INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2026, VOL. 72, NO. 1, PP. 1–10
Manuscript received September 5, 2025; revised February, 2026. doi: 10.24425/ijet.2026.157873

Synchronous and asynchronous Bertsekas methods for energy-aware networks and solution of regularized linear systems

Anthony Nwachukwu, and Andrzej Karbowski

***Abstract*—This paper develops and analyzes augmented Lagrangian-based methods for two classes of large-scale optimization problems relevant to modern computational systems: energy-aware network routing with bandwidth allocation and the solution of regularized linear systems. In the first problem, routing and bandwidth allocation are jointly optimized in communication networks including energy components, modeled as a mixed-integer nonlinear program. In the second, regularized linear systems are formulated to address ill-posed or ill-conditioned problems by introducing stabilization terms such as ℓ_2 regularization. For both problems, synchronous and asynchronous distributed optimization schemes are designed using decomposition techniques grounded in augmented Lagrangian theory. Extensive numerical experiments across diverse datasets, including network flow instances and benchmark regularized linear systems, demonstrate that the asynchronous variants retain comparable solution quality while significantly improving computational performance, particularly under delay and scalability conditions. These findings reinforce the practical value of asynchronous augmented Lagrangian methods for distributed, high-dimensional, and delay-sensitive optimization problems.**

***Keywords*—Augmented Lagrangian methods, Bertsekas decomposition, Asynchronous optimization, Synchronous optimization, Energy-aware routing, Bandwidth allocation, Linear Systems, Distributed algorithms, Mixed-integer nonlinear programming (MINLP), Dual decomposition, Regularized Linear Systems**

I. Introduction

With the surge in large-scale data generation and real-time communication needs, the demand for optimization frameworks that deliver efficiency, scalability, and sustainability has never been more pressing. Within contemporary telecommunication infrastructures, the simultaneous optimization of routing paths and bandwidth allocations plays a central role in balancing quality of service (QoS) with energy consumption, two often competing objectives. However, conventional approaches tend to decouple these components or rely on static heuristics, thereby limiting the achievable performance, particularly when energy usage is explicitly modeled as part of the system cost [1], [2]. A fully integrated formulation, while more reflective of real-world constraints, results in a mixed-integer nonlinear programming (MINLP) problem characterized by discrete variables and inherent non-convexities. Such formulations pose significant computational challenges and demand algorithmic strategies that can scale while maintaining solution quality [3], [4].

In parallel, the solution of regularized linear systems has become increasingly important across domains such as signal processing, numerical optimization, and machine learning. Such problems arise when seeking stable solutions to ill-posed or ill-conditioned systems, where regularization terms (e.g., ℓ_2 penalties) are introduced to ensure robustness and numerical stability [5], [6]. While these techniques enhance solution quality, they also introduce significant computational challenges, particularly in large-scale settings.

A unifying feature of these application domains is the reliance on distributed optimization methods, which decompose large-scale problems into subproblems solvable by multiple agents in parallel. Classical synchronous approaches, such as the Bertsekas Augmented Lagrangian (BAL) framework, enforce strict coordination among agents, ensuring orderly updates but often at the cost of idle waiting and vulnerability to communication delays. Asynchronous methods [7]–[12], by contrast, allow agents to proceed with updates based on local and potentially outdated information, reducing synchronization overheads and improving resilience in large and heterogeneous networks. This paradigm shift has been shown to accelerate convergence in practice while maintaining theoretical guarantees under mild assumptions.

Building upon this perspective, the present work introduces asynchronous augmented Lagrangian schemes for two structurally analogous problems: energy-aware routing with bandwidth allocation and the solution of regularized linear systems. For the network routing problem, we extend the prior synchronous formulation [13] by incorporating asynchronous update mechanisms into a model that couples binary routing decisions with continuous flow allocations. For regularized linear systems, we develop a tailored Bertsekas Augmented Lagrangian scheme under inequality constraints, considering both synchronized and asynchronized update schemes. Through comprehensive numerical experiments on synthetic and real-world datasets, we demonstrate that asynchronous implementations retain comparable solution accuracy and fea-

A. Nwachukwu is with Olin Business School, Washington University in St. Louis, USA (e-mail: anthonyn@wustl.edu).

A. Karbowski is with Institute of Control and Computation Engineering, Warsaw University of Technology, Warsaw, Poland (e-mail: andrzej.karbowski@pw.edu.pl).

sibility while yielding substantial reductions in runtime. The results confirm the robustness and versatility of Bertsekas-style augmented Lagrangian decomposition for combinatorial and continuous optimization problems in distributed and delay-prone environments.

