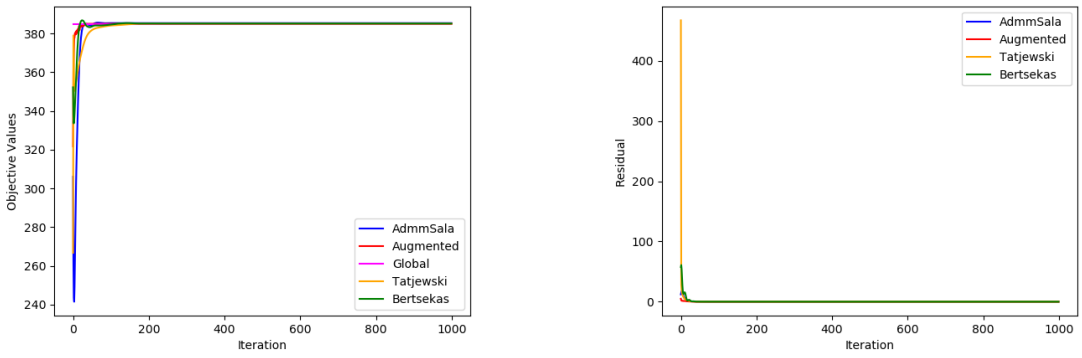
(b) Norm of residual values

Figure 5.6. Medium Problem ($\gamma = 1$ and $\delta = 2$)



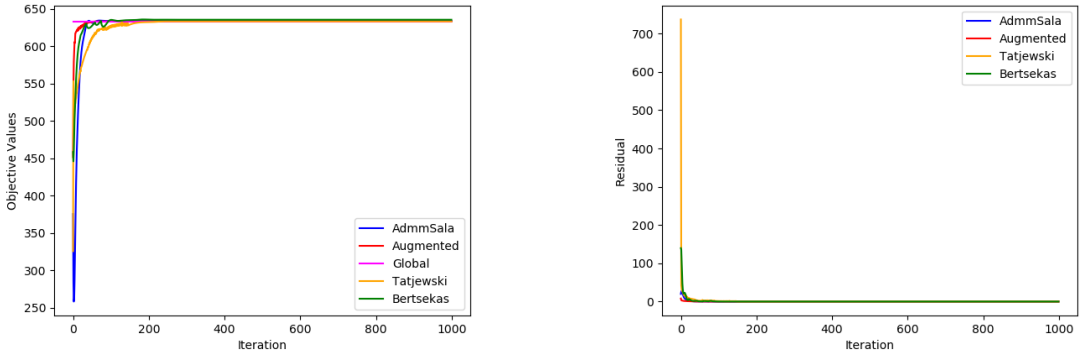
(a) Objective function values **(b)** Norm of residual values

Figure 5.7. Medium Problem ($\gamma = 2$ and $\delta = 2$)



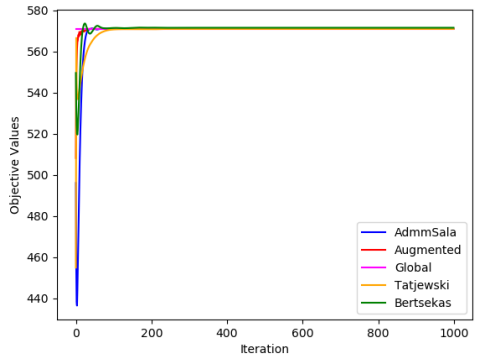
(a) Objective function values **(b)** Norm of residual values

Figure 5.8. Large Problem ($\gamma = 1$ and $\delta = 1$)

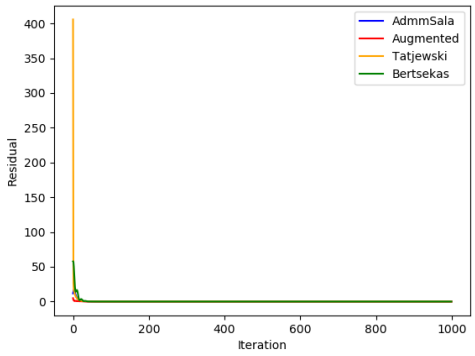


(a) Objective function values **(b)** Norm of residual values

Figure 5.9. Large Problem ($\gamma = 2$ and $\delta = 1$)

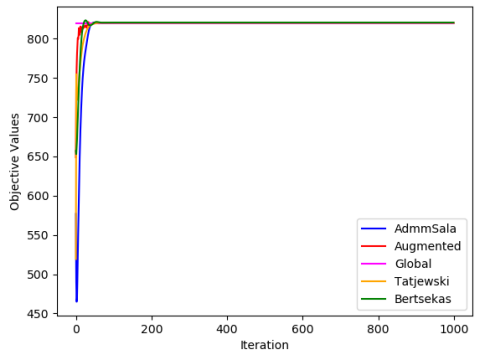


(a) Objective function values

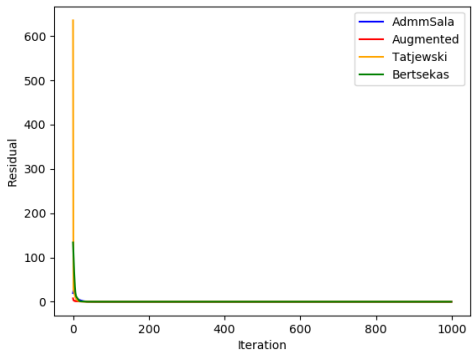


(b) Norm of residual values

Figure 5.10. Large Problem ($\gamma = 1$ and $\delta = 2$)

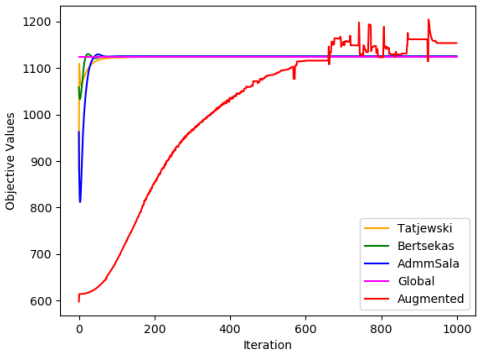


(a) Objective function values

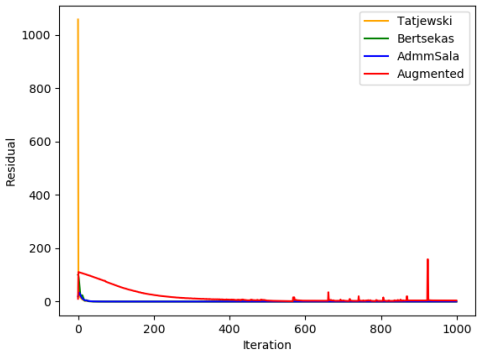


(b) Norm of residual values

Figure 5.11. Large Problem ($\gamma = 2$ and $\delta = 2$)

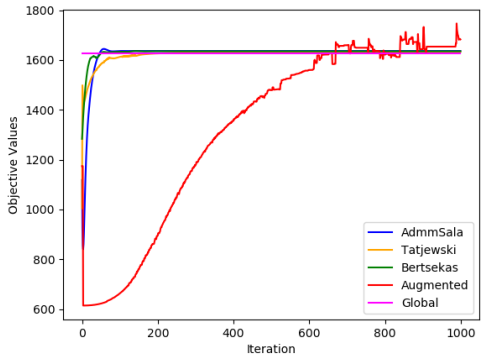


(a) Objective function values

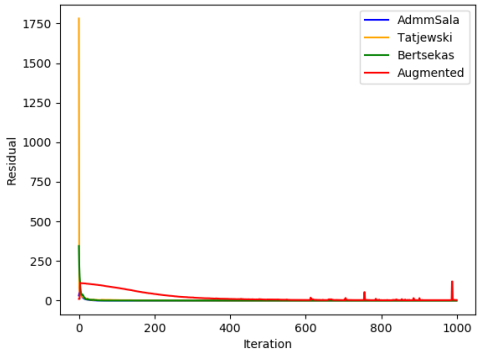


(b) Norm of residual values

Figure 5.12. Extra-Large Problem ($\gamma = 1$ and $\delta = 2$)

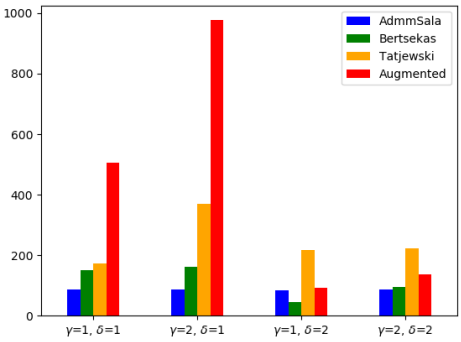


(a) Objective function values

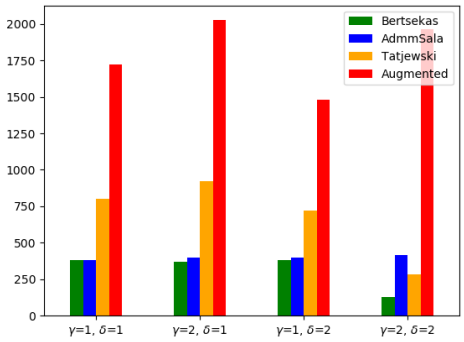


(b) Norm of residual values

Figure 5.13. Extra-Large Problem ($\gamma = 2$ and $\delta = 2$)



(a) Medium Problems



(b) Large Problem

Figure 5.14. Run Times

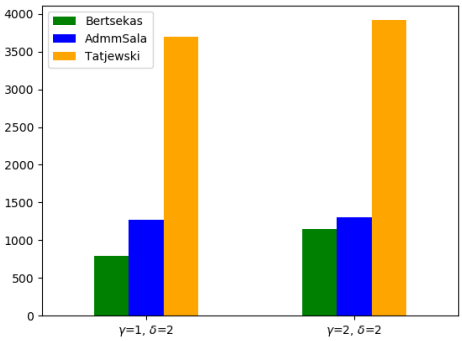


Figure 5.15. Run Times - Extra-Large Problem

6. Synchronous and Asynchronous Solutions of Simultaneous Routing and Bandwidth Allocation in Energy-Aware Networks Using Bertsekas Decomposition Method

Section 5 formulates and evaluates a range of augmented Lagrangian-based algorithms for solving the simultaneous routing and bandwidth allocation problem in energy-aware networks. Among the tested methods, the classical decomposition approach proposed by Bertsekas consistently demonstrates superior performance in terms of both convergence speed and solution quality.

Building on that result, this section investigates two variants of the Bertsekas method: the synchronous and the asynchronous implementations. The synchronous approach operates under a fully coordinated update schedule, requiring global synchronization of all variables at each iteration. While this guarantees consistency, it may introduce significant computational delays in distributed settings. In contrast, the asynchronous approach permits updates based on partially outdated information, allowing components of the algorithm to proceed independently and thus enabling greater parallelism and resilience to communication latency.

The iterative update rules for both variants are formulated below, and their empirical behavior is assessed across multiple network configurations. The objective is to determine the extent to which the asynchronous variant retains the performance benefits of the Bertsekas method while overcoming the coordination bottlenecks of synchronous execution.

6.1. Synchronous Iterative Updates

The augmented Lagrangian in Bertsekas method for the problem (175)-(182) with $x_w^s \in \mathbb{R}_+ \cup \{0\}$ and $b_{wl}^s \in \{0,1\}$ will have the form

$$\begin{aligned} L_\rho(x,b,y,x^s,b^s,y^s,\mu) = \sum_{w\in W} \Bigg\{ & \gamma\big(\bar{x}_w - x_w\big)^2 + \frac{\rho}{2}\,(x_w - x_w^s)^2 \\ & + \sum_{l\in L}\left[\delta\, b_{wl} + \mu_l\left(y_{wl} - \frac{c_l}{|w|}\right) + \frac{\rho}{2}\left((b_{wl}-b_{wl}^s)^2 + (y_{wl}-y_{wl}^s)^2\right)\right] \Bigg\} \\ \triangleq \sum_{w\in W} & \left\{ L_{\rho 1_w}(x_w, x_w^s) + \sum_{l\in L} L_{\rho 2_{wl}}(b_{wl}, y_{wl}, b_{wl}^s, y_{wl}^s, \mu_l) \right\} \end{aligned} \tag{222}$$

where

$$L_{\rho 1_w}(x_w, x_w^s) = \gamma\big(\bar{x}_w - x_w\big)^2 + \frac{\rho}{2}\,(x_w - x_w^s)^2 \tag{223}$$

$$L_{\rho 2_{wl}}(b_{wl}, y_{wl}, b_{wl}^s, y_{wl}^s, \mu_l) = \delta\; b_{wl} + \mu_l \left(\sum_{w \in W} y_{wl} - \frac{c_l}{|w|} \right) + \frac{\rho}{2} \left[(b_{wl} - b_{wl}^s)^2 + (y_{wl} - y_{wl}^s)^2 \right] \tag{224}$$

At iteration $k+1$ using $\xi_k \in [0,1)$,

$$\begin{aligned}
(x_w^{k+1}, b_w^{k+1}, y_w^{k+1}) = & \underset{(x_w, b_w, y_w) \in XBY_w}{\arg\min} \Big\{ L_{\rho 1_w}(x_w, x_w^{s^k}) \\
& + \sum_{l \in L} L_{\rho 2_{wl}}(b_{wl}, y_{wl}, b_{wl}^{s^k}, y_{wl}^{s^k}, \mu_l^k) \Big\}, \quad \forall\, w \in W
\end{aligned} \tag{225}$$

$$x_w^{s^{k+1}} = \xi_k\, x_w^{s^k} + (1 - \xi_k) x_w^{k+1}, \quad \forall\, w \in W \tag{226}$$

$$b_{wl}^{s^{k+1}} = \xi_k\, b_{wl}^{s^k} + (1 - \xi_k) b_{wl}^{k+1}, \quad \forall\, w \in W,\; l \in L \tag{227}$$

$$y_{wl}^{s^{k+1}} = \xi_k\, y_{wl}^{s^k} + (1 - \xi_k) y_{wl}^{k+1}, \quad \forall\, w \in W,\; l \in L \tag{228}$$

$$\mu_l^{k+1} = \max \left\{ 0, \quad \mu_l^k + \beta\rho \Big(\sum_{w \in W} y_{wl}^{k+1} - c_l \Big) \right\}, \quad \forall\, l \in L \tag{229}$$

6.2. Asynchronous Iterative Updates

Let $S_w \subseteq \{1, 2, \ldots, \}$ be a set of iterations at which w adjusts x_w, b_{wl}, y_{wl} based on current knowledge of $\mu_l\ \forall\, l \in L$ and $T_l \subseteq \{1, 2, \ldots, \}$ be a set of iterations at which l adjusts μ_l based on current knowledge of $y_{wl}\ \forall\, w \in W$. The optimization problem (175)-(182) will be solved using a similar approach found in [55] and the Augmented Lagrangian function defined in (222).

For each iteration $k+1$, the asynchronous updates proceed as follows:

w's Algorithm: At iterations, $k = 1, 2, \ldots,$ *and* w:

1: From iteration to iteration, w receives updates μ_l^k from all $l \in L$ and calculate $\hat{\mu}_l^k$.
2: At each update iteration $k' \in S_w$, w chooses the next $x_w^{k+1}, b_w^{k+1}, y_w^{k+1}$ and $x_w^{s^{k+1}}, b_w^{s^{k+1}}, y_w^{s^{k+1}}$

$$(x_w^{k+1}, b_w^{k+1}, y_w^{k+1}) = \underset{(x_w, b_w, y_w) \in XBY_w}{\arg\min} \left\{ L_{\rho 1_w}(x_w, x_w^{s^k}) + \sum_{l \in L} L_{\rho 2_{wl}}(b_{wl}, y_{wl}, b_{wl}^{s^k}, y_{wl}^{s^k}, \hat{\mu}_l^k) \right\} \tag{230}$$

$$x_w^{s^{k+1}} = \xi_k x_w^{s^k} + (1 - \xi_k) x_w^{k+1} \tag{231}$$

$$b_{wl}^{s^{k+1}} = \xi_k b_{wl}^{s^k} + (1 - \xi_k) b_{wl}^{k+1}, \quad \forall\, l \in L \tag{232}$$

$$y_{wl}^{s^{k+1}} = \xi_k y_{wl}^{s^k} + (1 - \xi_k) y_{wl}^{k+1}, \quad \forall\, l \in L \tag{233}$$

Transmission is done at this rate until the next update

3: Communicates $y_{wl}^{k+1}\ \forall\, l \in L$ to all constraints controllers.

l's Algorithm: At iterations, $k = 1, 2, \ldots,$ *and* l:

1: From iteration to iteration, l receives updates $y_{wl}^{k+1}\ \forall\, l \in L$ from all $w \in W$.

2: At each update iteration $k' \in T_l$, l updates μ_l^{k+1} such that

$$\mu_l^{k+1} = \max \left\{ 0, \quad \mu_l^k + \beta \rho \Big(\sum_{w \in W} \hat{y}_{wl}^{k+1} - c_l \Big) \right\} \tag{234}$$

Transmission is done at this rate until the next update

3: Communicates μ_l^{k+1} to all local optimizers.

Stop when $x_w^{k+1} = x_w^{s^{k+1}}$, $b_{wl}^{k+1} = b_{wl}^{s^{k+1}}$, $y_{wl}^{k+1} = y_{wl}^{s^{k+1}}$, otherwise increase k by 1 and go to step 1. Where

$$\hat{y}_{wl}^k = \sum_{k'=k-k^0}^{k} a_{wl}(k', k) y_{wl}^{k'}, \quad w \in W\ l \in L \tag{235}$$

$$\hat{\mu}_l^k = \sum_{k'=k-k^0}^{k} b_l(k', k) \mu_l^{k'}, \quad l \in L \tag{236}$$

and

$$\sum_{k'=k-k^0}^{k} a_{wl}(k', k) = 1, \quad \forall k,\ w \in W\ l \in L \tag{237}$$

$$\sum_{k'=k-k^0}^{k} b_l(k', k) = 1, \quad \forall k,\ l \in L \tag{238}$$

and

$$a_{wl}(k',k) \geq 0, \quad \forall k,\ w \in W\ l \in L \tag{239}$$

$$b_l(k',k) \geq 0, \quad \forall k,\ l \in L \tag{240}$$

6.3. Numerical Tests

The dataset is as described in detail in Section 5.3.1

6.3.1. Implementation Details

All implementations and numerical experiments were conducted using Python 3.12.0. The optimization models were formulated with the Pyomo modeling framework and solved using the Gurobi optimizer. Network construction and visualization tasks were handled using the NetworkX library [54], facilitating the representation of graph-based structures integral to the optimization problems. Computational experiments were performed on a machine equipped with an AMD Ryzen 5 4600H processor (3.00 GHz, Radeon Graphics), 32 GB of RAM, and a 512 GB SSD, running a 64-bit Windows 10 Pro operating system. All experiments were executed in parallel using Python's `multiprocessing.Pool` to simulate distributed computation.

The individual results were first aligned to a unified time axis to enable consistent comparison across datasets with potentially different time indices. A comprehensive timeline was constructed by taking the union of all time points in the datasets. Each dataset was then merged onto this unified timeline using nearest-neighbor matching using *merge_asof* from Pandas [56], ensuring that for each time point in the reference axis, the closest available record from each dataset was selected. This approach preserves temporal coherence while allowing synchronized evaluation of multiple time series, even in non-uniform or asynchronous sampling intervals.

Artificial delays were introduced in selected data partitions during optimization to simulate realistic conditions in distributed environments [1], [57]. These delays emulate scenarios where partial data or subproblem updates arrive late due to system latency, communication bottlenecks, or asynchronous computation schedules in large-scale systems. The effect of these delays was evaluated by comparing performance under expected timing and the delayed setting for both synchronous and asynchronous algorithm variants.

6.3.2. Results and Discussion

In Table 6.1, the objective values and runtimes are reported for each network problem instance under synchronous and asynchronous optimization, evaluated with and without simulated delays. Consistent trends are observed across all ten instances, spanning medium, large, and extra-large network topologies, which highlight the trade-offs between the two algorithmic paradigms.

Both formulations yielded nearly identical objective values across all settings without delay, indicating that the asynchronous method achieves comparable solution quality to the synchronous baseline. The minor discrepancies observed (e.g., Medium $\gamma = 2, \delta = 1$ with 163.49 vs. 163.54) are negligible and within numerical tolerance, affirming the correctness of the asynchronous approach. Runtime comparisons without delay show a more nuanced pattern. In medium-sized networks, asynchronous optimization is generally completed faster than synchronous optimization. For instance, in the Medium $\gamma = 2, \delta = 1$ case, asynchronous execution completed in 10 seconds compared to 19 seconds for its synchronous counterpart. However, in some large network instances (e.g., Large $\gamma = 1, \delta = 2$), the asynchronous method was slightly slower (91 seconds vs. 58 seconds), likely due to the overhead of managing loosely coupled updates at scale. Nevertheless, the differences in runtimes were not drastic under normal execution conditions.

The introduction of artificial delays in selected data partitions dramatically magnified performance differences. In all medium and large network instances, synchronous runtimes increased significantly, often by an order of magnitude or more. For example, when delays were introduced, the synchronous runtime in the Medium $\gamma = 1, \delta = 1$ case jumped from 16 to 592 seconds. In contrast, the asynchronous runtime increased modestly, from 15 to 42 seconds. Similar resilience was observed across all configurations. Even in large-scale instances, the asynchronous approach maintained substantially lower runtimes than its synchronous counterpart under delayed conditions. For example, in the Large $\gamma = 1, \delta = 2$ case, the asynchronous runtime with delay was 183 seconds, while the synchronous method required 473 seconds.