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

We can describe the problem of optimizing routing and flow rates simultaneously 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:

TABLE I: Network problem notation

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, we assume 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 [14]:

$$\min_{x,b,y} \quad \sum_{w \in W} \left[\gamma \left(\bar{x}_w - x_w \right)^2 + \delta \sum_{l \in L} b_{wl} \right] \tag{1}$$

subject to (s.t.)

$$\sum_{\{i \in N | (i,j) \in A\}} y_{w,e(i,j)} - \sum_{\{k \in N | (j,k) \in A\}} y_{w,e(j,k)} = \begin{cases} -x_w & j = S(w) \\ x_w & j = D(w) \\ 0 & \text{otherwise} \end{cases} \quad \forall w \in W, \forall j \in N \tag{2}$$

$$\underline{x}_w \le x_w \le \bar{x}_w, \qquad \forall w \in W \tag{3}$$

$$\sum_{w \in W} y_{wl} \le c_l, \qquad \forall l \in L \tag{4}$$

$$y_{wl} \ge 0, \qquad \forall w \in W, \forall l \in L \tag{5}$$

$$y_{wl} \le b_{wl} \bar{x}_w, \qquad \forall w \in W, \forall l \in L \tag{6}$$

$$\sum_{\{j \in N | (i,j) \in A\}} b_{w,e(i,j)} \le 1, \qquad \forall w \in W, \forall i \in N \tag{7}$$

$$b_{wl} \in \{0,1\}, \qquad \forall w \in W, \forall l \in L \tag{8}$$

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 formulation, denoted as P_{alt} in [14], features a quadratic and convex objective function, as expressed in (1). The first term, $f_w(x_w, b_w)$, represents the cost incurred when a connection $w \in W$ fails to achieve its maximum possible bandwidth. The second term accounts for the total energy consumption of the connection.

The flow conservation is modeled using auxiliary real variables y_{wl} and binary variables b_{wl}, subject to the equality constraint in (2) and three inequalities: (7), (5), and (6). The constraint (7) ensures single-path routing, while (6) enforces the relationship between the auxiliary and binary variables.

To solve the problem (1)-(8), we decompose it with respect to the flows $w \in W$. For this purpose, we introduce the admissible set XBY_w for each flow w, which results from the constraints (2), (3), (5)-(8). This set for flow w is defined as

$$XBY_w = \left\{ \begin{array}{l} x_w, y_{wl} \in \mathbb{R}, b_{wl} \in \{0,1\}, l \in L : \\ \sum_{\{i \in N | (i,j) \in A\}} y_{w,e(i,j)} \\ - \sum_{\{k \in N | (j,k) \in A\}} y_{w,e(j,k)} \\ \qquad = \begin{cases} -x_w & j = S(w) \\ x_w & j = D(w) \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in N, \\ \underline{x}_w \le x_w \le \bar{x}_w, \\ y_{wl} \ge 0, \quad \forall l \in L, \\ y_{wl} \le b_{wl} \bar{x}_w, \quad \forall l \in L \\ \sum_{\{j \in N | (i,j) \in A\}} b_{w,e(i,j)} \le 1, \quad \forall i \in N \end{array} \right\} \tag{9}$$