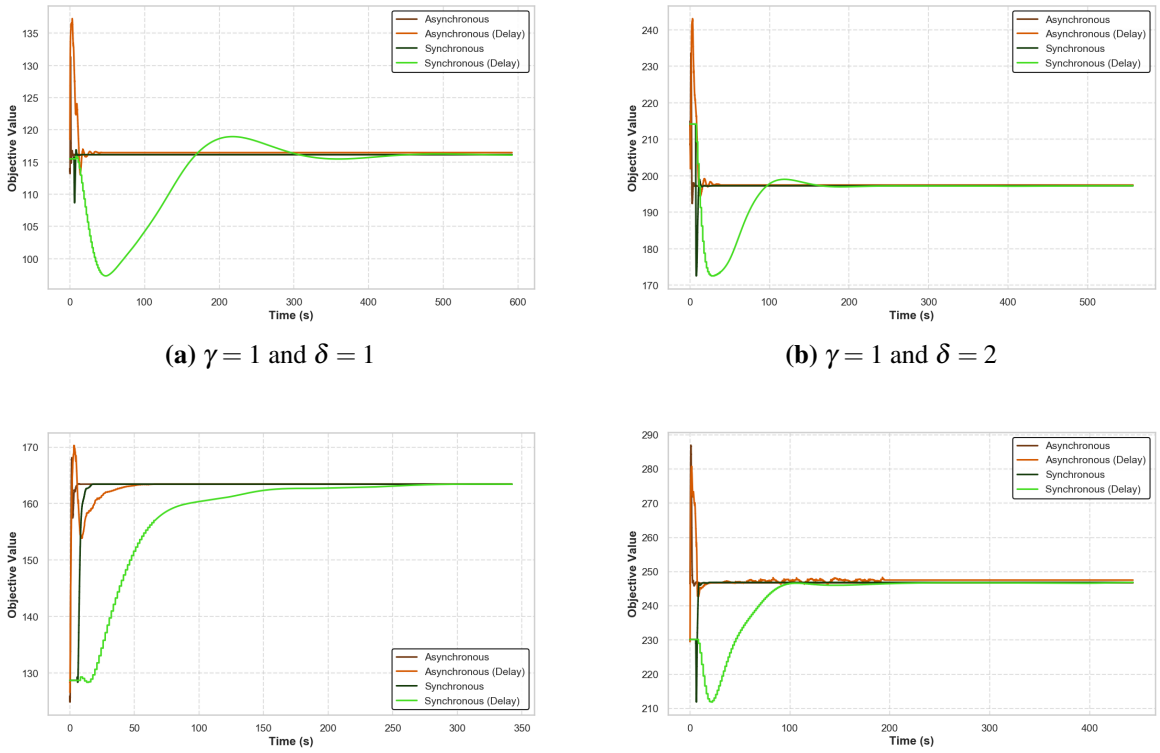
Figure 6.5 presents a visualization of routing paths over a directed network graph (Small Problem Networks), where each node represents a router or a network element, and directed edges represent potential communication links. Using the spring layout for spatial arrangement, the plot shows the routing solutions for multiple commodities $w \in W$, where each commodity represents a separate source-destination traffic demand pair. Flows are illustrated by color-coded directed edges, with each color corresponding to a different commodity. Edge labels indicate the commodity identifier and the flow value y_{wl} transmitted over that edge, truncated to a single decimal for clarity. Only edges with a non-negligible flow (above a threshold of 10^{-3}) are visualized to reduce clutter and emphasize active routes.

Optionally, each commodity's source and destination nodes can be annotated with $S(w)$ and $D(w)$, respectively, to highlight the entry and exit points of data within the network. This visualization effectively captures both the structure of the network and the routing decisions for multiple commodities, enabling a direct comparison of path overlaps, bottlenecks, and routing efficiency across different scenarios or algorithmic configurations.

Overall, the results demonstrate that while both methods yield solutions of equivalent quality, asynchronous optimization exhibits significantly better robustness to delays and scales more gracefully with problem size; this makes it a compelling strategy for real-world distributed optimization problems where partial staleness and communication latency are common.

Table 6.1. Routing Results

Data	Objective		Objective (with delay)		Time [s]		Time (with delay) [s]	
	Sync	Async	Sync	Async	Sync	Async		
Medium $\gamma = 1, \delta = 1$	116.13	116.13	116.1	116.52	16	15	592	42
Medium $\gamma = 1, \delta = 2$	197.27	197.27	197.27	197.47	30	28	557	38
Medium $\gamma = 2, \delta = 1$	163.49	163.54	163.49	163.54	19	10	342	65
Medium $\gamma = 2, \delta = 2$	246.84	246.69	246.85	246.62	19	14	443	195
Large $\gamma = 1, \delta = 1$	386.81	386.76	386.81	386.64	301	110	628	123
Large $\gamma = 1, \delta = 2$	636.63	636.78	636.63	636.68	58	91	473	183
Large $\gamma = 2, \delta = 1$	571.37	571.50	571.37	571.52	200	142	384	117
Large $\gamma = 2, \delta = 2$	821	821	821	821	109	78	1020	115
Extra Large $\gamma = 2, \delta = 1$	1125.21	1125.22	1125.21	1125.26	127	388	461	429
Extra Large $\gamma = 2, \delta = 2$	1636.44	1635.70	1636.44	1636.45	149	354	509	400



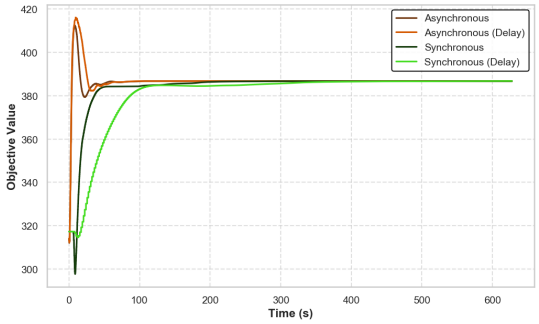
(a) $\gamma = 1$ and $\delta = 1$

(b) $\gamma = 1$ and $\delta = 2$

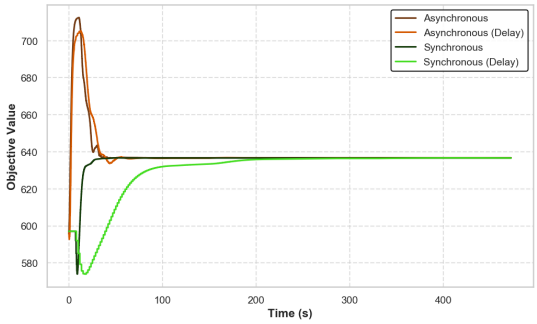
(c) $\gamma = 2$ and $\delta = 1$

(d) $\gamma = 2$ and $\delta = 2$

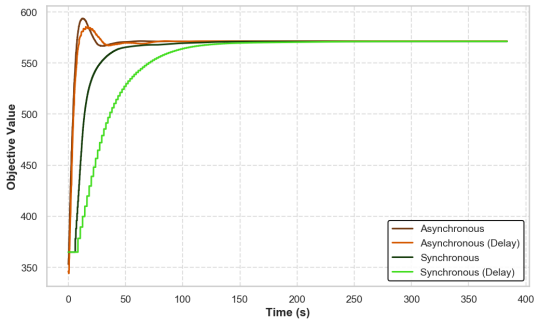
Figure 6.1. Medium Problem Objective Values (Routing)



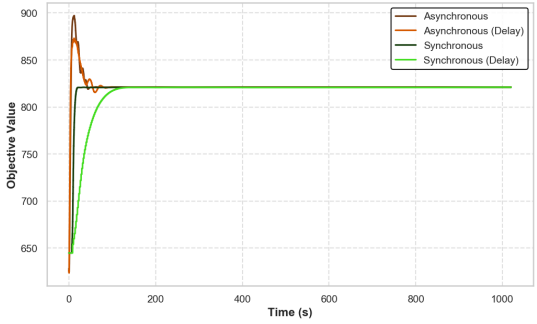
(a) $\gamma = 1$ and $\delta = 1$



(b) $\gamma = 1$ and $\delta = 2$

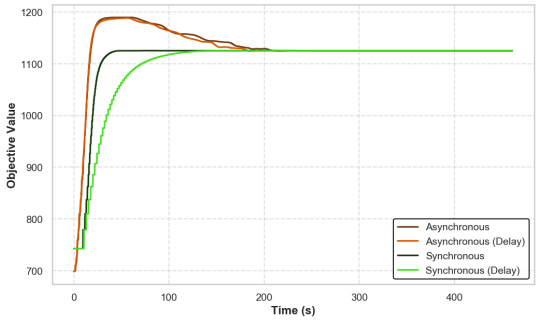


(c) $\gamma = 2$ and $\delta = 1$

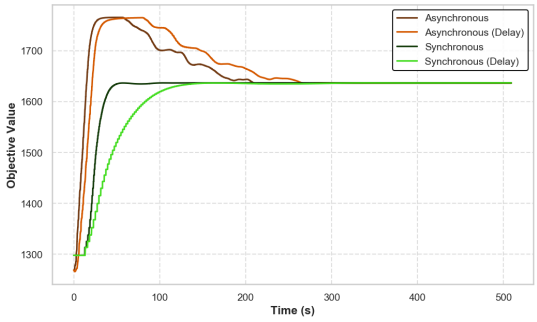


(d) $\gamma = 2$ and $\delta = 2$

Figure 6.2. Large Problem Objective Values (Routing)

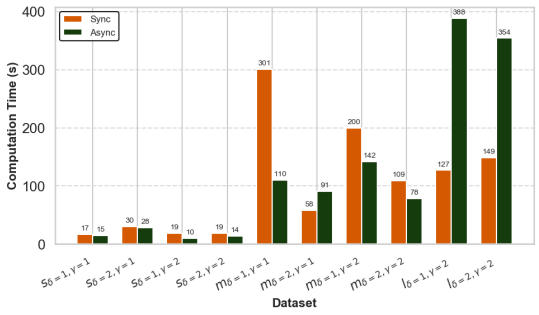


(a) $\gamma = 1$ and $\delta = 1$

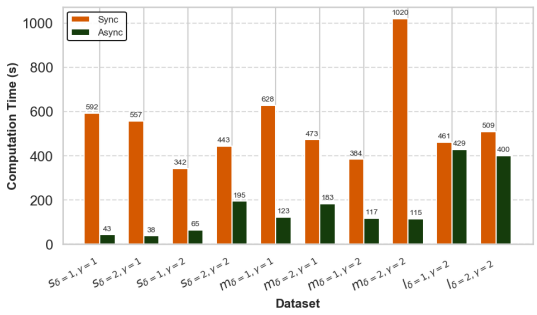


(b) $\gamma = 1$ and $\delta = 2$

Figure 6.3. Extra Large Problem Objective Values (Routing)

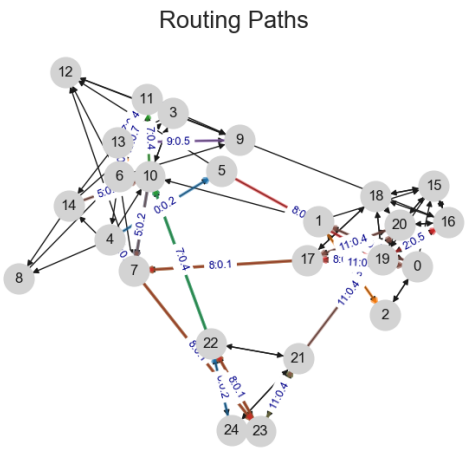


(a) Without Delay

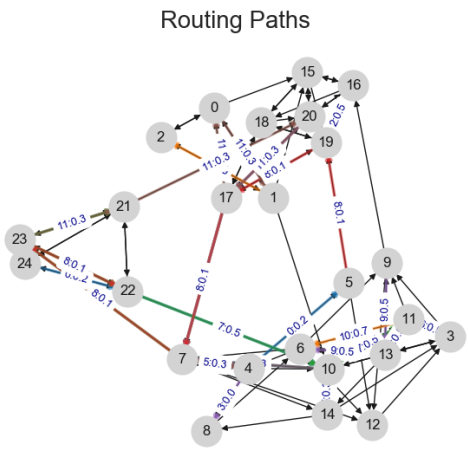


(b) With Delay

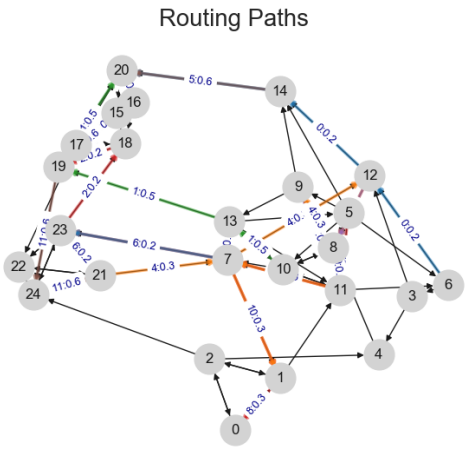
Figure 6.4. Run Time (Routing)



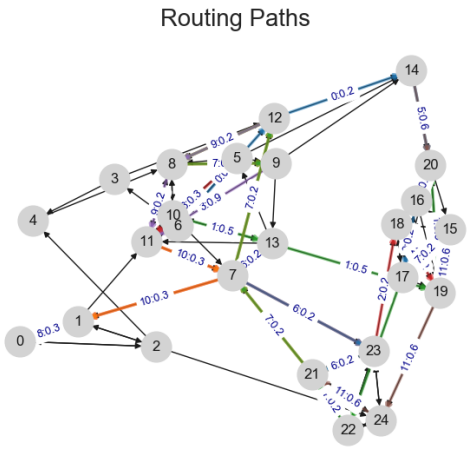
(a) Sync $\gamma = 1$ and $\delta = 1$



(b) Async $\gamma = 1$ and $\delta = 1$



(c) Sync $\gamma = 2$ and $\delta = 2$



(d) Async $\gamma = 2$ and $\delta = 2$

Figure 6.5. Small Problem Networks (Routing)

7. Solution of Regularized Linear Systems with Augmented Lagrangian Algorithms

In various fields such as signal processing, machine learning, numerical optimization, and high-dimensional statistics, the need to obtain stable and computationally efficient solutions from linear systems has driven the study of regularized approaches. Consider a general linear system of the form $Ax = b$, where $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and m, n are the number of samples and features respectively. Regardless of whether the system is underdetermined, overdetermined, or exactly determined, practical considerations often necessitate regularizing the solution to enhance stability, prevent overfitting, or deal with ill-conditioning in the data.

Among the most widely used strategies is ℓ_2-regularization, which discourages large solution norms and improves numerical conditioning without necessarily enforcing sparsity. Unlike the ℓ_0 pseudo-norm or its convex surrogate ℓ_1, the ℓ_2 norm promotes smoothness and penalizes large coefficients, yielding well-behaved solutions that are particularly attractive in noisy or high-dimensional regimes.

To formalize this, let $P \subseteq \{1,\dots,n\}$ index the decision variables and $M = \{1,\dots,m\}$ index the constraints. Let $a_i \in \mathbb{R}^{|P|}$ denote the feature vector of the i-th observation, and let $y_i \in \mathbb{R}$ denote the associated response. The regularized recovery problem can then be expressed as the following constrained ℓ_2-minimization problem:

$$\min_x \quad \sum_{j\in P} x_j^2, \quad \text{s.t.} \sum_{j\in P} a_{ij}x_j = y_i, \quad \forall i \in M \tag{241}$$

Here, $x_j \in \mathbb{R}$ for all $j \in P$ are the optimization variables, and the equality constraints enforce exact satisfaction of the observed data. The objective function imposes an ℓ_2-norm penalty, which encourages solutions with small magnitudes and improved robustness to noise and multicollinearity. This formulation underlies classical techniques such as Tikhonov regularization and ridge regression, and plays a central role in contemporary optimization-based methods for solving linear inverse problems.

7.1. Decomposition of the Regularized Linear Systems

The problem formulation in (241) lends itself naturally to decomposition techniques that are well-suited for large-scale and distributed optimization. This section presents several decomposition approaches for solving the regularized linear systems, each grounded in the framework of augmented Lagrangian methods and dual ascent strategies.

The key idea underlying these methods is to exploit the separability of the objective function

and structure of the constraints to design efficient iterative schemes. In particular, variants of the classical Lagrangian method are considered, as well as augmented Lagrangian formulations including the Multiplier Method, Bertsekas Method, Tatjewski Method, and the SALA (Separated Augmented Lagrangian Algorithm) version of the Alternating Direction Method of Multipliers (ADMM).

These methods enable the decoupling of variables and facilitate parallel or distributed updates, making them attractive for high-dimensional optimization problems commonly encountered in compressed sensing and machine learning.

7.1.1. The Lagrangian

The Lagrangian of Problem (241) can be written as

$$\begin{aligned} L(x,\lambda) &= \sum_{j\in P}\left[x_j^2 + \sum_{i\in M}\lambda_i\Big(a_{ij}x_j - \frac{y_i}{|P|}\Big)\right] \\ &= \sum_{j\in P} L_j(x_j,\lambda) \end{aligned} \tag{242}$$

where $L_j(x_j,\lambda) = x_j^2 + \sum_{i\in M}\lambda_i\Big(a_{ij}x_j - \frac{y_i}{|P|}\Big)$. At iteration $k+1$, with $\lambda_i \in \mathbb{R}$ and $\rho_k > 0$, the dual variables are updated according to the following rule:

$$x_j^{k+1} = \min_{x_j} L_j(x_j,\lambda^k), \quad \forall j \in N \tag{243}$$

$$\lambda_i^{k+1} = \lambda_i^k + \rho_k\Big(\sum_{j\in P} a_{ij}x_j^{k+1} - y_i\Big), \quad \forall i \in M \tag{244}$$

7.1.2. The Multiplier Method

The classical augmented Lagrangian method introduces a quadratic penalty term to improve convergence properties of the basic Lagrangian scheme. The augmented Lagrangian of Problem (241) becomes:

$$L_\rho(x,\lambda) = \sum_{j\in P} x_j^2 + \sum_{i\in M}\lambda_i\Big(\sum_{j\in P} a_{ij}x_j - y_i\Big) + \frac{\rho}{2}\sum_{i\in M}\Big(\sum_{j\in P} a_{ij}x_j - y_i\Big)^2 \tag{245}$$

At iteration $k+1$, with $\lambda_i \in \mathbb{R}$, and $\rho_k > 0$, the dual variables are updated according to the following rule:

$$
\begin{aligned}
x^{k+1} &= \min_{x} L_\rho(x, \lambda^k) \\
\lambda_i^{k+1} &= \lambda_i^k + \rho_k \Big(\sum_{j \in P} a_{ij} x_j^{k+1} - y_i \Big), \quad \forall i \in M
\end{aligned} \tag{246}
$$

7.1.3. The Bertsekas Method

The Bertsekas augmented Lagrangian method leverages a coordinate-wise structure, decoupling the objective further across variables x_j. The resulting formulation is:

$$
\begin{aligned}
L_\rho(x, x^s, \lambda) &= \sum_{j \in P} \left\{ x_j^2 + \sum_{i \in M} \lambda_i \Big(a_{ij} x_j - \frac{y_i}{|P|} \Big) + \frac{\rho}{2} \Big(x_j - x_j^s \Big)^2 \right\} \\
&= \sum_{j \in P} L_{\rho_j}(x_j, x_j^s, \lambda)
\end{aligned} \tag{247}
$$

Here, x_j^s represents a surrogate or anchor value from the previous iteration. At iteration $k+1$, with $\lambda_i \in \mathbb{R}$, $\rho_k > 0$, and $\zeta = [0, 1)$ the dual variables are updated according to the following rule:

$$
x_j^{k+1} = \min_{x_j} L_{\rho_j}(x_j, x_j^{s^k}, \lambda^k), \quad \forall j \in N \tag{248}
$$

$$
x_j^{s^{k+1}} = \zeta x_j^{s^k} + (1 - \zeta) x_j^{k+1}, \quad \forall j \in N \tag{249}
$$

$$
\lambda_i^{k+1} = \lambda_i^k + \rho_k \Big(\sum_{j \in P} a_{ij} x_j^{k+1} - y_i \Big), \quad \forall i \in M \tag{250}
$$

7.1.4. The Tatjewski Method

The Tatjewski method incorporates a more refined surrogate mechanism by coupling updates of x_j with fixed values of the remaining variables, leading to a partially separable augmented Lagrangian:

$$
\begin{aligned}
L_\rho(x, x^s, \lambda) &= \sum_{j \in P} \left\{ x_j^2 + \sum_{i \in M} \lambda_i \Big(a_{ij} x_j - \frac{y_i}{|P|} \Big) + \frac{\rho}{2} \sum_{i \in M} \Big(a_{ij} x_j - \frac{y_i}{|P|} + \sum_{l \in P, l \neq j} a_{il} x_l^s \Big)^2 \right\} \\
&= \sum_{j \in P} L_{\rho_j}(x_j, x_j^s, \lambda)
\end{aligned} \tag{251}
$$

This technique permits effective iterative refinement and is particularly useful in parallel implementations. At iteration $k+1$, with $\lambda_i \in \mathbb{R}$, $\rho_k > 0$, and $\zeta = [0, 1)$ the dual variables

are updated according to the following rule:

$$x_j^{k+1} = \min_{x_j} L_{\rho_j}(x_j, x_j^{s^k}, \lambda^k), \quad \forall j \in N \tag{252}$$

$$x_j^{s^{k+1}} = \zeta x_j^{s^k} + (1-\zeta) x_j^{k+1}, \quad \forall j \in N \tag{253}$$

$$\lambda_i^{k+1} = \lambda_i^k + \rho_k \Big(\sum_{j \in P} a_{ij} x_j^{k+1} - y_i \Big), \quad \forall i \in M \tag{254}$$

7.1.5. The ADMM (SALA Version)

The Separated Augmented Lagrangian Algorithm (SALA) introduces auxiliary variables s_{ij} to explicitly split the affine constraints (see section 3.7). The corresponding augmented Lagrangian is given by:

$$\begin{aligned} L_\rho(x,s,\lambda) &= \sum_{j \in P} \left\{ x_j^2 + \sum_{i \in M} \lambda_i \Big(a_{ij} x_j - \frac{y_i}{|P|} - s_{ij} \Big) + \frac{\rho}{2} \sum_{i \in M} \Big(a_{ij} x_j - \frac{y_i}{|P|} - s_{ij} \Big)^2 \right\} \\ &= \sum_{j \in P} L_{\rho_j}(x_j, s_j, \lambda) \end{aligned} \tag{255}$$

At iteration $k+1$, the algorithm proceeds with the following updates:

$$x_j^{k+1} = \arg\min_{x_j} L_{\rho_j}(x_j, s_j^k, \lambda^k), \quad j \in P \tag{256}$$

$$r_i^{k+1} = \sum_{j \in P} a_{ij} x_j^{k+1} - y_i, \quad i \in M \tag{257}$$

$$s_{ij}^{k+1} = a_{ij} x_j^{k+1} - \frac{y_i}{|P|} - \frac{r_i^{k+1}}{|P|}, \quad j \in P, i \in M \tag{258}$$

$$\lambda_i^{k+1} = \lambda_i^k + \frac{\rho_k}{|P|} r_i^{k+1}, \quad i \in M \tag{259}$$

$$\rho_{k+1} = \alpha \rho_k, \quad \alpha \geq 1 \tag{260}$$

This method allows efficient decoupling and parallel updates of the variables while maintaining primal feasibility via residual tracking.

7.2. Numerical Tests

7.2.1. Dataset Description

The experiments are conducted on three linear systems designed to reflect varying data structures and matrix conditions. Two are derived from an image inpainting task using randomized diagonal measurement operators, and the third is based on a biomedical dataset

reformulated as a compressed feature recovery problem. Each dataset adheres to the standard linear model $Ax = y$, with a known ground truth $x^\star$.

The first dataset simulates a diagonal sensing problem in grayscale image recovery. A 16×16 grayscale image is vectorized into $x^\star \in \mathbb{R}^{256}$, and a diagonal measurement matrix $A \in \mathbb{R}^{256\times256}$ is constructed with entries drawn independently from a uniform distribution over $[0, 1)$. The corresponding observation vector is computed as $y = Ax^\star$, resulting in a well-scaled and positive semi-definite system. This formulation captures moderate conditioning and serves as a stable reference for evaluating solution quality.

The second dataset uses the same underlying image and construction but replaces the uniform distribution with a standard Gaussian distribution for the diagonal entries of A. This results in a more ill-conditioned system where entries can be both positive and negative, potentially with large magnitude. The measurement vector $y = Ax^\star$, therefore, reflects a noisier and more variable scaling of the original image, posing greater challenges for algorithmic recovery under unstable and zero-mean multiplicative transformations.