Now we can write (1)-(8) as,

$$\min_{(x,b,y) \in XBY} \quad \sum_{w \in W} \left[\gamma \left(\bar{x}_w - x_w \right)^2 + \delta \sum_{l \in L} b_{wl} \right] \tag{10}$$

$$s.t. \quad \sum_{w \in W} y_{wl} - c_l \le 0, \quad \forall l \in L \tag{11}$$

A. Synchronous Iterative Updates

The augmented Lagrangian in Bertsekas method for the problem (1)-(8) with $x_w^s \in \mathbb{R}_+ \cup \{0\}$ and $b_{wl}^s \in \{0,1\}$ will have the form

$$
\begin{aligned}
&L_\rho(x,b,y,x^s,b^s,y^s,\mu)\\
&\quad=\sum_{w\in W}\Bigg\{\gamma\big(\bar{x}_w-x_w\big)^2+\frac{\rho}{2}\,(x_w-x_w^s)^2\\
&\quad+\sum_{l\in L}\Bigg[\delta\; b_{wl}+\mu_l\left(y_{wl}-\frac{c_l}{|w|}\right)\\
&\quad+\frac{\rho}{2}\left((b_{wl}-b_{wl}^s)^2+(y_{wl}-y_{wl}^s)^2\right)\Bigg]\Bigg\} \qquad (12)\\
&\quad\triangleq\sum_{w\in W}\Bigg\{L_{\rho 1_w}(x_w,x_w^s)\\
&\quad+\sum_{l\in L}L_{\rho 2_{wl}}(b_{wl},y_{wl},b_{wl}^s,y_{wl}^s,\mu_l)\Bigg\}
\end{aligned}
$$

where

$$
L_{\rho 1_w}(x_w,x_w^s)=\gamma\big(\bar{x}_w-x_w\big)^2+\frac{\rho}{2}\,(x_w-x_w^s)^2 \qquad (13)
$$

$$
\begin{aligned}
&L_{\rho 2_{wl}}(b_{wl},y_{wl},b_{wl}^s,y_{wl}^s,\mu_l)\\
&\quad=\delta\; b_{wl}+\mu_l\left(\sum_{w\in W}y_{wl}-\frac{c_l}{|w|}\right)\\
&\quad+\frac{\rho}{2}\left[(b_{wl}-b_{wl}^s)^2+(y_{wl}-y_{wl}^s)^2\right] \qquad (14)
\end{aligned}
$$

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

$$
\begin{aligned}
&(x_w^{k+1},b_w^{k+1},y_w^{k+1})\\
&\quad=\underset{(x_w,b_w,y_w)\in XBY_w}{\arg\min}\Bigg\{L_{\rho 1_w}(x_w,x_w^{s^k})\\
&\quad+\sum_{l\in L}L_{\rho 2_{wl}}(b_{wl},y_{wl},b_{wl}^{s^k},y_{wl}^{s^k},\mu_l^k)\Bigg\},\ \forall\, w\in W \qquad (15)\\
&x_w^{s^{k+1}}=\xi_k\,x_w^{s^k}+(1-\xi_k)x_w^{k+1},\quad \forall\, w\in W \qquad (16)\\
&b_{wl}^{s^{k+1}}=\xi_k\,b_{wl}^{s^k}+(1-\xi_k)b_{wl}^{k+1},\quad \forall\, w\in W,\ l\in L \qquad (17)\\
&y_{wl}^{s^{k+1}}=\xi_k\,y_{wl}^{s^k}+(1-\xi_k)y_{wl}^{k+1},\quad \forall\, w\in W,\ l\in L \qquad (18)\\
&\mu_l^{k+1}=\max\left\{0,\quad \mu_l^k+\beta\rho\Big(\sum_{w\in W}y_{wl}^{k+1}-c_l\Big)\right\},\\
&\quad\forall\, l\in L \qquad (19)
\end{aligned}
$$

B. Asynchronous Iterative Updates

Let $S_w\subseteq\{1,2,\dots,\}$ 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 $T_l\subseteq\{1,2,\dots,\}$ 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 (1)-(8) will be solved using a similar approach found in [9] and the Augmented Lagrangian function defined in (12).

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

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

1: From iteration to iteration, w receives updates μ_l^k from all $l\in L$ and calculates $\hat{\mu}_l^k$.