The third dataset is derived from the UCI Breast Cancer Wisconsin (Diagnostic) dataset [58] and formulated as a compressed sensing task. After standardizing the data, a single feature vector $x^\star \in \mathbb{R}^{30}$ is selected from the training partition. A sensing matrix $A \in \mathbb{R}^{10\times30}$ is generated with independent standard Gaussian entries scaled by $1/\sqrt{10}$. The measurement vector $y = Ax^\star$ thus represents a low-dimensional projection of the original biomedical profile. This setting is representative of practical dimensionality reduction problems in clinical data, where reconstruction must be achieved from limited and noisy observations.

7.2.2. Implementation Details

All implementations and numerical experiments were conducted using Python 3.12.0. The optimization models were formulated with the Pyomo modeling framework and solved using the Gurobi optimizer. Computational experiments were performed on a machine equipped with an AMD Ryzen 5 4600H processor (3.00 GHz, Radeon Graphics), 32 GB of RAM, and a 512 GB SSD, running a 64-bit Windows 10 Pro operating system.

The individual results were first aligned to a unified time axis to enable consistent comparison across datasets with potentially different time indices. A comprehensive timeline was constructed by taking the union of all time points in the datasets. Each dataset was then merged onto this unified timeline using nearest-neighbor matching via an as-of merge (***pandas.merge_asof*** *performs a merge by nearest key rather than exact matches, aligning rows based on the closest preceding key in a sorted dataset. It is especially useful for time-series data to join on nearest timestamps without requiring exact equality*), ensuring that for each time point in the reference axis, the closest available record from each dataset

was selected. This approach preserves temporal coherence while allowing synchronized evaluation of multiple time series, even in non-uniform or asynchronous sampling intervals.

7.2.3. Results and Discussion

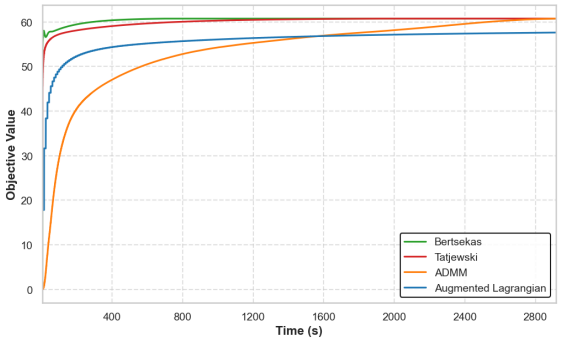
Table 7.1 summarizes the performance of four optimization algorithms, ADMM, Bertsekas, Tatjewski's method, and the classical Augmented Lagrangian, across three datasets with and without variable partitioning. Across the Gaussian and Uniform datasets, which feature large and ill-conditioned linear systems, Bertsekas' method consistently achieves the best trade-off between convergence and computational efficiency. It converges in all settings and does so significantly faster when the problem is decomposed into 12 partitions. ADMM and Tatjewski both fail to converge within the 5000-second limit in the unpartitioned setting but improve under decomposition, highlighting the benefits of problem structure exploitation. The Augmented Lagrangian method does not converge in time for either of the datasets and lacks implementation under decomposition.

On the Cancer dataset, which involves a significantly smaller and better-conditioned system, all algorithms (except Augmented Lagrangian) converge rapidly in both partitioned and unpartitioned forms. Here, Bertsekas again records the lowest runtime, while ADMM and Tatjewski exhibit modestly higher computational cost. The Augmented Lagrangian method produces a suboptimal objective value and performs poorly relative to the others.

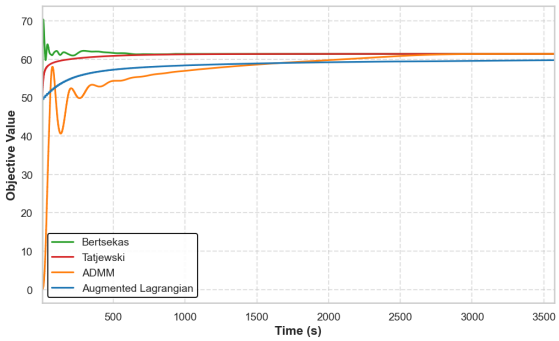
These results indicate that dual ascent approaches, particularly Bertsekas' method, are robust across system scales and benefit markedly from decomposition. In contrast, classical methods like ADMM and Augmented Lagrangian suffer in high-dimensional settings without partitioning, with the latter also being sensitive to problem scaling.

Table 7.1. Performance Comparison of Optimization Algorithms for Regularized Linear Systems Across Different Datasets

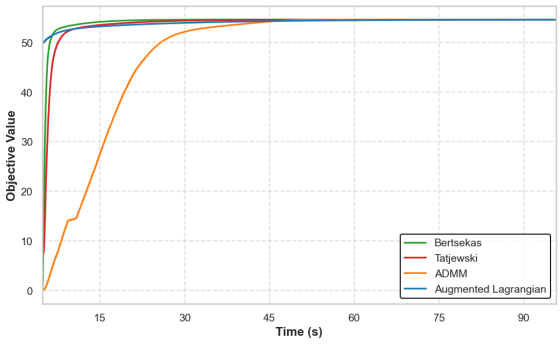
		No Partition			12 Partitions		
Dataset	**Algorithm**	**Objective**	**Time**(s)	Status	**Objective**	**Time**(s)	Status
Gaussian	ADMM	59.01	5001	Time Out	61.4	3396	Converged
	Bertsekas	61.41	4880	Converged	61.41	938	Converged
	Tatjewski	61.03	5010	Time Out	61.41	2372	Converged
	Aug Lagrangian	61.05	5001	Time Out	-	-	-
Uniform	ADMM	57.04	5000	Time Out	60.67	2916	Converged
	Bertsekas	60.87	3095	Converged	60.87	724	Converged
	Tatjewski	59.78	5009	Time Out	60.87	1983	Converged
	Aug Lagrangian	59.92	5003	Time Out	-	-	-
Cancer	ADMM	54.61	150	Converged	54.61	55	Converged
	Bertsekas	54.61	119	Converged	54.61	55	Converged
	Tatjewski	54.61	309	Converged	54.61	95	Converged
	Aug Lagrangian	187	150	Converged	-	-	-



(a) Uniform

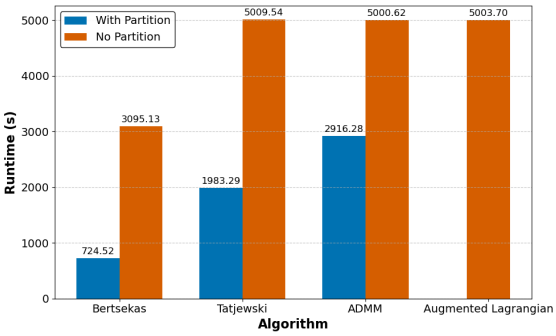


(b) Gaussian

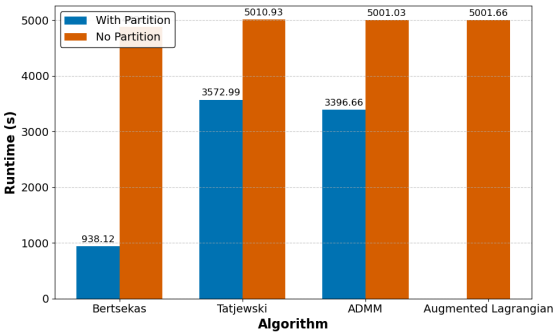


(c) Cancer Spars

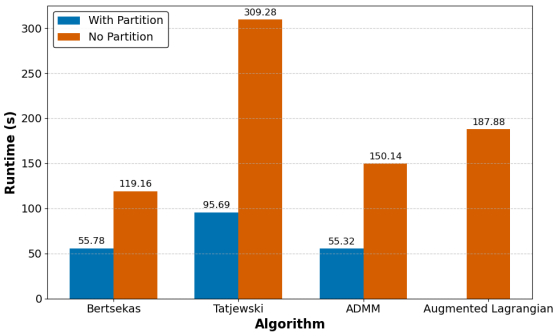
Figure 7.1. Quadratic: Objective (secs)



(a) Uniform

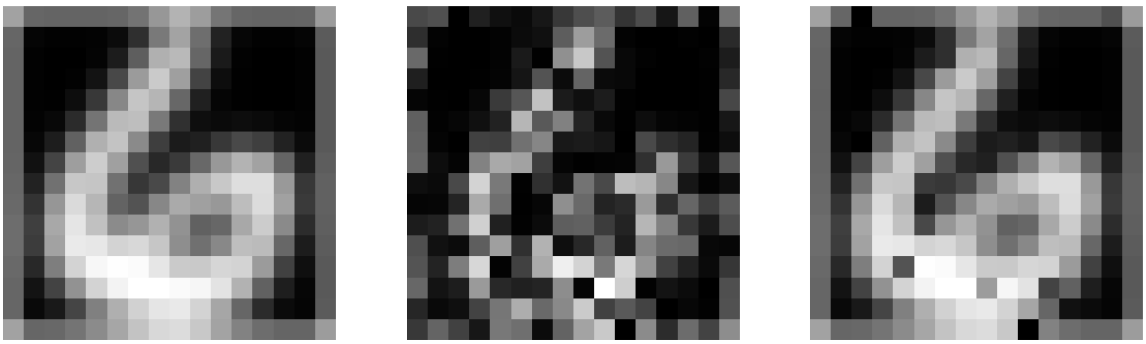


(b) Gaussian

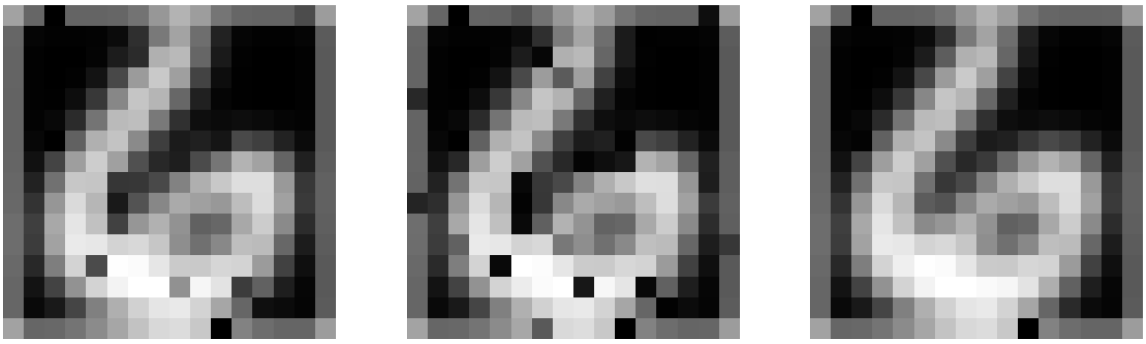


(c) Cancer Spars

Figure 7.2. Quadratic: Run Time (secs)

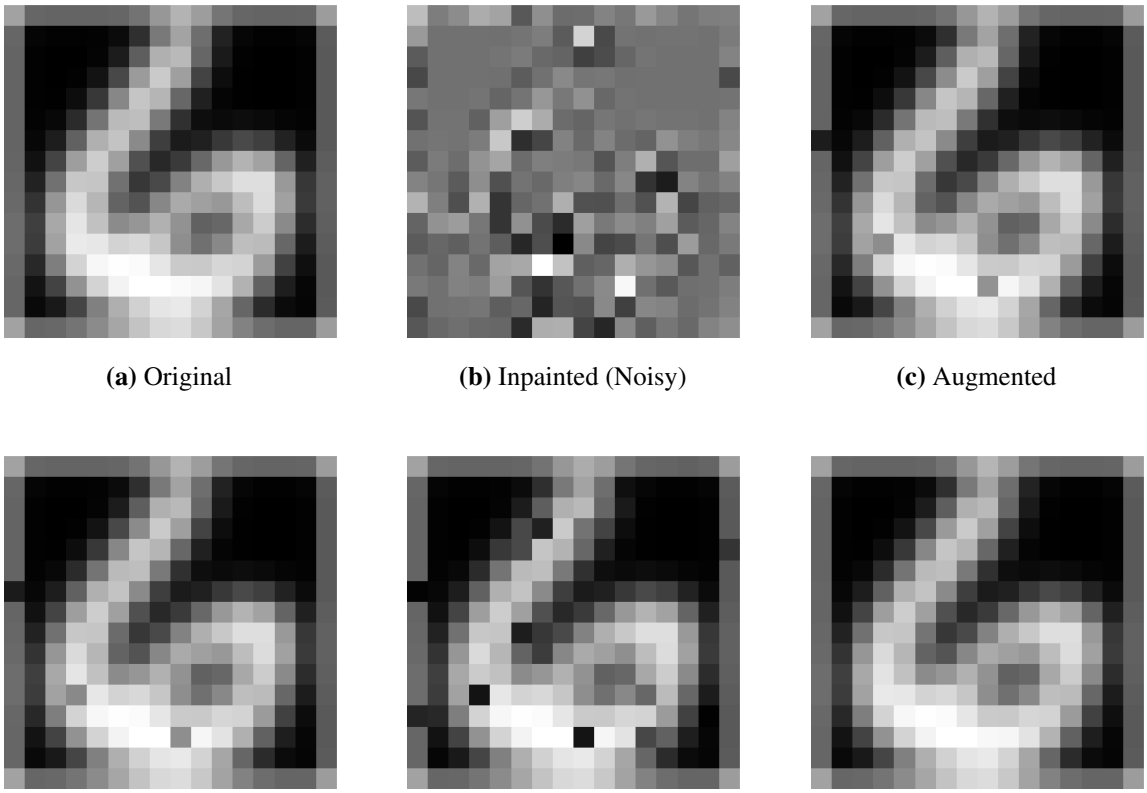


(a) Original **(b)** Inpainted (Noisy) **(c)** Augmented



(d) Tatjewski **(e)** ADMM **(f)** Bertsekas

Figure 7.3. Image Recovery under Uniform Additive Noise



(a) Original **(b)** Inpainted (Noisy) **(c)** Augmented

(d) Tatjewski **(e)** ADMM **(f)** Bertsekas

Figure 7.4. Image Recovery under Gaussian Additive Noise

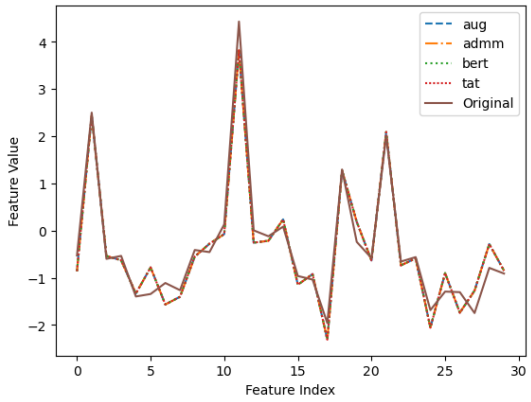


Figure 7.5. Compressed Measurements of Cancer Spars

8. Synchronous and Asynchronous Solutions of Regularized Systems of Linear Equations Using Bertsekas Decomposition Method

This chapter explores both synchronous and asynchronous update schemes within the Bertsekas augmented Lagrangian framework, highlighting their algorithmic structures and practical performance across diverse datasets. The following sections present the theoretical formulation, implementation details, and comparative results that demonstrate the effectiveness and efficiency of these approaches.

To facilitate the use of algorithms that operate on inequality constraints, the equality constraints are equivalently rewritten as a pair of inequalities; hence (241) becomes:

$$\min_{x} \quad \sum_{j\in P} x_j^2, \quad \text{s.t.} \begin{cases} \sum_{j\in P} A_{ij}x_j - y_i \leq 0, \\ -\sum_{j\in P} A_{ij}x_j + (1+\varepsilon)y_i \leq 0, \end{cases} \qquad \forall i \in M. \tag{261}$$

Where ε is a small constant. This reformulation preserves the feasible set of the original problem while ensuring compatibility with augmented Lagrangian-based optimization techniques that rely on inequality-constrained formulations.

8.1. Synchronous Iterative Updates

The augmented Lagrangian in Bertsekas method for the inequality-constrained regularized systems of linear equations problem (261) takes the form:

$$\begin{aligned} L_\rho(x, x^s, \mu^+, \mu^-) = \sum_{j\in P} \Bigg\{ & x_j^2 + \frac{\rho}{2}(x_j - x_j^s)^2 \\ & + \sum_{i\in M} \left(\mu_i^+ \left(\sum_{j\in P} A_{ij}x_j - y_i \right) + \mu_i^- \left(-\sum_{j\in P} A_{ij}x_j + (1+\varepsilon)y_i \right) \right) \Bigg\} \end{aligned} \tag{262}$$

This expression can be decomposed into coordinate-wise subproblems as:

$$L_{\rho_j}(x_j, x_j^s, \mu^+, \mu^-) = x_j^2 + \frac{\rho}{2}(x_j - x_j^s)^2 + \sum_{i\in M} A_{ij}x_j(\mu_i^+ - \mu_i^-) \tag{263}$$

At each iteration $k+1$, using a relaxation factor $\xi_k \in [0,1)$, the updates proceed as follows:

$$x_j^{k+1} = \arg\min_{x_j} L_{\rho_j}(x_j, x_j^{s^k}, \mu^{+,k}, \mu^{-,k}), \quad \forall j \in P \tag{264}$$

$$x_j^{s^{k+1}} = \xi_k x_j^{s^k} + (1-\xi_k) x_j^{k+1}, \quad \forall j \in P \tag{265}$$

$$\mu_i^{+,k+1} = \max\left\{0, \mu_i^{+,k} + \beta\rho\left(\sum_{j\in P} A_{ij} x_j^{k+1} - y_i\right)\right\}, \quad \forall i \in M \tag{266}$$

$$\mu_i^{-,k+1} = \max\left\{0, \mu_i^{-,k} + \beta\rho\left(-\sum_{j\in P} A_{ij} x_j^{k+1} + (1+\varepsilon) y_i\right)\right\}, \quad \forall i \in M \tag{267}$$

This formulation ensures that both upper and lower inequality constraints are respected by maintaining nonnegative dual multipliers μ^+ and μ^-, associated with the two-sided inequalities in the reformulated problem.

8.2. Asynchronous Iterative Updates

Let $S_j \subseteq \{1,2,\dots\}$ be the set of iterations at which variable j updates x_j and x_j^s based on the current estimates of μ_i^+, μ_i^- for all $i \in M$. Similarly, let $T_i \subseteq \{1,2,\dots\}$ be the set of iterations at which constraint i updates μ_i^+ and μ_i^- based on the most recent estimates of x_j for all $j \in P$. Using an approach inspired by [55] and the augmented Lagrangian function defined in (262), the asynchronous updates proceed as follows.

j's Algorithm: At iterations $k = 1,2,\dots$, for each $j \in P$:

1: From iteration to iteration, j receives updates $\mu_i^{+,k}$ and $\mu_i^{-,k}$ from all $i \in M$.

2: At each update iteration $k' \in S_j$, j chooses the next x_j^{k+1} and $x_j^{s^{k+1}}$:

$$x_j^{k+1} = \arg\min_{x_j} L_{\rho_j}(x_j, x_j^{s^k}, \hat{\mu}^{+,k}, \hat{\mu}^{-,k}), \quad \forall j \in P \tag{268}$$

$$x_j^{s^{k+1}} = \xi_k x_j^{s^k} + (1-\xi_k) x_j^{k+1}, \quad \forall j \in P \tag{269}$$

3: Transmission continues at this rate until the next update.

4: j communicates x_j^{k+1} to all constraint controllers.

i's Algorithm: At iterations $k = 1,2,\dots$, for each $i \in M$:

1: From iteration to iteration, i receives updates x_j^{k+1} from all $j \in P$.

2: At each update iteration $k' \in T_i$, i updates μ_i^+ and μ_i^- as follows:

$$\mu_i^{+,k+1} = \max\left\{0,\ \mu_i^{+,k} + \beta\rho\left(\sum_{j\in P} A_{ij}\hat{x}_j^{k+1} - y_i\right)\right\} \tag{270}$$

$$\mu_i^{-,k+1} = \max\left\{0,\ \mu_i^{-,k} + \beta\rho\left(-\sum_{j\in P} A_{ij}\hat{x}_j^{k+1} + (1+\varepsilon)y_i\right)\right\} \tag{271}$$

3: Transmission continues at this rate until the next update.

4: i communicates $\mu_i^{+,k+1}$ and $\mu_i^{-,k+1}$ to all local optimizers.

Stop when $x_j^{k+1} = x_j^{s^{k+1}}$ for all $j \in P$; otherwise, increment k and repeat.

The asynchronously aggregated values used in the updates are defined as:

$$\hat{x}_j^k = \sum_{k'=k-k_0}^{k} a_j(k',k)x_j^{k'}, \quad \forall j \in P \tag{272}$$

$$\hat{\mu}_i^{+,k} = \sum_{k'=k-k_0}^{k} b_i(k',k)\mu_i^{+,k'}, \quad \forall i \in M \tag{273}$$

$$\hat{\mu}_i^{-,k} = \sum_{k'=k-k_0}^{k} b_i(k',k)\mu_i^{-,k'}, \quad \forall i \in M \tag{274}$$

with the weights satisfying:

$$\sum_{k'=k-k_0}^{k} a_j(k',k) = 1, \quad a_j(k',k) \geq 0, \quad \forall k,\ j \in P \tag{275}$$

$$\sum_{k'=k-k_0}^{k} b_i(k',k) = 1, \quad b_i(k',k) \geq 0, \quad \forall k,\ i \in M \tag{276}$$

8.3. Numerical Tests

All dataset details used are as described in Section 7.2.1.

8.3.1. Implementation Details

All implementations and numerical experiments were conducted using Python 3.12.0. The optimization models were formulated with the Pyomo modeling framework and solved using the Gurobi optimizer. Computational experiments were performed on a machine equipped

with an AMD Ryzen 5 4600H processor (3.00 GHz, Radeon Graphics), 32 GB of RAM, and a 512 GB SSD, running a 64-bit Windows 10 Pro operating system.

The individual results were first aligned to a unified time axis to enable consistent comparison across datasets with potentially different time indices. A comprehensive timeline was constructed by taking the union of all time points in the datasets. Each dataset was then merged onto this unified timeline using nearest-neighbor matching via an as-of merge (***pandas.merge_asof*** *performs a merge by nearest key rather than exact matches, aligning rows based on the closest preceding key in a sorted dataset. It is especially useful for time-series data to join on nearest timestamps without requiring exact equality*), ensuring that for each time point in the reference axis, the closest available record from each dataset was selected. This approach preserves temporal coherence while allowing synchronized evaluation of multiple time series, even in non-uniform or asynchronous sampling intervals.

Both the synchronous and asynchronous variants of Bertsekas method were implemented using Python's *multiprocessing* module. In the synchronous version, a *Pool* of worker processes is used to update partitions of the primal variable in parallel. All updates are synchronized at each iteration before the dual variable is updated, ensuring a consistent global state. While this design is simple and deterministic, it suffers from synchronization delays due to straggler processes, especially under partitioned execution.