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 $x_w^{s^{k+1}},b_w^{s^{k+1}},y_w^{s^{k+1}}$

$$
\begin{aligned}
&(x_w^{k+1},b_w^{k+1},y_w^{k+1})\\
&\quad=\underset{(x_w,b_w,y_w)\in XBY_w}{\arg\min}\Bigg\{L_{\rho 1_w}(x_w,x_w^{s^k})\\
&\quad+\sum_{l\in L}L_{\rho 2_{wl}}(b_{wl},y_{wl},b_{wl}^{s^k},y_{wl}^{s^k},\hat{\mu}_l^k)\Bigg\} \qquad (20)\\
&x_w^{s^{k+1}}=\xi_k\,x_w^{s^k}+(1-\xi_k)x_w^{k+1} \qquad (21)\\
&b_{wl}^{s^{k+1}}=\xi_k\,b_{wl}^{s^k}+(1-\xi_k)b_{wl}^{k+1},\quad \forall\, l\in L \qquad (22)\\
&y_{wl}^{s^{k+1}}=\xi_k\,y_{wl}^{s^k}+(1-\xi_k)y_{wl}^{k+1},\quad \forall\, l\in L \qquad (23)
\end{aligned}
$$

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, $k=1,2,...,$ *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

$$
\mu_l^{k+1}=\max\left\{0,\quad \mu_l^k+\beta\rho\Big(\sum_{w\in W}\hat{y}_{wl}^{k+1}-c_l\Big)\right\} \qquad (24)
$$

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

$$
\hat{y}_{wl}^k=\sum_{k'=k-k^0}^{k}a_{wl}(k',k)y_{wl}^{k'},\quad w\in W\ l\in L \qquad (25)
$$

$$
\hat{\mu}_l^k=\sum_{k'=k-k^0}^{k}b_l(k',k)\mu_l^{k'},\quad l\in L \qquad (26)
$$

and

$$
\sum_{k'=k-k^0}^{k}a_{wl}(k',k)=1,\quad \forall k,\ w\in W\ l\in L \qquad (27)
$$

$$
\sum_{k'=k-k^0}^{k}b_l(k',k)=1,\quad \forall k,\ l\in L \qquad (28)
$$

and

$$
\begin{aligned}
a_{wl}(k',k)\geq 0,\quad \forall k,\ w\in W\ l\in L \qquad (29)\\
b_l(k',k)\geq 0,\quad \forall k,\ l\in L \qquad (30)
\end{aligned}
$$

III. Synchronous and Asynchronous Solutions of Regularized Systems of Linear Equations

Solving linear systems reliably is a central task across diverse areas, including optimization, machine learning, signal processing, and statistical modeling. In practice, the raw system $Ax=b$, where $A\in\mathbb{R}^{m\times n}$, $x\in\mathbb{R}^n$, and $b\in\mathbb{R}^m$, often arises under conditions that make direct solutions unstable or computationally demanding. Here, m and n denote the number

of observations and variables, respectively. Depending on the relationship between m and n, the system may be underdetermined, overdetermined, or square; in each case, challenges such as ill-conditioning, multicollinearity, and sensitivity to noise frequently occur.

To address these concerns, regularization methods are introduced to improve numerical stability and generalization. Among these, ℓ_2-based regularization is particularly common because it discourages excessively large coefficients without enforcing sparsity. Unlike ℓ_0 or its convex surrogate ℓ_1, the ℓ_2 penalty encourages smooth solutions and reduces the risk of overfitting, making it well-suited for high-dimensional and noisy settings. This principle underpins methods such as ridge regression and Tikhonov regularization, which remain foundational in modern computational approaches.

To describe this formally, let $P \subseteq \{1, \ldots, n\}$ index the set of decision variables and $M = \{1, \ldots, m\}$ represent the set of constraints. Define $a_i \in \mathbb{R}^{|P|}$ as the feature vector corresponding to the i-th observation and $y_i \in \mathbb{R}$ as its associated response. The regularized formulation can then be expressed as the following constrained ℓ_2-minimization problem:

$$\min_{x} \quad \sum_{j \in P} x_j^2, \quad \text{s.t.} \sum_{j \in P} A_{ij} x_j = y_i, \quad \forall i \in M \tag{31}$$

In this formulation, $x_j \in \mathbb{R}$ for all $j \in P$ are the optimization variables, and the equality constraints guarantee that the model fits the observed data exactly. The objective incorporates an ℓ_2 penalty that encourages smaller coefficients and enhances robustness against noise and multicollinearity. This framework forms the basis for well-known methods such as ridge regression and Tikhonov regularization and remains fundamental to modern optimization-based techniques for linear inverse problems.

To facilitate the use of algorithms that operate on inequality constraints, the equality constraints are equivalently rewritten as a pair of inequalities; hence $\forall i \in M$, (31) becomes:

$$\min_{x} \quad \sum_{j \in P} x_j^2, \quad \text{s.t.} \begin{cases} \sum_{j \in P} A_{ij} x_j - y_i - \varepsilon \leq 0, \\ -\sum_{j \in P} A_{ij} x_j + y_i - \varepsilon \leq 0 \end{cases} \tag{32}$$

where ε is a small relaxing 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.

A. Synchronous Iterative Updates

The augmented Lagrangian in Bertsekas method for the inequality-constrained regularized systems of linear equations problem (32) takes the form:

$$\begin{aligned} L_\rho(x, x^s, \mu^+, \mu^-) = \sum_{j \in P} \Bigg\{ & x_j^2 + \frac{\rho}{2}(x_j - x_j^s)^2 \\ & + \sum_{i \in M} \Bigg[\mu_i^+ \Big(\sum_{j \in P} A_{ij} x_j - y_i - \varepsilon \Big) \\ & + \mu_i^- \Big(- \sum_{j \in P} A_{ij} x_j + y_i - \varepsilon \Big) \Bigg] \Bigg\} \end{aligned} \tag{33}$$

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

$$L_{\rho_j}(x_j, x_j^s, \mu^+, \mu^-) = x_j^2 + \frac{\rho}{2}(x_j - x_j^s)^2 + \sum_{i \in M} A_{ij} x_j (\mu_i^+ - \mu_i^-) \tag{34}$$

plus components dependent on Lagrange multipliers μ_i^+, μ_i^- and constants ε, $y_i, i \in M$.

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

$$x_j^{k+1} = \arg\min_{x_j} L_{\rho_j}(x_j, x_j^{s^k}, \mu^{+,k}, \mu^{-,k}) \tag{35}$$

$$x_j^{s^{k+1}} = \xi_k x_j^{s^k} + (1 - \xi_k) x_j^{k+1} \tag{36}$$

$$\mu_i^{+,k+1} = \max \left\{ 0, \mu_i^{+,k} + \beta\rho \left(\sum_{j \in P} A_{ij} x_j^{k+1} - y_i - \varepsilon \right) \right\} \tag{37}$$

$$\mu_i^{-,k+1} = \max \left\{ 0, \mu_i^{-,k} + \beta\rho \left(- \sum_{j \in P} A_{ij} x_j^{k+1} + y_i - \varepsilon \right) \right\} \tag{38}$$

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.

B. Asynchronous Iterative Updates

Let $S_j \subseteq \{1, 2, \ldots\}$ 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 $T_i \subseteq \{1, 2, \ldots\}$ 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 [9] and the augmented Lagrangian function defined in (33), the asynchronous updates proceed as follows.

j's Algorithm: At iterations $k = 1, 2, \ldots$, 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 $x_j^{s^{k+1}}$:

$$x_j^{k+1} = \arg\min_{x_j} L_{\rho_j}(x_j, x_j^{s^k}, \hat{\mu}^{+,k}, \hat{\mu}^{-,k}), \quad \forall j \in P \tag{39}$$

$$x_j^{s^{k+1}} = \xi_k \, x_j^{s^k} + (1 - \xi_k) \, x_j^{k+1}, \quad \forall j \in P \tag{40}$$

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 $k = 1, 2, \ldots$, 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:

$$\mu_i^{+,k+1} = \max\left\{0,\ \mu_i^{+,k} + \beta\rho\left(\sum_{j\in P} A_{ij}\hat{x}_j^{k+1} - y_i - \varepsilon\right)\right\} \tag{41}$$

$$\mu_i^{-,k+1} = \max\left\{0,\ \mu_i^{-,k} + \beta\rho\left(-\sum_{j\in P} A_{ij}\hat{x}_j^{k+1} + y_i - \varepsilon)\right)\right\} \tag{42}$$