In contrast, the asynchronous variant eliminates global barriers by allowing each worker to proceed independently using bounded-delay averaging. Shared memory buffers (*multiprocessing.shared_memory*) and *locks* coordinate access to global variables, while a shared counter tracks progress across tasks. Primal and dual workers run in parallel without waiting, exchanging delayed but consistent updates through shared state. This design improves runtime efficiency, particularly in large-scale or heterogeneous environments.

Both variants periodically log convergence metrics (e.g., norm differences, objective values, constraint violations), and checkpoints are saved for analysis and reproducibility. Convergence is detected via multiple stopping criteria, including change in objective, constraint satisfaction, and relative error.

8.3.2. Results and Discussion

Table 8.1, Fig 8.1, Fig 8.3, Fig 8.4, Fig 8.5, and Fig 8.2 report the performance of the synchronous and asynchronous variants of the Bertsekas algorithm on three regularized linear systems: one derived from clinical data (Cancer) and two from synthetic image recovery tasks with uniform and Gaussian diagonal measurement matrices. All experiments were conducted using 12 data partitions to reflect realistic distributed processing conditions.

Across all three datasets, both synchronous and asynchronous algorithms achieved the

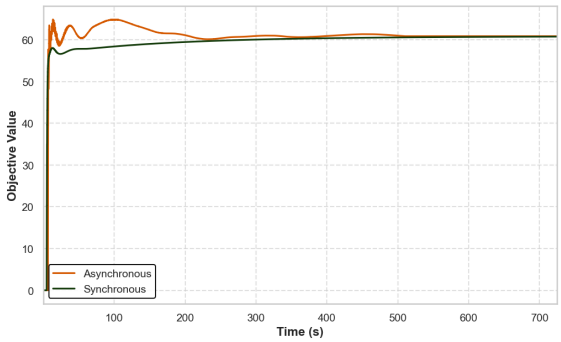
same objective values, indicating convergence to consistent solutions. For the synthetic image-based systems, the objective values were 60.87 and 61.41 for the uniform and Gaussian matrices, respectively. The Cancer dataset yielded a lower final objective value of 54.61, likely due to its lower dimensionality and better numerical conditioning compared to the randomized sensing matrices used in the synthetic cases. The consistency in solution quality across all runs confirms that asynchronous updates do not compromise the convergence accuracy of the algorithm, even in ill-conditioned or noisy settings.

In terms of runtime, the asynchronous variant demonstrated clear computational advantages. On the low-dimensional Cancer dataset, it reduced the execution time from 55 seconds (synchronous) to just 8 seconds, yielding a nearly seven-fold speedup. This is particularly relevant in real-world scenarios involving clinical diagnostics, where quick and reliable inference is essential. For the Gaussian diagonal sensing matrix, the asynchronous version reduced runtime from 724 to 511 seconds—approximately a 30% improvement. While the gain was less dramatic in the uniform matrix case (938 vs. 918 seconds), the asynchronous approach still maintained its efficiency without introducing instability.

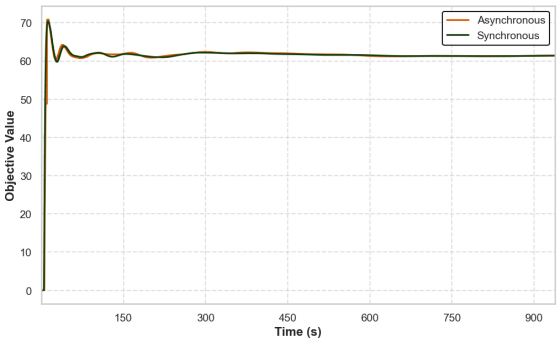
Overall, all configurations resulted in successful convergence. The asynchronous scheme consistently matched the solution quality of its synchronous counterpart while significantly reducing computation time in most cases. These results suggest that asynchronous distributed optimization is especially effective in scenarios involving low-dimensional or poorly conditioned systems, where it can deliver substantial efficiency gains without sacrificing solution accuracy.

Table 8.1. Performance Comparison of Optimization Algorithms for Regularized Systems of Linear Equations Across Different Problem Regimes with 12 Partitions

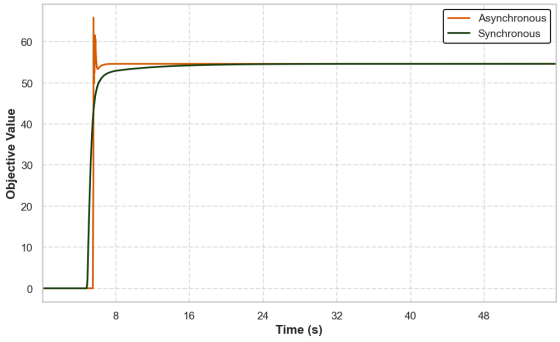
	Synchronous			**Asynchronous**		
Dataset	**Objective**	Time(s)	Status	**Objective**	Time(s)	Status
Gaussian	61.41	724	Converged	61.41	511	Converged
Uniform	60.87	938	Converged	60.87	918	Converged
Cancer	54.61	55	Converged	54.61	8	Converged



(a) Uniform



(b) Gaussian



(c) Cancer Spars

Figure 8.1. Async-Sync: Objectives

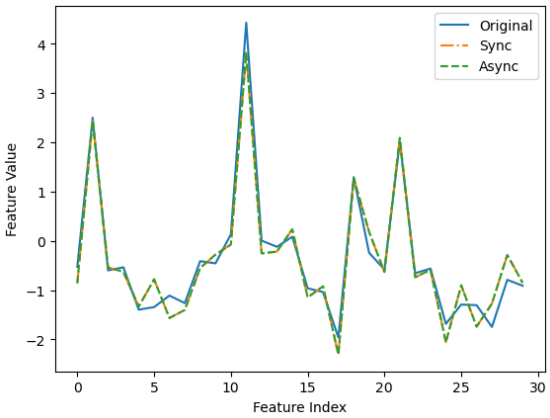


Figure 8.2. Async-Sync: Compressed Measurements of Cancer Spars

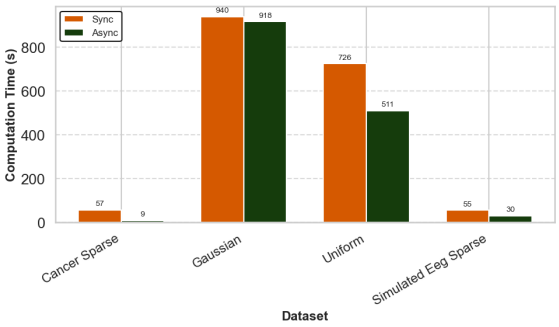
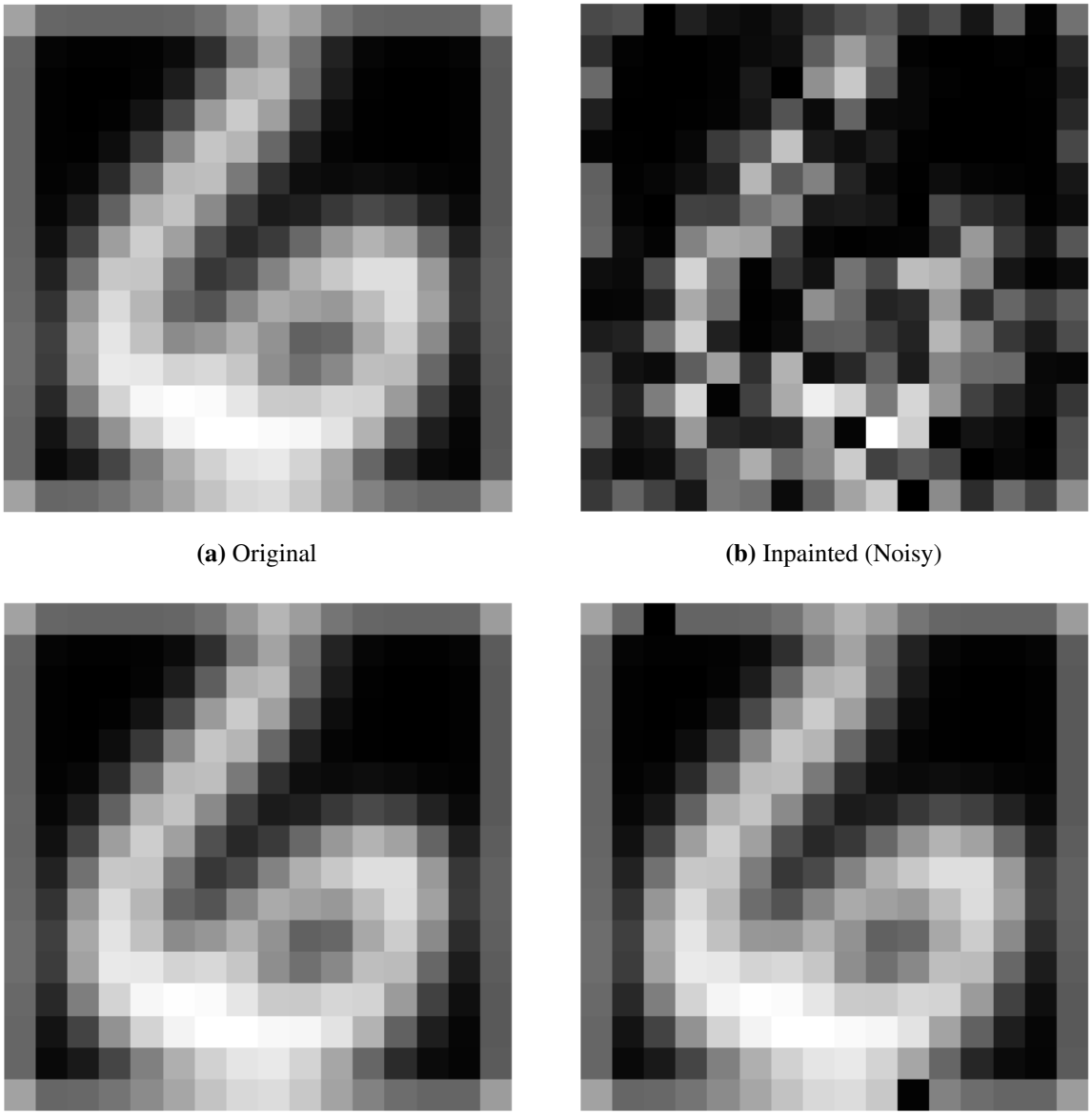


Figure 8.3. Async-Sync: Run Time (secs)



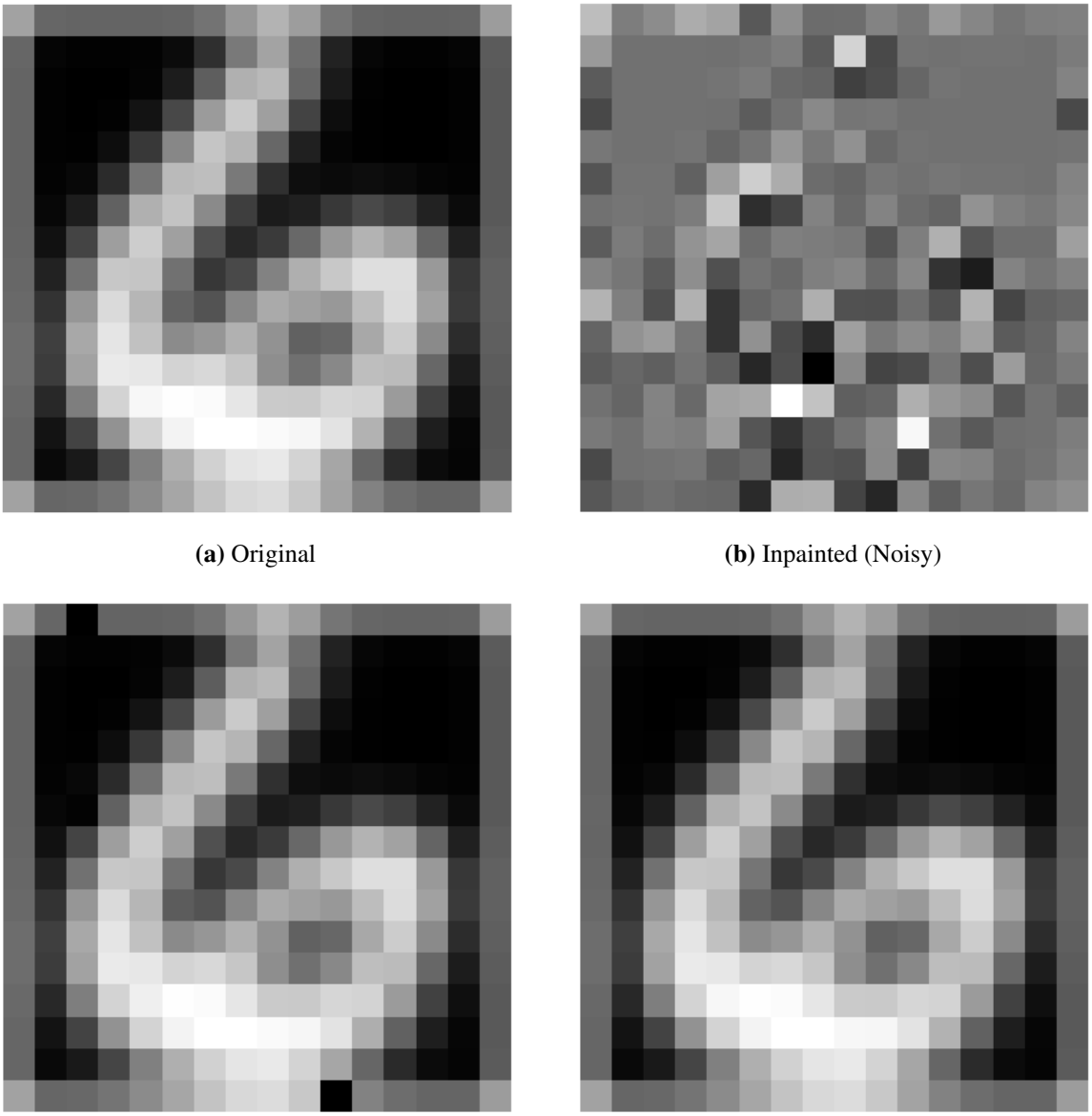
(a) Original

(b) Inpainted (Noisy)

(c) Synchronous

(d) Asynchronous

Figure 8.4. Image Recovery under Uniform Additive Noise



(a) Original

(b) Inpainted (Noisy)

(c) Synchronous

(d) Asynchronous

Figure 8.5. Image Recovery under Gaussian Additive Noise

9. K-Means Clustering with Augmented Lagrangian Algorithms

In various fields such as data mining, machine learning, image analysis, and bioinformatics, the need to uncover interpretable and computationally efficient patterns in large datasets has driven the study of *clustering methods*. Among these, K-means clustering is one of the most widely used techniques for partitioning data into homogeneous groups. The primary objective is to identify cluster centers that minimize intra-cluster variation, yielding a compact data representation.

K-means clustering is a technique to divide a dataset into n clusters. Given a dataset $A = \left[A_{iq}\right]_{m\times Q}$ containing m samples in $\mathbb{R}^Q$, the objective is to identify n cluster centers $\phi_1, \phi_2, \ldots, \phi_n$, such that the total squared distance between each point A_i and the closest cluster center ϕ_j is minimized. This can be formulated as:

$$\min_{x,\phi} \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} \left(\frac{1}{2} \sum_{q=1}^{Q} \left(A_{iq} - \phi_{jq}\right)^2 \right) \tag{277}$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, m, \tag{278}$$

$$x_{ij} \in \{0,1\}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \tag{279}$$

The algorithm proceeds iteratively with two main steps: first, each point A_i is assigned to the nearest cluster, then, the cluster centers ϕ_j are updated as the mean of the points assigned to each cluster. This process repeats until convergence.

However, K-Means can sometimes produce empty or very small clusters, especially when applied to high-dimensional datasets. To address this, a constrained version of K-Means was introduced in [59]. This approach modifies the optimization problem by adding constraints that ensure each cluster has at least τ_j points. The paper also suggested solving (277)-(279) in x for fixed ϕ, then solving (277) in ϕ for fixed x. The new optimization problem is defined as:

At iteration $k+1$:

- **Cluster Assignment**: Let x_{ij}^k be a solution to the following problem with ϕ_j^k fixed.

$$\min_x \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} \left(\frac{1}{2} \sum_{q=1}^{Q} \left(A_{iq} - \phi_{jq}^k \right)^2 \right) \tag{280}$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, m, \tag{281}$$

$$\sum_{i=1}^{m} x_{ij} \geq \tau_j, \quad j = 1, \ldots, n, \tag{282}$$

$$x_{ij} \in \{0,1\}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \tag{283}$$

- **Cluster Center Update**: The cluster centers are updated as:

$$\phi_{jq}^{k+1} = \begin{cases} \frac{\sum_{i=1}^{m} x_{ij}^k A_{iq}}{\sum_{i=1}^{m} x_{ij}^k}, & \text{if } \sum_{i=1}^{m} x_{ij}^k > 0 \\ \phi_{jq}^k, & \text{otherwise} \end{cases}, \quad \forall j = 1, \ldots, n; q = 1, \ldots, Q \tag{284}$$

Stop when $\phi_{jq}^{k+1} = \phi_{jq}^k$ for all j, q, else increment k by 1 and go to step 1.

The challenge with the algorithm is that it is impossible to decompose it due to the coupling constraint, $\sum_{i=1}^{m} x_{ij} \geq \tau_j$. To overcome this, the Bertsekas Decomposition method is used. It will be convenient to introduce for every training example i the admissible set X_i resulting from the equations (281), (283). This set for each training example i will be defined as:

$$X_i = \left\{ \begin{array}{l} \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, m \\ \\ x_{ij} \in \{0,1\}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \end{array} \right\} \tag{285}$$

9.1. Decomposition of the Constrained K-Means Problem

The constrained K-Means clustering problem, as formulated in (280)–(283), admits a structure that is well-suited for decomposition techniques, particularly in the context of large-scale and distributed optimization. While the objective function is partially separable across data points, the assignment constraints introduce coupling through the cardinality conditions $\sum_{i=1}^{m} x_{ij} \geq \tau_j$. This blend of separability and structured coupling motivates using augmented Lagrangian and dual decomposition methods to solve the resulting mixed-integer optimization problem efficiently.

To address the coupling constraints and enable scalable computation, consider a family

of decomposition strategies rooted in classical and modern dual optimization frameworks. These include the Multiplier Method, the Bertsekas Decomposition technique, the Tatjewski formulation and the SALA (Separated Augmented Lagrangian Algorithm) variant of the Alternating Direction Method of Multipliers (ADMM). Each approach introduces dual variables associated with the cluster-level constraints and iteratively updates the primal variables x and the cluster centers ϕ while coordinating through dual ascent or primal-dual updates. The augmented Lagrangian terms serve both to penalize constraint violations and to improve convergence stability.

These decomposition techniques are especially attractive in distributed environments where data points are stored across multiple compute nodes. By decoupling subproblems, typically across data samples, they enable parallel updates of the assignment variables, subject to coordination via dual variables.

9.1.1. The Lagrangian

The Lagrangian of Problem (280)-(283) can be written as

$$L(x,\phi,\mu) = \sum_{i\in M}\sum_{j\in P}\left[\sum_{q\in Q} x_{ij}\left(a_{iq}-\phi_{jq}\right)^2 + \mu_j\Big(\frac{\tau}{|M|}-x_{ij}\Big)\right] = \sum_{i\in M}\sum_{j\in P} L_{ij}(x_{ij},\phi_j,\mu_j) \quad (286)$$

where

$$L_{ij}(x_{ij},\phi_j,\mu_j) = \sum_{q\in Q} x_{ij}\left(a_{iq}-\phi_{jq}\right)^2 + \mu_j\Big(\frac{\tau}{|M|}-x_{ij}\Big) \quad (287)$$

At iteration $k+1$, with $\mu_j^k \in R$, and $\rho > 0$,

$$x_i^{k+1} = \arg\min_{x_i\in X_i} \sum_{j\in P} L_{ij}(x_{ij},\phi_j^k,\mu_j^k) \quad (288)$$

$$\phi_{jq}^{k+1} = \begin{cases} \frac{\sum_{i=1}^m x_{ij}^{k+1} a_{iq}}{\sum_{i=1}^m x_{ij}^{k+1}}, & \text{if } \sum_{i=1}^m x_{ij}^{k+1} > 0 \\ \phi_{jq}^k, & \text{otherwise} \end{cases}, \quad \forall j\in P; q\in Q \quad (289)$$

$$\mu_j^{k+1} = \max\left\{0, \mu_j^k + \rho\Big(\tau - \sum_{i\in M} x_{ij}\Big)\right\}, \quad \forall j\in P \quad (290)$$

9.1.2. The Multiplier Method

The Augmented Lagrangian of Problem (280)-(283) can be written as

$$L_\rho(x,\phi,\mu) = \sum_{j\in P}\left[\sum_{i\in M}\sum_{q\in Q} x_{ij}\left(a_{iq}-\phi_{jq}\right)^2 + \mu_j\Big(\tau_j - \sum_{i\in M} x_{ij}\Big)\right] + \frac{\rho}{2}\sum_{j\in P}\Big(\tau_j - \sum_{i\in M} x_{ij}\Big)^2 \tag{291}$$

At iteration $k+1$,

$$x^{k+1} = \arg\min_{x_i\in X_i,\forall i\in M} L_\rho(x,\phi^k,\mu^k) \tag{292}$$

$$\phi_{jq}^{k+1} = \begin{cases} \frac{\sum_{i=1}^m x_{ij}^{k+1} a_{iq}}{\sum_{i=1}^m x_{ij}^{k+1}}, & \text{if } \sum_{i=1}^m x_{ij}^{k+1} > 0 \\ \phi_{jq}^k, & \text{otherwise} \end{cases}, \quad \forall j\in P; q\in Q \tag{293}$$

$$\mu_j^{k+1} = \max\left\{0, \mu_j^k + \rho\Big(\tau - \sum_{i\in M} x_{ij}\Big)\right\}, \quad \forall j\in P \tag{294}$$

9.1.3. The Bertsekas Method

The Bertsekas Augmented Lagrangian method for problem (280)-(283) has this form:

$$\begin{aligned} L_\rho(x,\phi,x^s,\phi^s,\mu) &= \sum_{i\in M}\sum_{j\in P}\left\{\sum_{q\in Q} x_{ij}\left(a_{iq}-\phi_{jq}\right)^2 + \mu_j\Big(\frac{\tau}{|M|} - x_{ij}\Big)\right. \\ &\quad \left. + \frac{\rho}{2}\left[\left(x_{ij}-x_{ij}^s\right)^2 + \frac{1}{|M|}\sum_{q\in Q}\left(\phi_{jq}-\phi_{jq}^s\right)^2\right]\right\} \\ &= \sum_{i\in M}\sum_{j\in P} L_{\rho_{ij}}(x_{ij},\phi_j,x_{ij}^s,\mu_j) \end{aligned} \tag{295}$$

where

$$L_{\rho_{ij}}(x_{ij},\phi_j,x_{ij}^s,\mu_j) = \sum_{q\in Q} x_{ij}\left(a_{iq}-\phi_{jq}\right)^2 + \mu_j\Big(\frac{\tau}{|M|} - x_{ij}\Big) + \frac{\rho}{2}\left(x_{ij}-x_{ij}^s\right)^2 \tag{296}$$