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:

$$\hat{x}_j^k = \sum_{k'=k-k_0}^{k} a_j(k',k)x_j^{k'}, \quad \forall j \in P \tag{43}$$

$$\hat{\mu}_i^{+,k} = \sum_{k'=k-k_0}^{k} b_i(k',k)\mu_i^{+,k'}, \quad \forall i \in M \tag{44}$$

$$\hat{\mu}_i^{-,k} = \sum_{k'=k-k_0}^{k} b_i(k',k)\mu_i^{-,k'}, \quad \forall i \in M \tag{45}$$

with the weights satisfying:

$$\sum_{k'=k-k_0}^{k} a_j(k',k) = 1, \quad a_j(k',k) \geq 0, \quad \forall k,\ j \in P \tag{46}$$

$$\sum_{k'=k-k_0}^{k} b_i(k',k) = 1, \quad b_i(k',k) \geq 0, \quad \forall k,\ i \in M \tag{47}$$

IV. Numerical Tests

A. 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. Network construction and visualization tasks were handled using the NetworkX library [15], facilitating the representation of graph-based structures integral to the optimization problems.

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 $merge\_asof$ from Pandas [16], 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 methods 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 version 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.

B. Network Optimization Problem

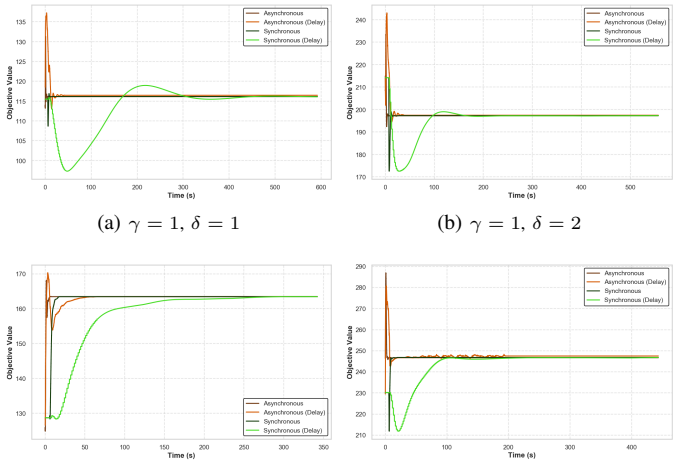
1) Dataset Description: The algorithms developed in this study were implemented and evaluated on three network topologies of increasing complexity: medium, large, and extra-large. Each topology was tested under multiple objective parameter configurations (γ, δ), yielding ten problem instances. The networks were designed as loosely connected clusters of nodes exhibiting strong intra-cluster connectivity. Each instance was solved using a synchronous and an asynchronous formulation to evaluate comparative performance. All experiments were executed in parallel using Python's `multiprocessing.Pool` to simulate distributed computation. The detailed characteristics of the tested network instances are summarized in Table II.

2) Results and Discussion: In Table III, 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-

TABLE II: Summary of network problem instances tested

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

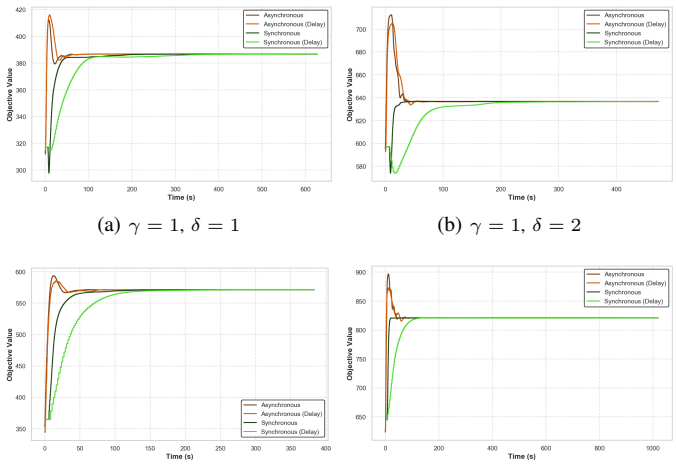
offs between the two algorithmic paradigms. They can be visualized in Figs. 1–5.