At iteration $k+1$, with $x_{ij}^s \in \{0,1\}$, $\phi_j^s \in R$,

$$x_i^{k+1} = \arg\min_{x_i \in X_i} \sum_{j \in P} L_{\rho_{ij}}(x_{ij}, \phi_j^k, x_{ij}^{s^k}, \mu_j^k), \quad \forall i \in M \tag{297}$$

$$\phi_{jq}^{k+1} = \begin{cases} \frac{\rho \phi_{jq}^k + \sum_{i=1}^m x_{ij}^{k+1} a_{iq}}{\rho + \sum_{i=1}^m x_{ij}^{k+1}}, & \text{if } \sum_{i=1}^m x_{ij}^{k+1} > 0 \\ \phi_{jq}^k, & \text{otherwise} \end{cases}, \quad \forall j \in P; q \in Q \tag{298}$$

$$x_{ij}^{s^{k+1}} = \xi_k x_{ij}^{s^k} + (1-\xi_k) x_{ij}^{k+1}, \quad \forall i \in M; j \in P \tag{299}$$

$$\phi_{jq}^{s^{k+1}} = \xi_k \phi_{jq}^{s^k} + (1-\xi_k) \phi_{jq}^{k+1}, \quad \forall i \in M; j \in P \tag{300}$$

$$\mu_j^{k+1} = \max\left\{0, \mu_j^k + \rho\left(\tau - \sum_{i \in M} x_{ij}\right)\right\}, \quad \forall j \in P \tag{301}$$

9.1.4. The Tatjewski Method

The Tatjewski Augmented Lagrangian method for Problem (280)-(283) has this form

$$L_\rho(x, \phi, x^s, \mu) = \sum_{i \in M} \sum_{j \in P} \left\{ \sum_{q \in Q} x_{ij} \left(a_{iq} - \phi_{jq}\right)^2 + \mu_j \left(\frac{\tau}{|M|} - x_{ij}\right) + \frac{\rho}{2}\left(\tau_j - x_{ij} - \sum_{l \in M, l \neq i} x_{lj}^s\right)^2 \right\} = \sum_{i \in M} \sum_{j \in P} L_{\rho_{ij}}(x_{ij}, \phi_j, x_{ij}^s, \mu_j) \tag{302}$$

where

$$L_{\rho_{ij}}(x_{ij}, \phi_j, x_{ij}^s, \mu_j) = \sum_{q \in Q} x_{ij} \left(a_{iq} - \phi_{jq}\right)^2 + \mu_j \left(\frac{\tau}{|M|} - x_{ij}\right) + \frac{\rho}{2}\left(\tau_j - x_{ij} - \sum_{l \in M, l \neq i} x_{lj}^s\right)^2 \tag{303}$$

At iteration $k+1$,

$$x_i^{k+1} = \arg\min_{x_i \in X_i} \sum_{j \in P} L_{\rho_{ij}}(x_{ij}, \phi_j^k, x_{ij}^{s^k}, \mu_j^k), \quad \forall i \in M \tag{304}$$

$$\phi_{jq}^{k+1} = \begin{cases} \frac{\sum_{i=1}^m x_{ij}^{k+1} a_{iq}}{\sum_{i=1}^m x_{ij}^{k+1}}, & \text{if } \sum_{i=1}^m x_{ij}^{k+1} > 0 \\ \phi_{jq}^k, & \text{otherwise} \end{cases}, \quad \forall j \in P; q \in Q \tag{305}$$

$$x_{ij}^{s^{k+1}} = \xi_k x_{ij}^{s^k} + (1-\xi_k) x_{ij}^{k+1}, \quad \forall i \in M; j \in P \tag{306}$$

$$\mu_j^{k+1} = \max\left\{0, \mu_j^k + \rho\left(\tau - \sum_{i \in M} x_{ij}\right)\right\}, \quad \forall j \in P \tag{307}$$

9.1.5. The ADMM (SALA Version)

Adapting ADMM SALA (50)-(54) to solve Problem (280)-(283), we have,

$$\begin{aligned} L_\rho(x,\phi,s,\mu) &= \sum_{i\in M}\sum_{j\in P}\Bigg[\sum_{q\in Q} x_{ij}\left(a_{iq}-\phi_{jq}\right)^2+\mu_j\Big(\frac{\tau}{|M|}-x_{ij}-s_{ij}\Big) \\ &\quad +\frac{\rho}{2}\Big(\frac{\tau}{|M|}-x_{ij}-s_{ij}\Big)^2\Bigg] \\ &= \sum_{i\in M}\sum_{j\in P} L_{\rho_{ij}}(x_{ij},\phi_j,s_{ij},\mu_j) \end{aligned} \tag{308}$$

where

$$L_{\rho_{ij}}(x_{ij},\phi_j,s_{ij},\mu_j)=\sum_{q\in Q} x_{ij}\left(a_{iq}-\phi_{jq}\right)^2+\mu_j\Big(\frac{\tau}{|M|}-x_{ij}-s_{ij}\Big)+\frac{\rho}{2}\Big(\frac{\tau}{|M|}-x_{ij}-s_{ij}\Big)^2 \tag{309}$$

At iteration $k+1$,

$$x_i^{k+1}=\arg\min_{x_i\in X_i}\sum_{j\in P} L_{\rho_{ij}}(x_{ij},\phi_j^k,s_{ij}^k,\mu_j^k) \tag{310}$$

$$s_{ij}^{k+1}=\frac{\tau}{|M|}-x_{ij}^{k+1}-\frac{\mu_j^{k+1}}{\rho},\quad \forall j\in P, i\in M \tag{311}$$

$$\phi_{jq}^{k+1}=\begin{cases}\frac{\sum_{i=1}^m x_{ij}^{k+1}a_{iq}}{\sum_{i=1}^m x_{ij}^{k+1}}, & \text{if } \sum_{i=1}^m x_{ij}^{k+1}>0\\ \phi_{jq}^k, & \text{otherwise}\end{cases},\quad \forall j\in P; q\in Q \tag{312}$$

$$\mu_j^{k+1}=\max\left\{0,\mu_j^k+\rho\Big(\tau-\sum_{i\in M}x_{ij}\Big)\right\},\quad \forall j\in P \tag{313}$$

9.2. Numerical Tests

All implementation details used are as described in Section 8.3.1.

9.2.1. Dataset Description

To evaluate the performance and generalizability of the proposed algorithms, experiments were conducted on both synthetic and real-world datasets representing a diverse range of clustering challenges. All datasets were processed to ensure comparability and reproducibility, with dimensionality reduction applied where appropriate.

Four synthetic datasets were created using **make_blobs** from the Scikit-learn library [60], which generates isotropic Gaussian clusters commonly used for clustering evaluation. The

datasets vary in the number of samples, features, and clusters to simulate different levels of complexity as shown below:

- **High-Dimensional Multi-Cluster (Synthetic-HD-MC)**: 1,000 samples with 100 features grouped into 15 clusters.
- **Low-Dimensional Few-Cluster (Synthetic-LD-FC)**: 5,000 samples with 10 features and 5 clusters.
- **2D Multi-Cluster Visualization Set (Synthetic-2D-MC)**: 1,000 samples in 2D space with 15 clusters, suitable for visual inspection.
- **2D Few-Cluster Visualization Set (Synthetic-2D-FC)**: 5,000 samples in 2D space with 5 clusters.

These synthetic datasets help assess clustering performance under controlled distributions, with cluster separability influenced by feature dimensionality and the number of clusters.

Two real-world datasets, **ISIC 2019** and **MedQuAD**, were also used to test the algorithms in more practical, noisy scenarios.

The ISIC 2019 dataset [61]–[63], the most widely available publicly available collection of quality-controlled dermatology skin images, was used to test clustering models with image data. The dataset contains 25331 dermoscopic images of skin lesions, each associated with ground-truth diagnoses (benign, malignant) and clinical metadata. These standardized images provided a diverse and high-quality foundation for generating features customized to clustering tasks in the medical imaging domain.

To prepare the images for clustering, they were converted to grayscale to simplify the data while retaining essential visual features. Each image was then resized for uniformity, converted into arrays, and flattened into one-dimensional vectors. These flattened arrays served as the feature set for the clustering models. By preprocessing ISIC 2019 images this way, the clustering analysis could focus on the underlying patterns and relationships in the data, enabling a robust comparison of clustering approaches across this rich medical imaging dataset.

- **ISIC Lesion Embeddings (ISIC-PCA10)**: Derived from the ISIC 2019 Challenge dataset [61]–[63], a publicly available collection of quality-controlled dermatology skin images. 2,000 image feature vectors were selected, reduced to 10 principal components.
- **ISIC Low-Dimensional Variant (ISIC-PCA2)**: The same set as above but reduced to 2 PCA components, creating a more compressed representation to test performance under extreme dimensionality reduction.
- **MedQuAD QA Representations (MedQA-PCA20)**: Based on the MedQuAD dataset [64], which contains medically relevant question-answer pairs. To generate the features for the clustering problem, each row of the "answers" column of the MedQuAD dataset

was processed using BioClinicalBERT [65], a pre-trained language model explicitly designed for clinical text to create numerical embeddings. These embeddings, which capture the semantic essence of the text, were then used as the feature set for the clustering models. 1,000 instances were sampled, and PCA was applied to reduce the feature space to 20 components.

- **MedQuAD Minimal Representation (MedQA-PCA2)**: A simplified variant with only 2 PCA components to evaluate performance in a very low-dimensional semantic space.

The ISIC dataset serves as a representative for medical image data, while MedQuAD captures structured medical text, providing a rich testbed for evaluating clustering robustness across modalities and dimensionalities.

9.2.2. Results and Discussion

Tables 9.1 and 9.2 summarize the performance of four optimization algorithms for K-means clustering on synthetic and real-world datasets.

Synthetic Datasets

Across all synthetic datasets, the algorithms converged to identical objective values and clustering scores, confirming the correctness and equivalence of the underlying formulations. However, notable differences emerged in computational efficiency:

- **Bertsekas algorithm** consistently exhibited the **best overall runtime**, achieving convergence with low computational cost across all datasets and settings. Its lightweight updates and rapid convergence make it particularly suitable for both low- and high-dimensional clustering tasks.
- **ADMM** performed competitively, with slightly higher runtimes than Bertsekas, but showed consistent performance and reliable convergence. It also benefited from partitioning, which further reduced runtime (e.g., from 43s to 31s on Synthetic-2D-FC).
- **Tatjewski's method**, while converging reliably, was substantially slower in the non-partitioned setting. However, partitioning significantly improved its efficiency (e.g., a 5× speedup on Synthetic-2D-MC), showing its potential scalability in distributed environments.
- **Augmented Lagrangian** approach struggled with scalability. Although it converged in smaller datasets, it frequently timed out on partitioned settings, especially where synchronization overhead was high. This highlights its limited practicality in parallelized scenarios.

Real-World Datasets

The findings on real-world data mirrored those on synthetic data, though the impact of dimensionality reduction (via PCA) made performance distinctions more pronounced:

- **Bertsekas' algorithm** again stood out for its speed, particularly on MedQA-PCA2 and ISIC-PCA2, where it completed in under 25 seconds while maintaining optimal objective values and clustering scores.
- **ADMM** also remained robust, striking a good balance between runtime and reliability. It scaled well with data complexity, converging even on larger datasets like ISIC-PCA10 with reduced runtime when partitioned (107s $\rightarrow$ 50s).
- **Tatjewski** benefited notably from partitioning (e.g., 1699s $\rightarrow$ 385s on ISIC-PCA10), reinforcing its value in distributed systems despite its relatively heavier compute cost per iteration.
- **Augmented Lagrangian** again proved inefficient for high-dimensional or partitioned data. It either failed to converge within the time budget or offered no computational advantage over simpler alternatives.

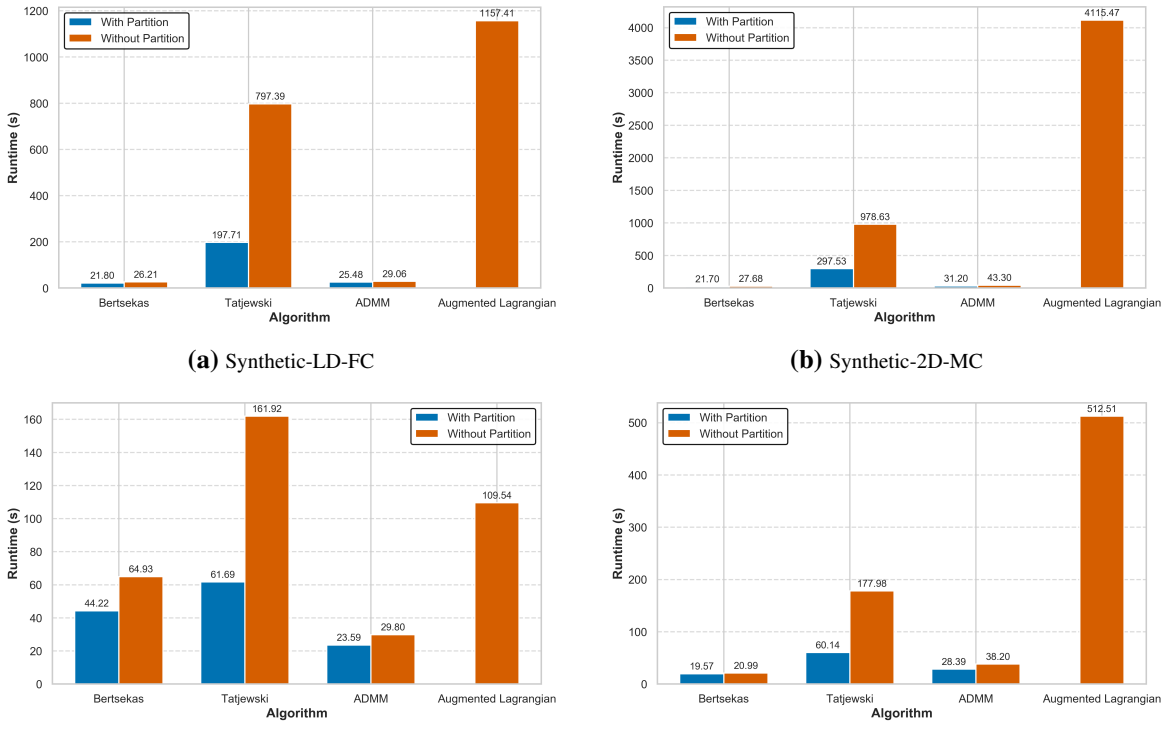
Overall, **Bertsekas' method emerged as the most efficient**, delivering fast and reliable convergence across all datasets and settings. **ADMM** was a strong second, offering consistent performance and good scalability. **Tatjewski** lagged in speed but benefited markedly from partitioning, making it viable for parallel environments. In contrast, the **Augmented Lagrangian** method showed limited utility due to its sensitivity to problem decomposition and coordination overhead.

Table 9.1. Performance Comparison of Optimization Algorithms for K-Means Clustering on Synthetic Datasets

		No Partition				**8 Partitions**			
Dataset	**Algorithm**	**Objective**	Score	**Time**(s)	Status	**Objective**	Score	**Time**(s)	Status
Synthetic-LD-FC	ADMM	24807.09	0.74	29	Converged	24807.09	0.74	25	Converged
	Bertsekas	24807.09	0.74	26	Converged	24807.09	0.74	21	Converged
	Tatjewski	24807.09	0.74	797	Converged	24807.09	0.74	197	Converged
	Aug Lagrangian	24807.09	0.74	1157	Converged	-	-	-	-
Synthetic-HD-MC	ADMM	346173.75	0.55	29	Converged	346173.75	0.55	23	Converged
	Bertsekas	346173.75	0.55	64	Converged	346173.75	0.55	44	Converged
	Tatjewski	346173.75	0.55	161	Converged	346173.75	0.55	61	Converged
	Aug Lagrangian	346173.75	0.55	109	Converged	-	-	-	-
Synthetic-2D-MC	ADMM	1064.55	0.45	38	Converged	1064.55	0.45	28	Converged
	Bertsekas	1064.55	0.45	20	Converged	1064.55	0.45	19	Converged
	Tatjewski	1064.55	0.45	177	Converged	1064.55	0.45	60	Converged
	Aug Lagrangian	1064.55	0.45	512	Converged	-	-	-	-
Synthetic-2D-FC	ADMM	4505.94	0.55	43	Converged	4505.94	0.55	31	Converged
	Bertsekas	4505.94	0.55	27	Converged	4505.94	0.55	21	Converged
	Tatjewski	4505.94	0.55	978	Converged	4505.94	0.55	297	Converged
	Aug Lagrangian	4506.03	0.55	4115	Time Out	-	-	-	-

Table 9.2. Performance Comparison of Optimization Algorithms for K-Means Clustering on Real World Datasets

Dataset	Algorithm	No Partition				8 Partitions			
		Objective	Score	**Time**(s)	Status	**Objective**	Score	**Time**(s)	Status
ISIC-PCA10	ADMM	234036414.61	0.17	107	Converged	234036414.61	0.17	50	Converged
	Bertsekas	234036414.61	0.17	88	Converged	234036414.61	0.17	61	Converged
	Tatjewski	234036414.61	0.17	1699	Converged	234036414.61	0.17	385	Converged
	Aug Lagrangian	237761989.80	0.17	4108	Time Out	-	-	-	-
ISIC-PCA2	ADMM	80010359.93	0.37	56	Converged	80010359.93	0.37	35	Converged
	Bertsekas	80010359.93	0.37	42	Converged	80010359.93	0.37	25	Converged
	Tatjewski	80010359.93	0.37	859	Converged	80010359.93	0.37	200	Converged
	Aug Lagrangian	80064769.15	0.37	4060	Time Out	-	-	-	-
MedQA-PCA20	ADMM	1289.70	0.14	251	Converged	1289.70	0.14	74	Converged
	Bertsekas	1289.70	0.14	35	Converged	1289.70	0.14	24	Converged
	Tatjewski	1289.70	0.14	251	Converged	1289.70	0.14	74	Converged
	Aug Lagrangian	1289.70	0.14	708	Converged	-	-	-	-
MedQA-PCA2	ADMM	57.95	0.36	55	Converged	57.95	0.36	33	Converged
	Bertsekas	57.95	0.36	36	Converged	57.95	0.36	23	Converged
	Tatjewski	57.95	0.36	715	Converged	57.95	0.36	103	Converged
	Aug Lagrangian	57.95	0.36	1258	Converged	-	-	-	-



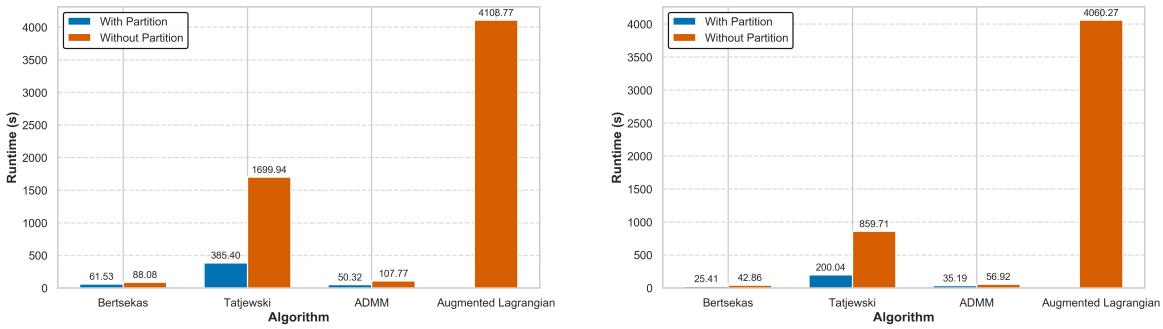
(a) Synthetic-LD-FC

(b) Synthetic-2D-MC

(c) Synthetic-HD-MC

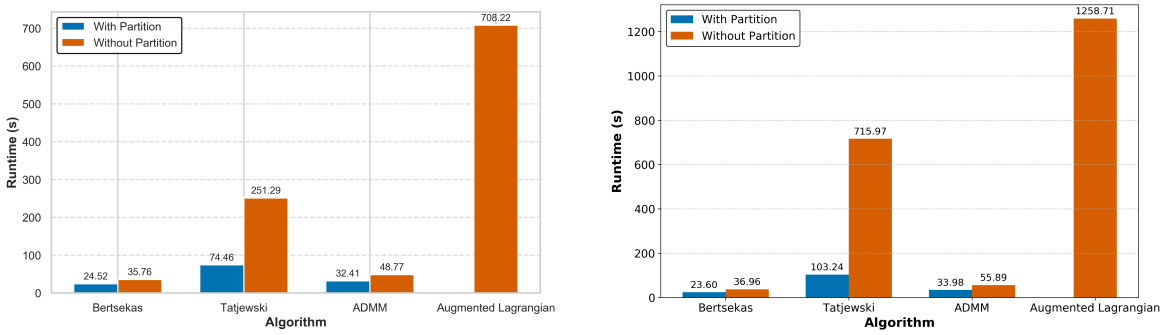
(d) Synthetic-2D-FC

Figure 9.1. K-means Clustering Synthetic Dataset: Run Time (secs)



(a) ISIC-PCA10

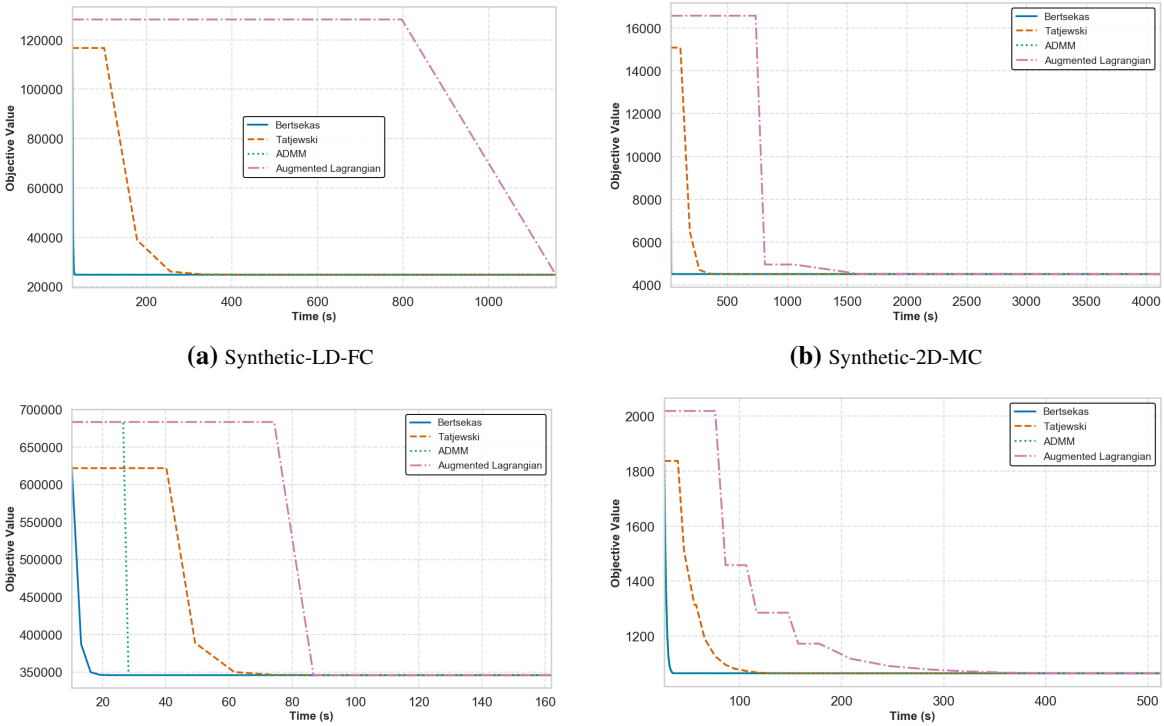
(b) ISIC-PCA2



(c) MedQA-PCA20

(d) MedQA-PCA2

Figure 9.2. K-means Clustering Real World Dataset: Run Time (secs)



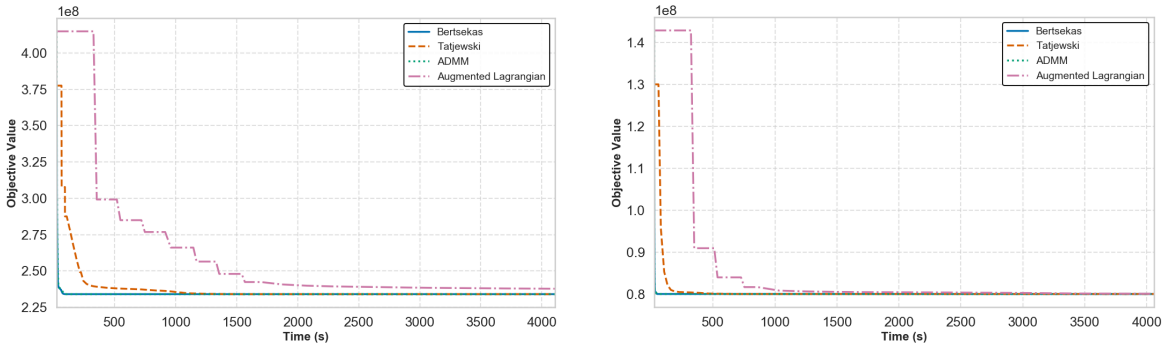
(a) Synthetic-LD-FC

(b) Synthetic-2D-MC

(c) Synthetic-HD-MC

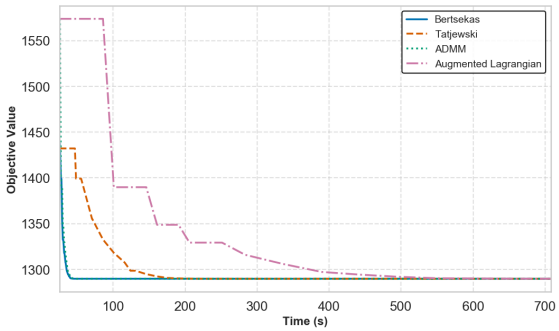
(d) Synthetic-2D-FC

Figure 9.3. K-means Clustering Synthetic Dataset: Objective Values

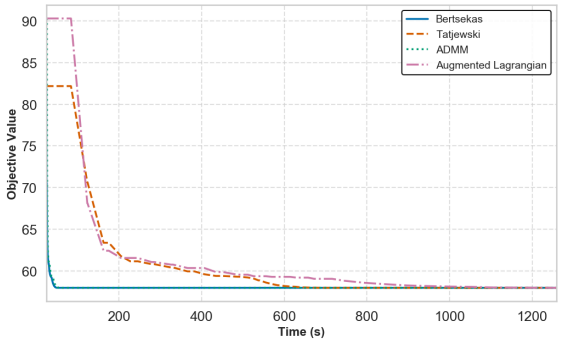


(a) ISIC-PCA10

(b) ISIC-PCA2

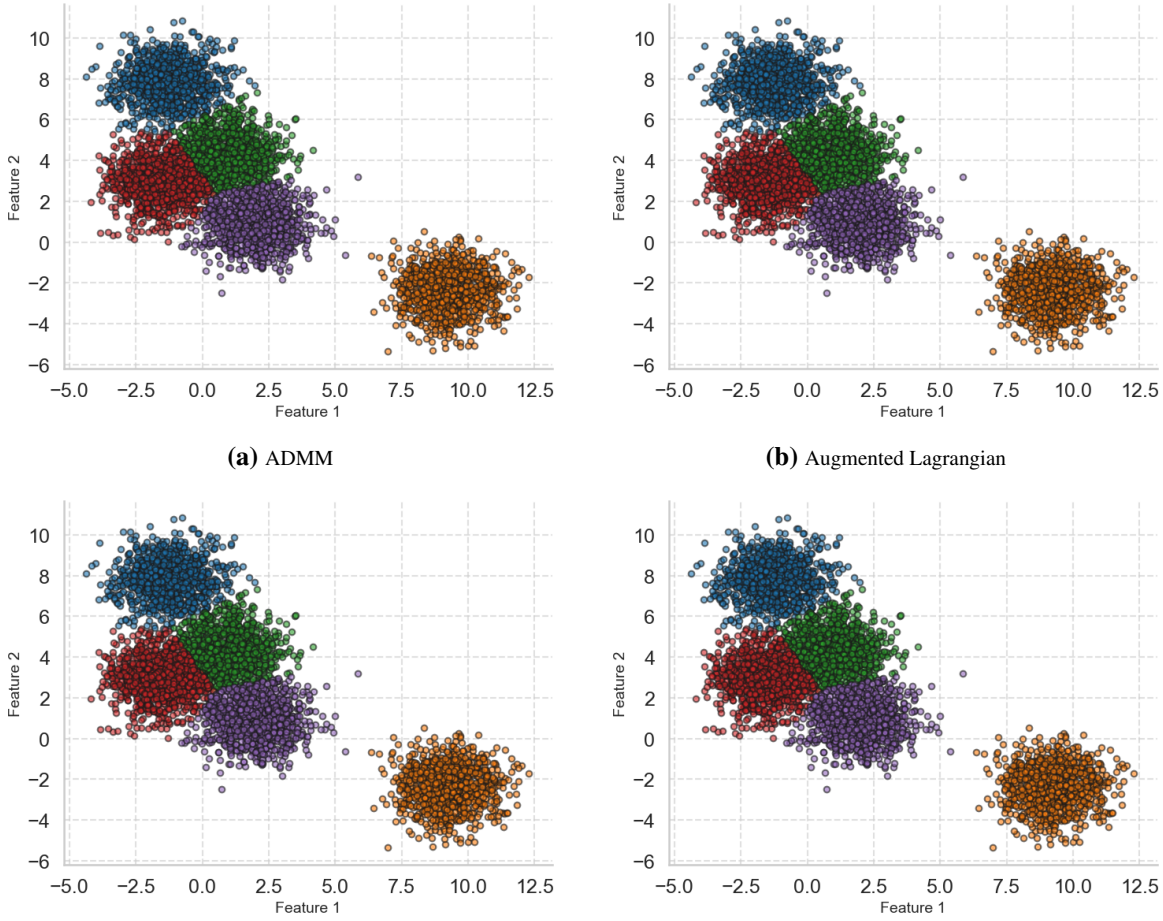


(c) MedQA-PCA20



(d) MedQA-PCA2

Figure 9.4. K-means Clustering Real World Dataset: Run Time (secs)



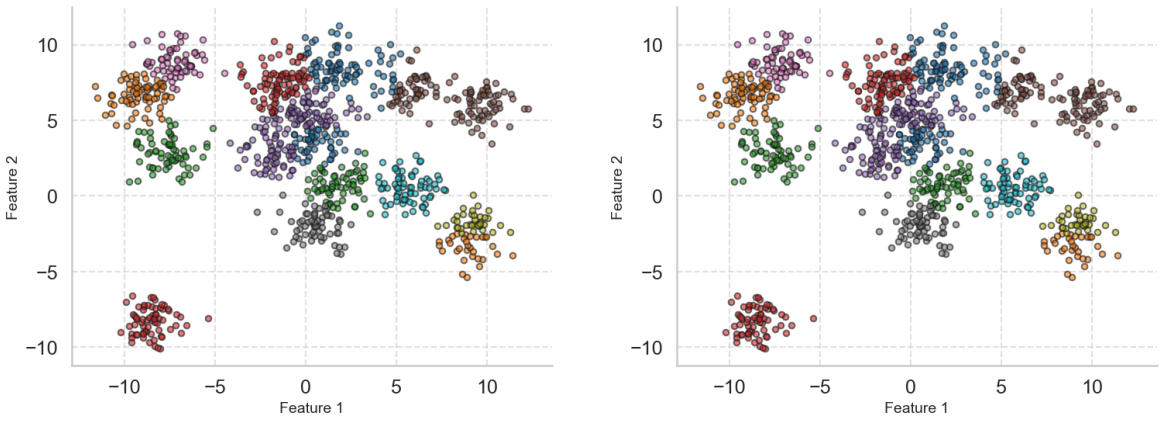
(a) ADMM

(b) Augmented Lagrangian

(c) Bertsekas

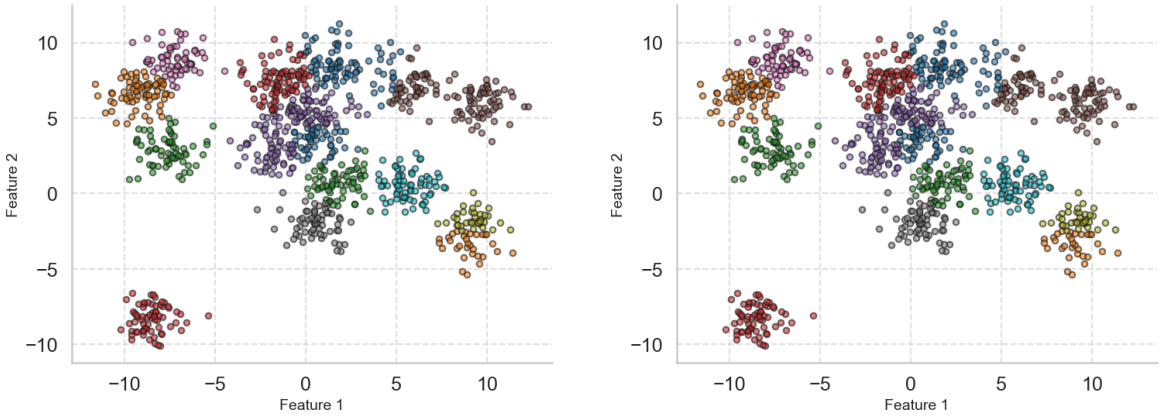
(d) Tatjewski

Figure 9.5. Clusters Synthetic Dataset: Synthetic-LD-FC



(a) ADMM

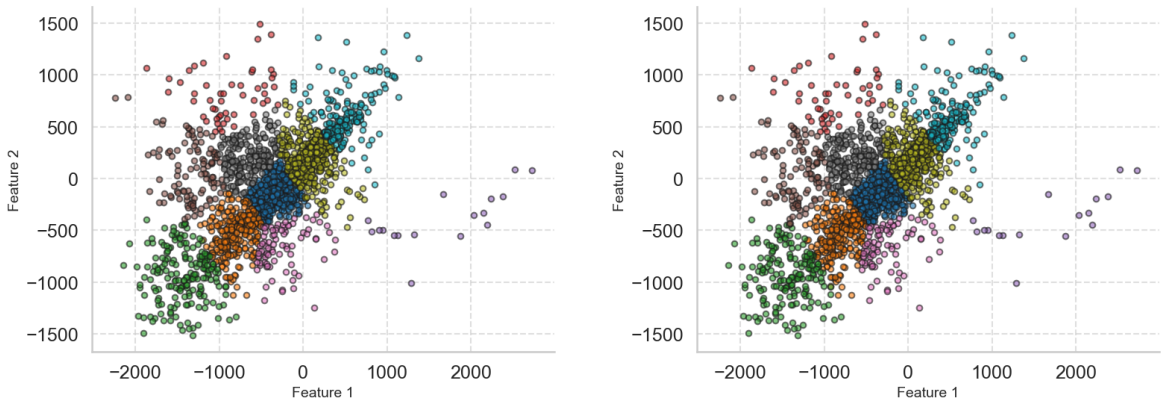
(b) Augmented Lagrangian



(c) Bertsekas

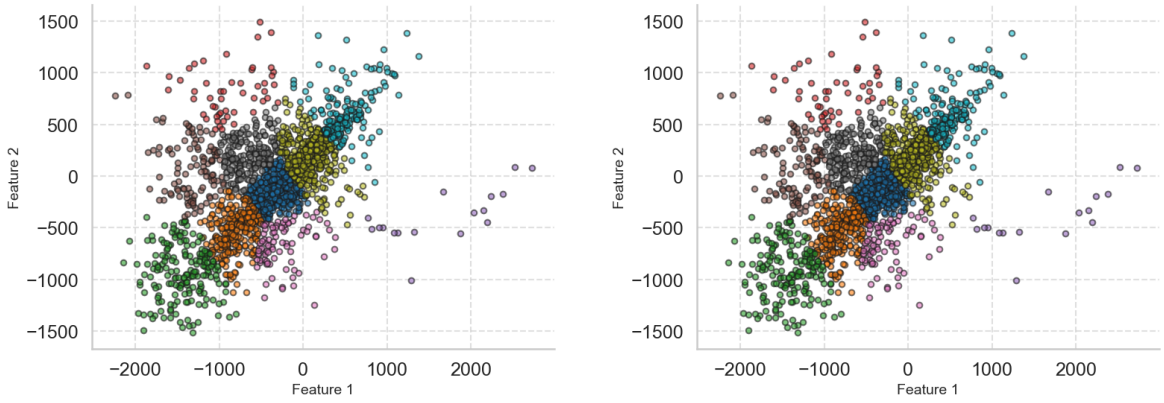
(d) Tatjewski

Figure 9.6. Clusters Synthetic Dataset: Synthetic-2D-MC



(a) ADMM

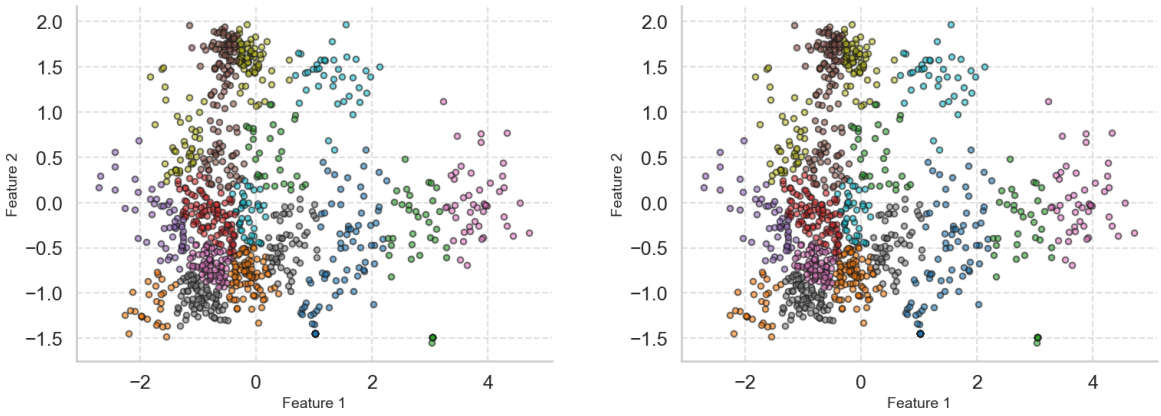
(b) Augmented Lagrangian



(c) Bertsekas

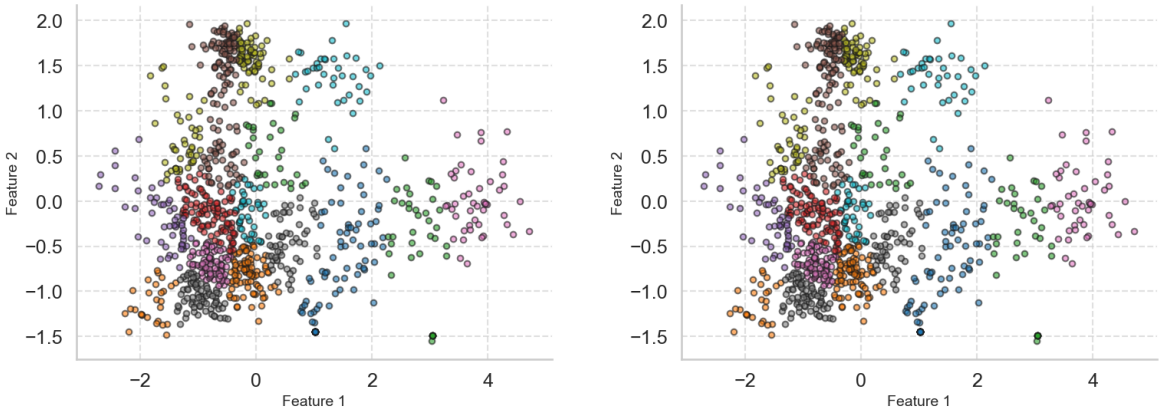
(d) Tatjewski

Figure 9.7. Clusters Real World Dataset: ISIC-PCA2



(a) ADMM

(b) Augmented Lagrangian



(c) Bertsekas

(d) Tatjewski

Figure 9.8. Clusters Real World Dataset: MedQA-PCA2

10. Synchronous and Asynchronous Solutions of K-Means Clustering Using Bertsekas Decomposition Method

Building on the formulation and algorithmic foundations introduced in Section 4, this section presents a practical application of Bertsekas Augmented Lagrangian method in solving the K-Means clustering problem in both synchronous and asynchronous settings. While the conceptual formulation of the K-Means objective and constraints has already been detailed, here the focus is on the application of decomposition techniques that allow for distributed and scalable optimization.

In particular, the investigation focuses on how the synchronous and asynchronous variants of the Bertsekas method affect clustering performance, computational time, and parallel efficiency. The synchronous approach requires global coordination at each iteration, potentially causing delays in large-scale systems. In contrast, the asynchronous method allows updates to proceed independently, based on partially updated information, making it more suitable for parallel and distributed environments.

The following subsections provide the formulation of Bertsekas Augmented Lagrangian for K-Means, along with detailed iterative update rules for both approaches. Subsequent experiments evaluate their effectiveness across real-world datasets of varying size and complexity.

10.1. Synchronous Iterative Updates

The optimization problem (280)–(283) is solved using the Bertsekas Augmented Lagrangian method discussed in Section 4. The augmented Lagrangian for this problem is:

$$\begin{aligned} L_\rho(x,\phi,x^s,\phi^s,\mu) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left\{\sum_{q=1}^{Q}\left[x_{ij}\left(A_{iq}-\phi_{jq}\right)^2+\frac{\rho}{m}\left(\phi_{jq}-\phi_{jq}^s\right)^2\right]\right. \\ &\quad \left.+2\mu_j\left(\frac{\tau_j}{m}-x_{ij}\right)+\rho\left(x_{ij}-x_{ij}^s\right)^2\right\} \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} L_{\rho_{ij}}(x_{ij},\phi_j,x_{ij}^s,\phi_j^s,\mu_j) \end{aligned} \tag{314}$$

where,

$$L_{\rho_{ij}}(x_{ij},\phi_j,x^s_{ij},\phi^s_j,\mu_j) = \frac{1}{2}\Bigg[x_{ij}\sum_{q=1}^{Q}\left(A_{iq}-\phi_{jq}\right)^2 + 2\mu_j\left(\frac{\tau_j}{m}-x_{ij}\right) + \rho\left(x_{ij}-x^s_{ij}\right)^2 + \frac{\rho}{m}\left(\phi_{jq}-\phi^s_{jq}\right)^2\Bigg] \tag{315}$$

For each iteration $k+1$, the updates proceed as follows:

- **Cluster Assignment**: For each chunk $i = 1,\ldots,M$, the new assignment x_i^{k+1} is computed by minimizing the Lagrangian over the admissible set X_i:

$$x_i^{k+1} = \min_{x_i \in X_i} \sum_{i=1}^{m}\sum_{j=1}^{n} L_{\rho_{ij}}(x_{ij},\phi_j,x^s_{ij},\phi^s_j,\mu_j) \tag{316}$$

- **Auxiliary variable Update**: The auxiliary variable $x_{ij}^{s^{k+1}}$ is updated as:

$$x_{ij}^{s^{k+1}} = \xi_k x_{ij}^{s^k} + (1-\xi_k)x_{ij}^{k+1}, \quad i = 1,\ldots,m, \quad j = 1,\ldots,n \tag{317}$$

- **Cluster Center Update**: The cluster centers are updated as:

$$\begin{aligned}\phi_{jq}^{k+1} &= \begin{cases}\frac{\rho\phi_{jq}^k + \sum_{i=1}^m x_{ij}^{k+1}A_{iq}}{\sum_{i=1}^m x_{ij}^{k+1}+\rho}, & \text{if } \sum_{i=1}^m x_{ij}^{k+1} > 0 \\ \phi_{jq}^k, & \text{otherwise}\end{cases}, \quad \forall j = 1,\ldots,n; q = 1,\ldots,Q \\ \phi_{jq}^{s^{k+1}} &= \xi_k\phi_{jq}^{s^k} + (1-\xi_k)\phi_{jq}^{k+1}\end{aligned} \tag{318}$$

- **Dual Variable Update**: Finally, the dual variable μ_j is updated as:

$$\mu_j^{k+1} = \max\left\{0, \mu_j^k + \rho\left(\tau_j - \sum_{i=1}^{m} x_{ij}^{k+1}\right)\right\}, \quad j = 1,\ldots,n \tag{319}$$

Stop when $\phi_{jq}^{k+1} = \phi_{jq}^k \;\forall\; j,q$, else increment k by 1 and go to step 1.

Another challenge arises from the requirement to have all x_i^{k+1} values available before computing ϕ^{k+1} and μ^{k+1}. This dependency introduces delays and limits the benefits of full decomposition. To address this issue, the asynchronous approach discussed in Section 4 is employed.

10.2. Asynchronous Iterative Updates

Optimization problem (280)–(283) is solved using the asynchronous Bertsekas Augmented Lagrangian method formulated in Section 4. The augmented Lagrangian used is defined in (314).

For each iteration $k+1$, the asynchronous updates proceed as follows:

i's Algorithm: At iterations, $k = 1, 2, ...,$ *and* i:

1: From iteration to iteration, i receives updates μ_j^k and ϕ_j^k from all $j = 1, \ldots, n$.

2: At each update iteration $k' \in S_i$, i chooses the next x_i^{k+1} and $x_i^{s^{k+1}}$

$x_i^{k+1} = \arg\min_{x_i \in X_i} \sum_{j=1}^{n} L_{\rho_{ij}}(x_{ij}, \hat{\phi}_j^k, x_{ij}^{s^k}, \hat{\phi}_j^{s^k}, \hat{\mu}_j^k)$

$x_i^{s^{k+1}} = \xi_k x_i^{s^k} + (1-\xi_k) x_i^{k+1}$

Transmission is done at this rate until the next update

3: communicates x_i^{k+1} to all constraints controllers.

j's Algorithm: At iterations, $k = 1, 2, ...,$ *and* j:

1: From iteration to iteration, j receives updates x_i^k from all $i = 1, \ldots, N$.

2: At each update iteration $k' \in T_j$, j updates μ_j^{k+1} and ϕ_j^{k+1} such that

$$\phi_{jq}^{k+1} = \begin{cases} \frac{\rho \phi_{jq}^k + \sum_{i=1}^m \hat{x}_{ij}^{k+1} A_{iq}}{\rho + \sum_{i=1}^m \hat{x}_{ij}^{k+1}}, & \text{if } \sum_{i=1}^m \hat{x}_{ij}^{k+1} > 0 \\ \phi_{jq}^k, & \text{otherwise} \end{cases}, \quad \forall q = 1, \ldots, Q \tag{320}$$

$$\phi_{jq}^{s^{k+1}} = \xi_k \phi_{jq}^{s^k} + (1-\xi_k) \phi_{jq}^{k+1} \tag{321}$$

$$\mu_j^{k+1} = \max\left\{0, \mu_j^k + \rho\left(\tau_j - \sum_{i=1}^m \hat{x}_{ij}^{k+1}\right)\right\} \tag{322}$$

Transmission is done at this rate until the next update

3: communicates μ_j^{k+1} and ϕ_{jq}^{k+1} to all local optimizers.

Stop when $\phi_{jq}^{k+1} = \phi_{jq}^k \; \forall \; j, q$, otherwise increase k by 1 and go to step 1.

where

$$\hat{x}_{ij}^{k} = \sum_{k'=k-k^0}^{k} a_{ij}(k',k) x_{ij}^{k'}, \quad i = 1,\ldots,m; \quad j = 1,\ldots,n \tag{323}$$

$$\hat{\mu}_{j}^{k} = \sum_{k'=k-k^0}^{k} b_{j}(k',k) \mu_{j}^{k'}, \quad j = 1,\ldots,n \tag{324}$$

$$\hat{\phi}_{jq}^{k} = \sum_{k'=k-k^0}^{k} c_{jq}(k',k) \phi_{jq}^{k'}, \quad j = 1,\ldots,n; \quad q = 1,\ldots,Q \tag{325}$$

and

$$\sum_{k'=k-k^0}^{k} a_{ij}(k',k) = 1, \quad \forall k, \quad i = 1,\ldots,m; \quad j = 1,\ldots,n \tag{326}$$

$$\sum_{k'=k-k^0}^{k} b_{j}(k',k) = 1, \quad \forall k, \quad j = 1,\ldots,n \tag{327}$$

$$\sum_{k'=k-k^0}^{k} c_{jq}(k',k) = 1, \quad \forall k, \quad j = 1,\ldots,n; \quad q = 1,\ldots,Q \tag{328}$$

and

$$a_{ij}(k',k) \geq 0, \quad \forall k, \quad i = 1,\ldots,m; \quad j = 1,\ldots,n \tag{329}$$

$$b_{j}(k',k) \geq 0, \quad \forall k, \quad j = 1,\ldots,n \tag{330}$$

$$c_{jq}(k',k) \geq 0, \quad \forall k, \quad j = 1,\ldots,n; \quad q = 1,\ldots,Q \tag{331}$$

10.3. Numerical Tests

All implementation details and datasets used are as described in Section 8.3.1 and 9.2.1 respectively.

10.4. Results and Discussion

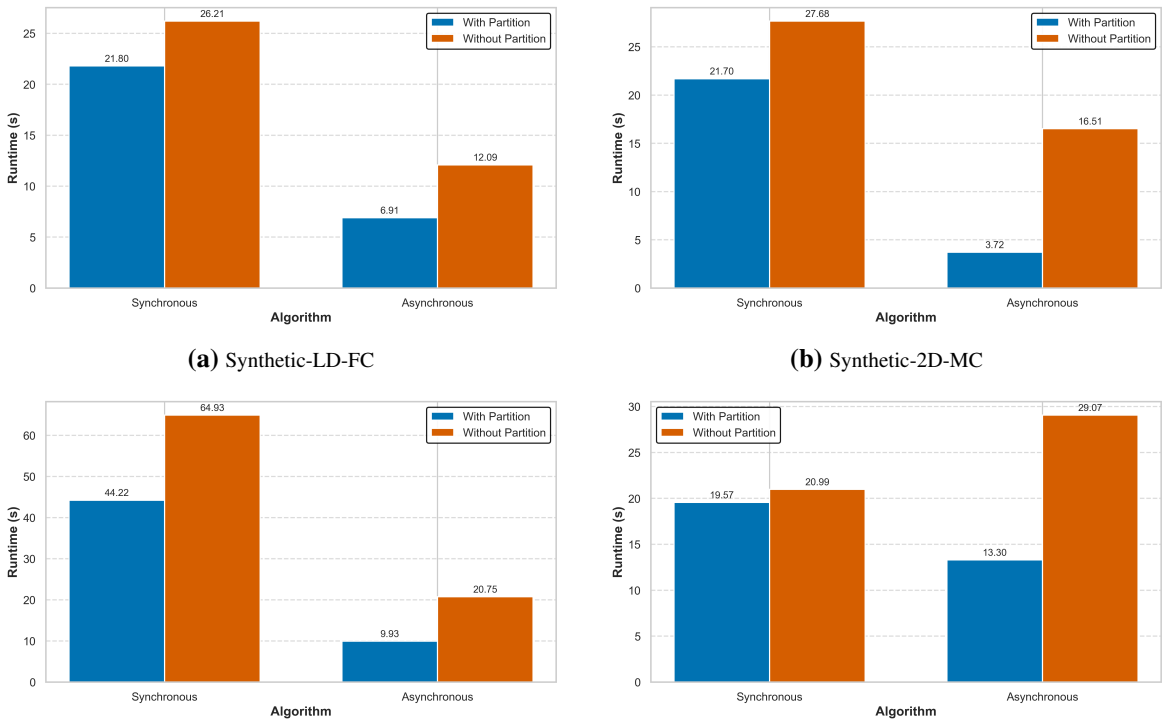
Tables 10.1 and 10.2 show the results of the synchronous and asynchronous variants of the Bertsekas algorithm for K-means clustering.

Table 10.1. Performance Comparison of Synchronous and Asynchronous Bertsekas Algorithms for K-Means Clustering on Synthetic Datasets

		No Partition				8 Partitions			
Dataset	**Algorithm**	**Objective**	Score	**Time**(s)	Status	**Objective**	Score	**Time**(s)	Status
Synthetic-LD-FC	Synchronous	24807.09	0.74	26	Converged	24807.09	0.74	21	Converged
	Asynchronous	24807.09	0.74	12	Converged	24807.09	0.74	6	Converged
Synthetic-HD-MC	Synchronous	346173.75	0.55	64	Converged	346173.75	0.55	44	Converged
	Asynchronous	346173.75	0.55	20	Converged	346170.58	0.55	9	Converged
Synthetic-2D-MC	Synchronous	1064.55	0.45	20	Converged	1064.55	0.45	19	Converged
	Asynchronous	1064.55	0.45	29	Converged	1064.55	0.45	13	Converged
Synthetic-2D-FC	Synchronous	4505.94	0.55	27	Converged	4505.94	0.55	21	Converged
	Asynchronous	4505.94	0.55	16	Converged	4505.94	0.55	3	Converged

Table 10.2. Performance Comparison of Synchronous and Asynchronous Bertsekas Algorithms for K-Means Clustering on Real World Datasets

		No Partition				8 Partitions			
Dataset	**Algorithm**	**Objective**	Score	**Time**(s)	Status	**Objective**	Score	**Time**(s)	Status
ISIC-PCA10	Synchronous	234036414.61	0.17	88	Converged	234036414.61	0.17	61	Converged
	Asynchronous	234036414.61	0.17	123	Converged	234036414.61	0.17	52	Converged
ISIC-PCA2	Synchronous	80010359.93	0.37	42	Converged	80010359.93	0.37	25	Converged
	Asynchronous	80010359.93	0.37	31	Converged	80010359.93	0.37	9	Converged
MedQA-PCA20	Synchronous	1289.70	0.14	35	Converged	1289.70	0.14	24	Converged
	Asynchronous	1289.70	0.14	29	Converged	1289.70	0.14	13	Converged
MedQA-PCA2	Synchronous	57.95	0.36	36	Converged	57.95	0.36	23	Converged
	Asynchronous	57.47	0.36	22	Converged	57.12	0.36	8	Converged



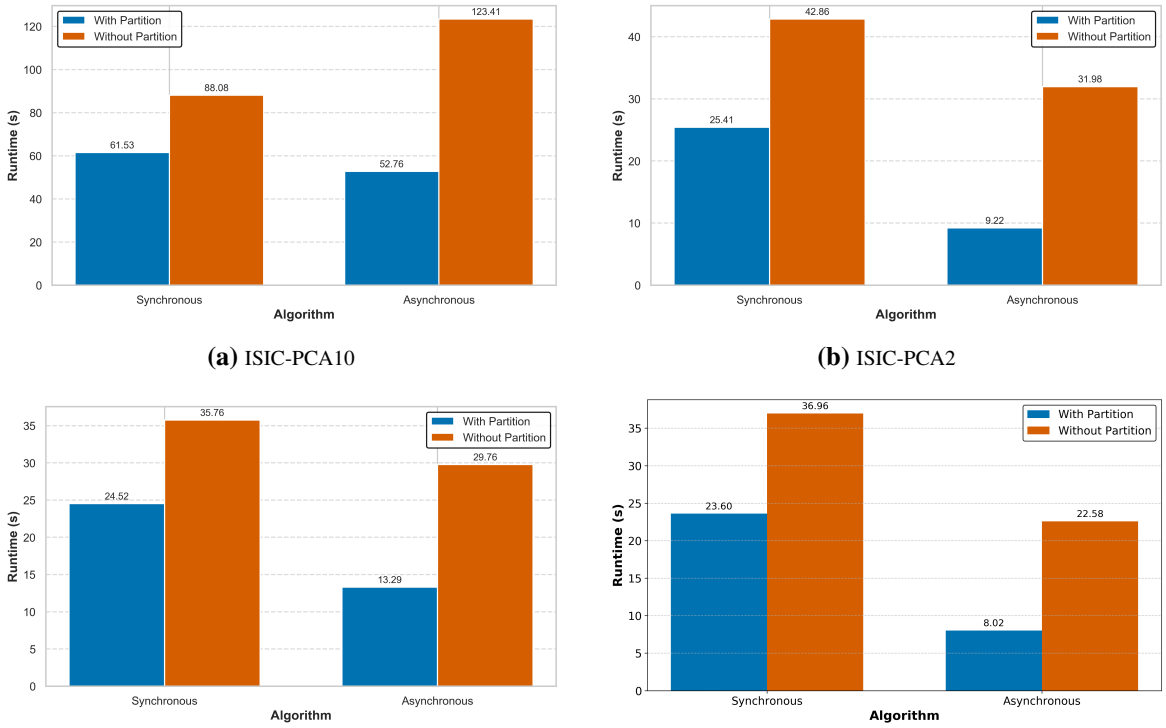
(a) Synthetic-LD-FC

(b) Synthetic-2D-MC

(c) Synthetic-HD-MC

(d) Synthetic-2D-FC

Figure 10.1. Sync-Async Algorithms (Synthetic Datasets): Run Time (secs)



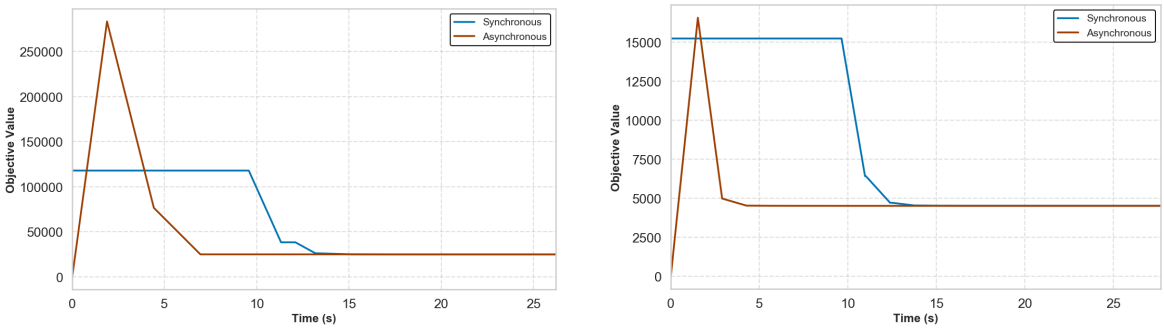
(a) ISIC-PCA10

(b) ISIC-PCA2

(c) MedQA-PCA20

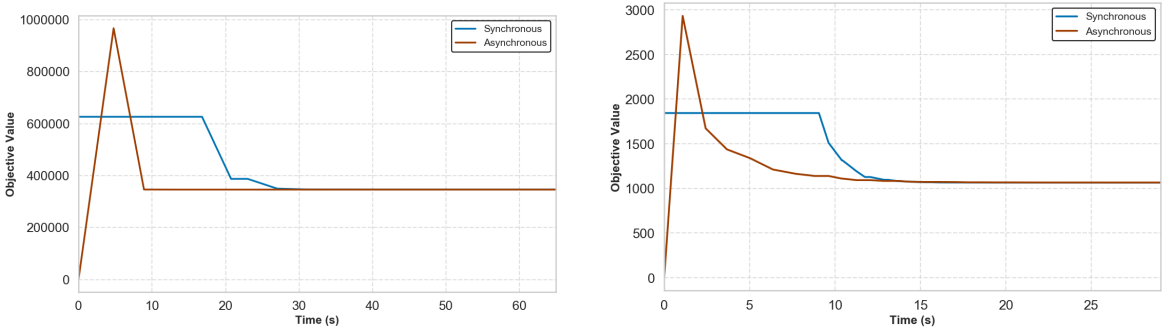
(d) MedQA-PCA2

Figure 10.2. Sync-Async Algorithms (Real World Datasets): Run Time (secs)



(a) Synthetic-LD-FC

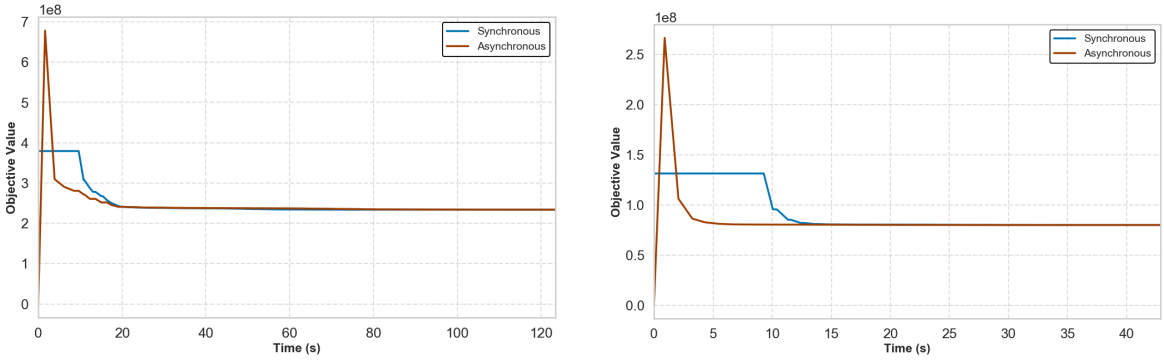
(b) Synthetic-2D-MC



(c) Synthetic-HD-MC

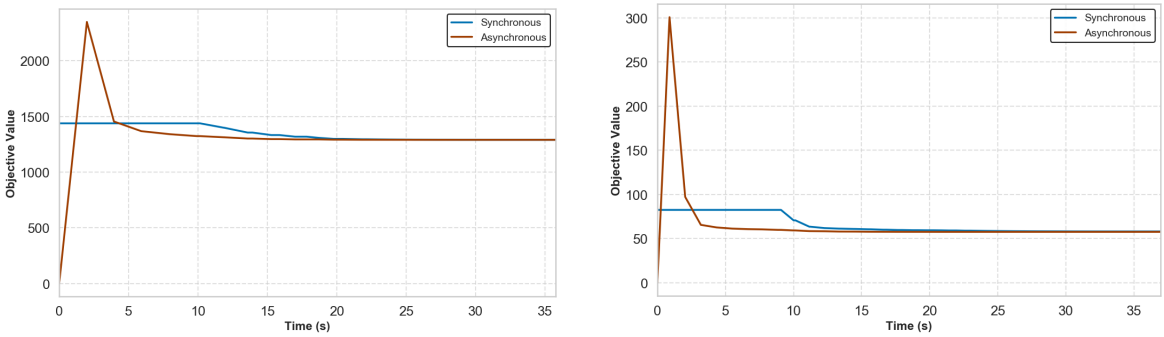
(d) Synthetic-2D-FC

Figure 10.3. Sync-Async Algorithms (Synthetic Datasets): Objective Values



(a) ISIC-PCA10

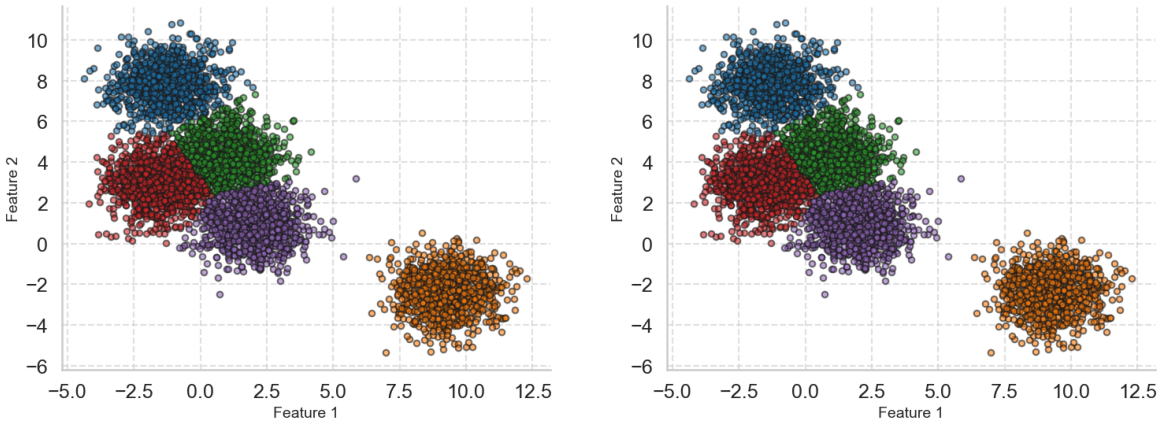
(b) ISIC-PCA2



(c) MedQA-PCA20

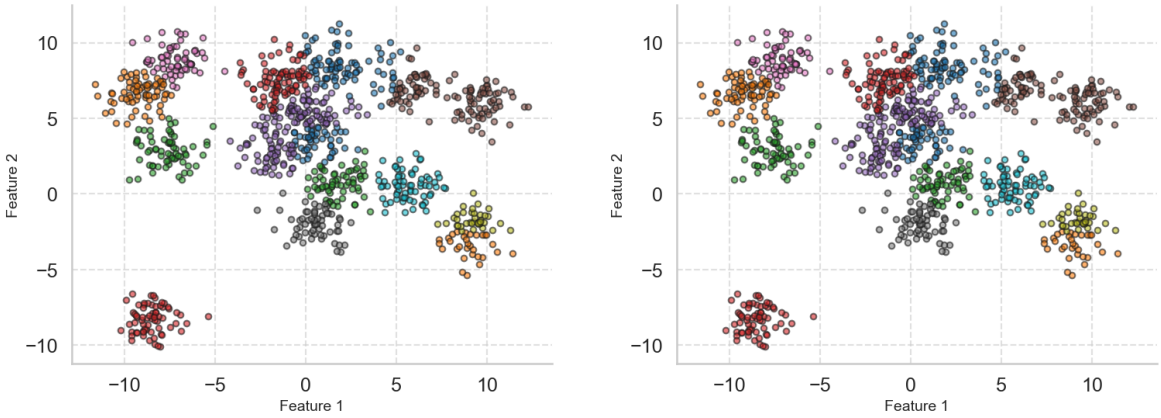
(d) MedQA-PCA2

Figure 10.4. Sync-Async Algorithms (Real World Datasets): Objective Values



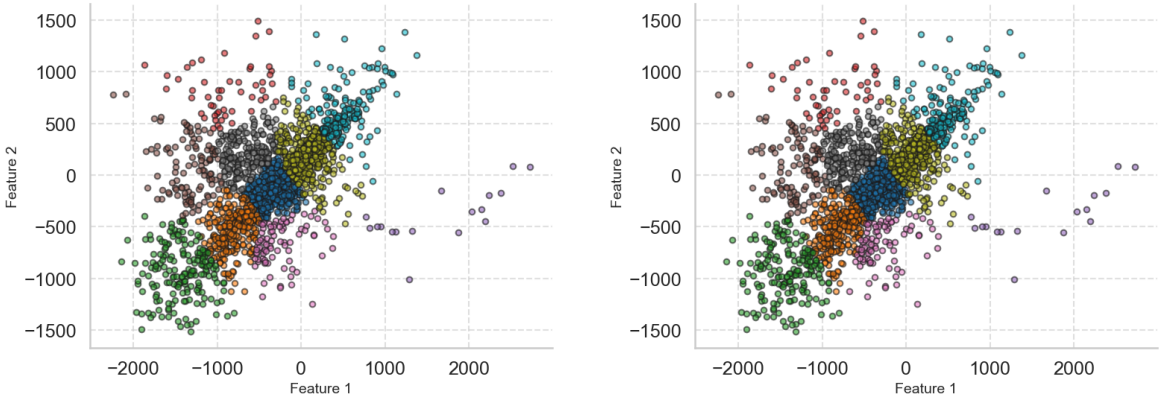
(a) Synchronous **(b)** Asynchronous

Figure 10.5. Sync-Async Algorithms: Synthetic-LD-FC



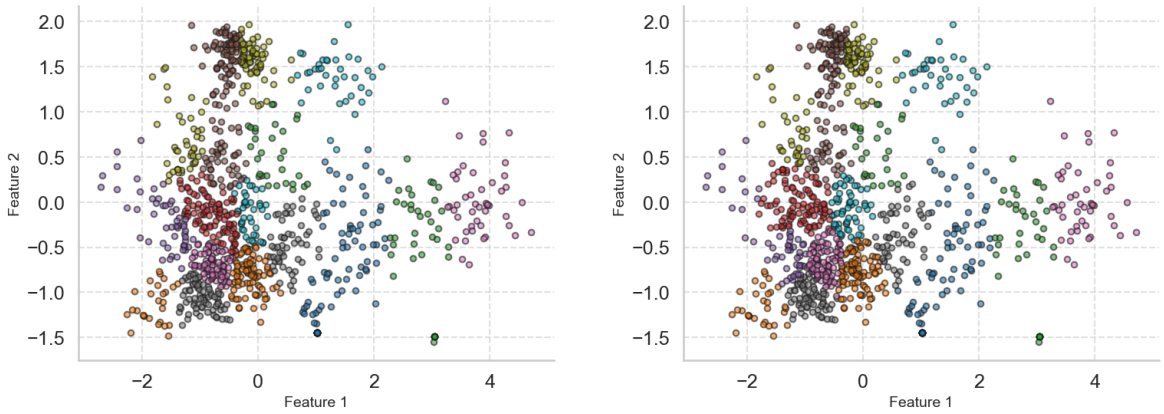
(a) Synchronous **(b)** Asynchronous

Figure 10.6. Sync-Async Algorithms: Synthetic-2D-MC



(a) Synchronous **(b)** Asynchronous

Figure 10.7. Sync-Async Algorithms: ISIC-PCA2



(a) Synchronous **(b)** Asynchronous

Figure 10.8. Sync-Async Algorithms: MedQA-PCA2

Synthetic Datasets

Across all synthetic datasets, asynchronous execution consistently reduced runtime without sacrificing solution quality. The objective values and clustering scores remained unchanged, demonstrating that asynchronous updates preserve convergence properties.

- In **low-dimensional datasets** (e.g., Synthetic-2D-FC), asynchronous execution nearly **halved the runtime** (e.g., 27s $\rightarrow$ 16s without partitioning, and 21s $\rightarrow$ 3s with partitioning).
- In **high-dimensional datasets** like Synthetic-HD-MC, the gains were even more significant under partitioning (44s $\rightarrow$ 9s), showcasing the scalability of the asynchronous variant.
- A notable exception occurred in **Synthetic-2D-MC (non-partitioned)**, where asynchronous execution took slightly longer (29s vs 20s), likely due to minor synchronization delays or load imbalance. However, with partitioning, performance improved substantially (13s vs 19s).

Overall, asynchronous execution **improved runtime in most cases**, particularly when partitioning was applied. The improvements were more pronounced as the data dimensionality or complexity increased.

Real-World Datasets

Asynchronous Bertsekas also performed favorably on real-world datasets, with **consistent convergence** and clustering quality across both variants.

- In most cases, **asynchronous execution was faster**—especially in the **partitioned**

setting, where it nearly halved the runtime on datasets like ISIC-PCA2 (25s $\rightarrow$ 9s) and MedQA-PCA20 (24s $\rightarrow$ 13s).

- **Notably**, asynchronous execution slightly underperformed in runtime on **ISIC-PCA10 without partitioning** (123s vs 88s), suggesting that for larger, unpartitioned datasets, synchronization overhead may be outweighed by update consistency in the synchronous version.
- On **MedQA-PCA2**, asynchronous execution achieved a slightly better objective value (57.12 vs 57.95) while maintaining identical clustering scores, suggesting a possible advantage in exploring descent directions more freely.

In both synthetic and real-world scenarios, the **asynchronous variant of Bertsekas' method demonstrated superior runtime performance**, particularly when the algorithm was partitioned and parallelized across multiple processing units. The ability to perform updates without global synchronization enables faster iteration and better utilization of parallel resources. These results validate the asynchronous framework as a scalable and efficient alternative for distributed K-means optimization, with negligible trade-offs in convergence quality.

11. Conclusion and Future Work

11.1. Summary of Contributions

This dissertation investigated distributed augmented Lagrangian methods for large-scale non-convex problems, focusing on identifying the most effective approach and extending it to asynchronous computation. Through extensive experiments on network flow, regularized systems of linear equations, and clustering tasks, the classic Bertsekas decomposition method consistently outperformed other candidates (e.g. ADMM, Tatjewski, SALA) in terms of solution quality, scalability, and convergence speed. This empirical advantage guided the methodological choice: the Bertsekas augmented Lagrangian method was adopted as the basis for further development. Building on this foundation, a novel *asynchronous Bertsekas method* was formulated, in which computational nodes update local variables without global synchronization. The theoretical model of this algorithm was then derived, and its convergence was proved: under mild regularity conditions and bounded communication delays, the asynchronous iterates converge to a KKT point of the original non-convex problem. These proofs align with classical asynchronous convergence theory and recent results for related algorithms [55]. In parallel, the new algorithm was implemented and validated on both synthetic benchmarks and real-world datasets. The experiments confirmed that the asynchronous variant retains solution quality while significantly reducing wall-clock time when deployed on multiple processors. In fact, consistent with prior studies, it was observed that allowing workers to proceed without waiting often yields faster convergence in practice, especially as the number of partitions grows.

The key contributions of this work can be summarized as follows:

- **Comprehensive comparative study:** Several decomposition-based Lagrangian algorithms were analyzed on non-convex separable problems (network flow, regularized systems of linear equations, clustering). The Bertsekas method emerged as superior in practice with respect to convergence rate and solution accuracy.
- **Development of an asynchronous variant:** A novel asynchronous algorithm based on the Bertsekas decomposition method was designed. This included the formulation of update rules, a communication model accommodating delays and partial information, and an implementation framework suited for distributed environments.
- **Rigorous convergence proof:** A formal convergence proof was established, showing that the asynchronous Bertsekas method converges to a KKT point under general non-convex assumptions. This addresses a gap in the literature, where convergence in asynchronous settings for this method was previously unproven.
- **Algorithmic extensions:** The method was extended to incorporate inequality con-

straints and mixed-integer variables by adapting the augmented Lagrangian formulation, thus expanding its applicability to a wider range of practical optimization problems.

- **Extensive numerical validation:** Numerical experiments on both synthetic and real datasets (including K-Means clustering, Regularized Systems of Linear Equations, and network optimization benchmarks) demonstrated that the asynchronous method achieves objective values comparable to the synchronous version, while delivering substantial reductions in runtime.
- **Insights on synchronous vs. asynchronous performance:** Comparative evaluation of synchronous and asynchronous implementations revealed trade-offs in coordination overhead and computational throughput. The asynchronous variant showed better scalability by reducing idle time and more efficiently leveraging parallel resources.

In summary, this work systematically addresses the research questions posed in Chapter 1. By first identifying the Bertsekas method as the most effective Lagrangian approach for the target application scenarios, and subsequently extending it into a provably convergent asynchronous algorithm, both theoretical and practical advances are achieved.

11.2. Limitations

Despite the positive outcomes, several limitations of the current work must be acknowledged. First, the convergence analysis assumes certain technical conditions: in particular, it requires that communication delays remain bounded and that updates occur often enough. In practice, this means that extreme network latencies or system failures could violate the assumptions of our theory. Second, the theoretical results are asymptotic and do not specify convergence rates. While our experiments suggest fast convergence in practice, a precise characterization of the rate is not provided here. Third, like most Lagrangian decomposition schemes, our method requires tuning of hyperparameters to achieve optimal performance. Fourth, the algorithm has been validated on a representative set of problems, but its behavior on other types of non-convex problems remains to be explored. Finally, the asynchronous implementation incurs some overhead in bookkeeping and communication to handle delayed updates; in small-scale or low-delay settings, the benefits may be less pronounced.

11.3. Future Directions

This research opens several avenues for further work. Key future directions include:

- **Generalizing the asynchronous model:** Extend the convergence analysis to more relaxed asynchronous settings, such as unbounded but decaying delays, stochastic update patterns, or partially unreliable communication.

- **Convergence rates and complexity:** Derive theoretical rates of convergence (e.g. sublinear or linear bounds) for the asynchronous Bertsekas method, including iteration or time complexity results.
- **Algorithmic acceleration:** Explore techniques to accelerate convergence, such as Nesterov-type momentum, adaptive penalty update rules, or quasi-Newton approximations.
- **Broader application domains:** Apply the asynchronous Bertsekas algorithm to additional problem classes, including real-time resource allocation, distributed learning, or large-scale inference.
- **Robustness and fault tolerance:** Investigate enhancements for robustness, such as redundancy, error-correction, or adaptive compensation for stale or missing updates.
- **Hybrid synchronization strategies:** Study hybrid schemes that interpolate between fully synchronous and fully asynchronous updates, such as grouped worker synchronization.

In conclusion, this dissertation lays a foundation for asynchronous augmented-Lagrangian optimization in distributed, non-convex settings. The contributions made here, methodological, theoretical, and empirical, advance the state of the art and provide a basis for future developments.

References

[1] R. Zhang and J. Kwok, "Asynchronous distributed admm for consensus optimization", in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, Bejing, China: PMLR, Jun. 2014, pp. 1701–1709. [Online]. Available: https://proceedings.mlr.press/v32/zhange14.html.

[2] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers", *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010. DOI: 10.1561/2200000016. [Online]. Available: https://doi.org/10.1561/2200000016.

[3] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms", in *1984 American Control Conference*, IEEE, Jul. 1984, pp. 484–489. DOI: 10.23919/acc.1984.4788427. [Online]. Available: http://dx.doi.org/10.23919/acc.1984.4788427.

[4] J. A. Fessler, *Duality in convex optimization*, https://web.eecs.umich.edu/~fessler/course/598/l/n-07-dual.pdf, Lecture notes, EECS 598, University of Michigan, 2007.

[5] E. K. Ryu and S. P. Boyd, "A primer on monotone operator methods", 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:17513145.

[6] H. W. Kuhn and A. W. Tucker, "Nonlinear programming", in *Second Berkeley Symposium on Mathematical Statistics and Probability*, J. Neyman, Ed., vol. 2, Berkeley, CA: University of California Press, 1951, pp. 481–492. [Online]. Available: https://projecteuclid.org/proceedings/berkeley-symposium-on-mathematical-statistics-and-probability/Proceedings-of-the-Second-Berkeley-Symposium-on-Mathematical-Statistics-and/Chapter/Nonlinear-Programming/bsmsp/1200500249.

[7] K. J. Arrow and L. Hurwicz, "Reduction of constrained maxima to saddlepoint problems", in *Third Berkeley Symposium on Mathematical Statistics and Probability*, J. Neyman, Ed., vol. 5, Berkeley, CA: University of California Press, 1956, pp. 1–20. [Online]. Available: https://projecteuclid.org/proceedings/berkeley-symposium-on-mathematical-statistics-and-probability/Proceedings-of-the-Third-Berkeley-Symposium-on-Mathematical-Statistics-and/Chapter/Reduction-of-Constrained-Maxima-to-Saddle-point-Problems/bsmsp/1200511853.

[8] K. Arrow, L. Hurwicz, and Uzawa, *Studies in Linear and Nonlinear Programming* (Stanford mathematical studies in the social sciences). Stanford University Press, 1958, ISBN: 9780804705622. [Online]. Available: https://books.google.pl/books?id=TkRlnQEACAAJ.

[9] M. J. D. Powell, "A method for nonlinear constraints in minimization problems", *Optimization*, pp. 283–298, 1969. [Online]. Available: https://ci.nii.ac.jp/naid/20000922074/en/.

[10] M. R. Hestenes, "Multiplier and gradient methods", *Journal of Optimization Theory and Applications*, vol. 4, no. 5, pp. 303–320, Nov. 1969. DOI: 10.1007/bf00927673. [Online]. Available: https://doi.org/10.1007/bf00927673.

[11] D. P. Bertsekas, "Convexification procedures and decomposition methods for nonconvex optimization problems", *Journal of Optimization Theory and Applications*, vol. 29, no. 2, pp. 169–197, Oct. 1979. DOI: 10.1007/bf00937167. [Online]. Available: https://doi.org/10.1007/bf00937167.

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

[13] R. Fletcher, "A class of methods for nonlinear programming with termination and convergence properties", in *Integer and Nonlinear programming*, J. Abadie, Ed., Amsterdam: North-Holland, 1970, pp. 157–173.

[14] H. Mukai and E. Polak, "A quadratically convergent primal-dual algorithm with global convergence properties for solving optimization problems with equality constraints", *Mathematical Programming*, vol. 9, no. 1, pp. 336–349, 1975.

[15] A. Tanikawa and H. Mukai, "A new technique for nonconvex primal-dual decomposition of a large-scale separable optimization problem", *IEEE Transactions on Automatic Control*, vol. 30, no. 2, pp. 133–143, 1985. DOI: 10.1109/TAC.1985.1103899.

[16] P. Tatjewski and B. Engelmann, "Two-level primal-dual decomposition technique for large-scale nonconvex optimization problems with constraints", *Journal of Optimization Theory and Applications*, vol. 64, no. 1, pp. 183–205, Jan. 1990. DOI: 10.1007/bf00940031. [Online]. Available: https://doi.org/10.1007/bf00940031.

[17] P. Tatjewski, "New dual-type decomposition algorithm for nonconvex separable optimization problems", *Automatica*, vol. 25, no. 2, pp. 233–242, Mar. 1989. DOI: 10.1016/0005-1098(89)90076-9. [Online]. Available: https://doi.org/10.1016/0005-1098(89)90076-9.

[18] R. Glowinski and A. Marroco, "On the approximation, by finite elements of order one, and the resolution, by penalization-duality of a class of non-dirichlet problems lin 'eaires", fr, *ESAIM: Mathematical Modeling and Numerical Analysis - Mathematical Modeling and Numerical Analysis*, vol. 9, no. R2, pp. 41–76, 1975. [Online]. Available: http://www.numdam.org/item/M2AN_1975__9_2_41_0/.

[19] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation", *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17–40, 1976. DOI: 10.1016/0898-1221(76)90003-1. [Online]. Available: https://doi.org/10.1016/0898-1221(76)90003-1.

[20] A. Hamdi, P. Mahey, and J. P. Dussault, "A new decomposition method in nonconvex programming via a separable augmented lagrangian", in *Recent Advances in Optimization*, P. Gritzmann, R. Horst, E. Sachs, and R. Tichatschke, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 90–104.

[21] A. Hamdi, "Two-level primal-dual proximal decomposition technique to solve large scale optimization problems", *Applied Mathematics and Computation*, vol. 160, no. 3, pp. 921–938, 2005. DOI: 10.1016/j.amc.2003.11.040.

[22] A. Hamdi and S. K. Mishra, "Decomposition methods based on augmented lagrangians: A survey", *Springer Optimization and Its Applications*, vol. 50, pp. 175–203, 2011. DOI: 10.1007/978-1-4419-9640-4_11.

[23] D. P. Bertsekas, *Lagrange Multiplier Methods in Constrained Optimization*. Academic Press, 1982.

[24] E. Minguzzi, "The Equality of Mixed Partial Derivatives", *Real Analysis Exchange*, vol. 40, no. 1, pp. 81–98, 2015.

[25] H. H. Bauschke and P. L. Combettes, "Fenchel–rockafellar duality", in *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Cham: Springer International Publishing, 2017, pp. 247–262, ISBN: 978-3-319-48311-5. DOI: 10.1007/978-3-319-48311-5_15. [Online]. Available: https://doi.org/10.1007/978-3-319-48311-5_15.

[26] D. Bertsekas, *Nonlinear programming*, en. Athena Scientific, Sep. 2016.

[27] A. Karbowski, "Comments on "optimization flow control .i. basic algorithm and convergence"", *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 338–339, Apr. 2003, ISSN: 1558-2566. DOI: 10.1109/tnet.2003.810318. [Online]. Available: http://dx.doi.org/10.1109/TNET.2003.810318.

[28] *Data centres and data transmission networks*, https://www.iea.org/reports/data-centres-and-data-transmission-networks, Accessed: 2024-02-12.

[29] M. Koot and F. Wijnhoven, "Usage impact on data center electricity needs: A system dynamic forecasting model", *Applied Energy*, vol. 291, 2021. DOI: 10.1016/j.apenergy.2021.116798.

[30] P. Jaskóła, P. Arabas, and A. Karbowski, "Simultaneous routing and flow rate optimization in energy-aware computer networks", *International Journal of Applied Mathematics and Computer Science*, vol. 26(1), pp. 231–243, 2016.

[31] J. Wang, L. Li, S. H. Low, and J. C. Doyle, "Cross-layer optimization in tcp/ip networks", *IEEE/ACM Transactions on Networking*, vol. 13(3), pp. 582–595, 2005.

[32] M. Jünger, T. Liebling, D. Naddef, *et al.*, *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*. Springer, 2010.

[33] D. Li and X. Sun, *Nonlinear Integer Programming*. Springer, 2006.

[34] I. Ruksha and A. Karbowski, "Decomposition methods for the network optimization problem of simultaneous routing and bandwidth allocation based on lagrangian relaxation", *Energies*, vol. 15, no. 20, p. 7634, Oct. 2022. DOI: 10.3390/en15207634. [Online]. Available: https://doi.org/10.3390/en15207634.

[35] R. Rockafellar, "Augmented lagrange multiplier functions and duality in nonconvex programming", *SIAM J Control*, vol. 12, no. 2, pp. 268–285, 1974. DOI: 10.1137/0312021.

[36] X. Huang and X. Yang, "Duality and exact penalization via a generalized augmented lagrangian function", in *Optimization and Control with Applications*, L. Qi, K. Teo, and X. Yang, Eds., Boston, MA: Springer US, 2005, pp. 101–114.

[37] X. Huang and X. Yang, "Further study on augmented lagrangian duality theory", *Journal of Global Optimization*, vol. 31, no. 2, pp. 193–210, 2005. DOI: 10.1007/s10898-004-5695-7.

[38] R. S. Burachik and A. Rubinov, "On the absence of duality gap for lagrange-type functions", *Journal of Industrial and Management Optimization*, vol. 1, no. 1, pp. 33–38, 2005. DOI: 10.3934/jimo.2005.1.33.

[39] A. Nedich and A. Ozdaglar, "A geometric framework for nonconvex optimization duality using augmented lagrangian functions", *Journal of Global Optimization*, vol. 40, no. 4, pp. 545–573, 2008. DOI: 10.1007/s10898-006-9122-0.

[40] N. Boland and A. Eberhard, "On the augmented lagrangian dual for integer programming", *Mathematical Programming*, vol. 150, no. 2, pp. 491–509, 2015. DOI: 10.1007/s10107-014-0763-3.

[41] X. Gu, S. Ahmed, and S. S. Dey, "Exact augmented lagrangian duality for mixed integer quadratic programming", *SIAM Journal on Optimization*, vol. 30, no. 1, pp. 781–797, 2020. DOI: 10.1137/19M1271695.

[42] D. P. Bertsekas, "Multiplier methods: A survey", *Automatica*, vol. 12, no. 2, pp. 133–145, 1976. DOI: 10.1016/0005-1098(76)90077-7.

[43] A. P. Wierzbicki, "A penalty function shifting method in constrained static optimization and its convergence properties", *Archiwum Automatyki i Telemechaniki*, vol. 16, pp. 395–416, 1971.

[44] O. Stein, J. Oldenburg, and W. Marquardt, "Continuous reformulations of discrete-continuous optimization problems", *Computers and Chemical Engineering*, vol. 28, no. 10, pp. 1951–1966, 2004. DOI: 10.1016/j.compchemeng.2004.03.011.

[45] M. A. Bragin, P. B. Luh, B. Yan, and X. Sun, "A scalable solution methodology for mixed-integer linear programming problems arising in automation", *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 531–541, 2019. DOI: 10.1109/TASE.2018.2835298.

[46] Y. Chen, Q. Guo, and H. Sun, "Decentralized unit commitment in integrated heat and electricity systems using sdm-gs-alm", *IEEE Transactions on Power Systems*, vol. 34, no. 3, pp. 2322–2333, 2019. DOI: 10.1109/TPWRS.2018.2885805.

[47] M. Cordova, W. d. Oliveira, and C. Sagastizábal, "Revisiting augmented lagrangian duals", *Mathematical Programming*, vol. 196, no. 1-2, pp. 235–277, 2022. DOI: 10.1007/s10107-021-01703-5.

[48] Z. Liu and O. Stursberg, "Distributed solution of mixed-integer programs by admm with closed duality gap", in *IEEE Conference on Decision and Control*, 2022, pp. 279–286. DOI: 10.1109/CDC51059.2022.9992566.

[49] M. Hong, "A distributed, asynchronous, and incremental algorithm for nonconvex optimization: An admm approach", *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 935–945, 2018. DOI: 10.1109/TCNS.2017.2657460.

[50] Z. Lin, H. Li, and C. Fang, *Alternating Direction Method of Multipliers for Machine Learning*. Springer, 2022.

[51] K. Sun and X. A. Sun, "A two-level distributed algorithm for nonconvex constrained optimization", *Computational Optimization and Applications*, vol. 84, no. 2, pp. 609–649, 2023. DOI: 10.1007/s10589-022-00433-4.

[52] B. Houska, J. Frasch, and M. Diehl, "An augmented lagrangian based algorithm for distributed NonConvex optimization", *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1101–1127, 2016. DOI: 10.1137/140975991. [Online]. Available: https://doi.org/10.1137/140975991.

[53] N. Boland, J. Christiansen, B. Dandurand, A. Eberhard, and F. Oliveira, "A parallelizable augmented lagrangian method applied to large-scale non-convex-constrained optimization problems", *Mathematical Programming*, vol. 175, no. 1-2, pp. 503–536, 2019. DOI: 10.1007/s10107-018-1253-9.

[54] *Networkx - network analysis in python*, https://networkx.org/, Accessed: 2024-02-12.

[55] S. Low and D. Lapsely, "Optimization flow control. i. basic algorithm and convergence", *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, 1999, ISSN: 1063-6692. DOI: 10.1109/90.811451. [Online]. Available: http://dx.doi.org/10.1109/90.811451.

[56] The Pandas Development Team, *pandas.merge_asof — pandas documentation*, Accessed: 2025-06-20, 2024. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.merge_asof.html.

[57] Z. Zhou, P. Mertikopoulos, N. Bambos, *et al.*, "Distributed asynchronous optimization with unbounded delays: How slow can you go?", in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Jul. 2018, pp. 5970–5979. [Online]. Available: https://proceedings.mlr.press/v80/zhou18b.html.

[58] W. H. Wolberg and O. L. Mangasarian, *Breast cancer wisconsin (diagnostic)*, 1993. DOI: https://doi.org/10.24432/C5DW2B.

[59] P. Bradley, K. Bennett, and A. Demiriz, "Constrained k-means clustering", Microsoft Research, Redmond, Tech. Rep. MSR-TR-2000-65, 2000, Microsoft Research Technical Report. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2000-65.pdf.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[61] P. Tschandl, C. Rosendahl, and H. Kittler, "The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions", *Scientific Data*, vol. 5, no. 1, Aug. 2018, ISSN: 2052-4463. DOI: 10.1038/sdata.2018.161. [Online]. Available: http://dx.doi.org/10.1038/sdata.2018.161.

[62] N. C. F. Codella, D. Gutman, M. E. Celebi, *et al.*, *Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic)*, 2017. DOI: 10.

48550/ARXIV.1710.05006. [Online]. Available: https://arxiv.org/abs/1710.05006.

[63] M. Combalia, N. C. F. Codella, V. Rotemberg, *et al.*, *Bcn20000: Dermoscopic lesions in the wild*, 2019. DOI: 10.48550/ARXIV.1908.02288. [Online]. Available: https://arxiv.org/abs/1908.02288.

[64] A. Ben Abacha and D. Demner-Fushman, "A question-entailment approach to question answering", *BMC Bioinform.*, vol. 20, no. 1, 511:1–511:23, 2019. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-3119-4.

[65] E. Alsentzer, J. R. Murphy, W. Boag, *et al.*, "Publicly available clinical BERT embeddings", *CoRR*, vol. abs/1904.03323, 2019. arXiv: 1904.03323. [Online]. Available: http://arxiv.org/abs/1904.03323.