(a) $\gamma = 1, \delta = 1$ (b) $\gamma = 1, \delta = 2$

(c) $\gamma = 2, \delta = 1$ (d) $\gamma = 2, \delta = 2$

Fig. 1: Medium Problem Objective Values (Routing)

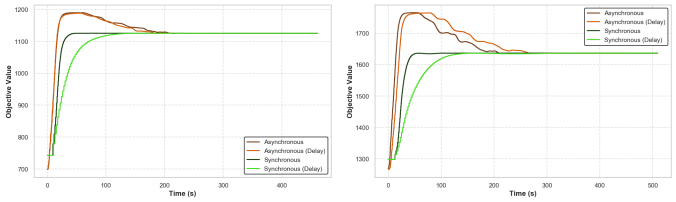


(a) $\gamma = 1, \delta = 1$ (b) $\gamma = 1, \delta = 2$

(c) $\gamma = 2, \delta = 1$ (d) $\gamma = 2, \delta = 2$

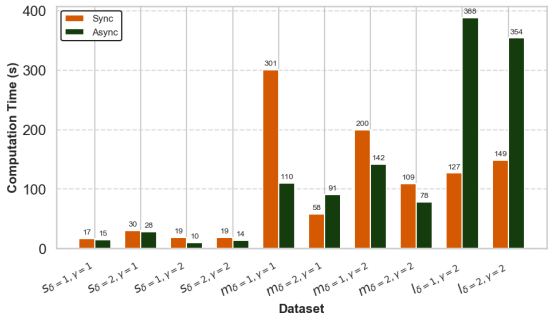
Fig. 2: Large Problem Objective Values (Routing)

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

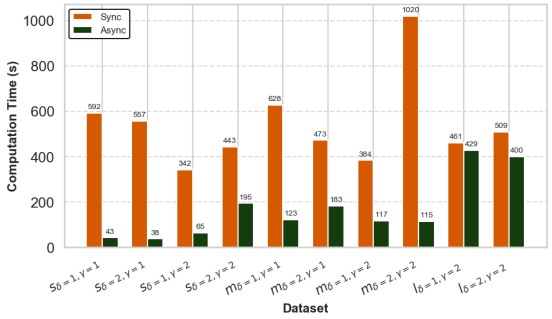


(a) $\gamma = 1, \delta = 1$ (b) $\gamma = 1, \delta = 2$

Fig. 3: Extra Large Problem Objective Values (Routing)



(a) Without Delay



(b) With Delay

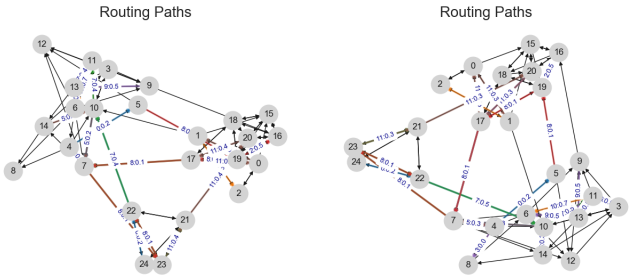
Fig. 4: Run Time (Routing)

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

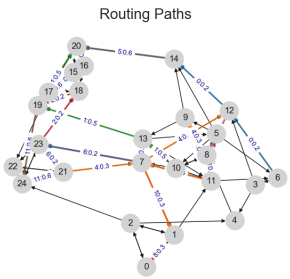
TABLE III: Routing Results

Data	Objective		Objective (with delay)		Time [s]		Time (with delay) [s]	
	Sync	Async	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

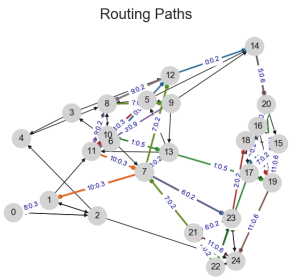


(a) Sync $\gamma = 1$, $\delta = 1$

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



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



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

Fig. 5: Small Problem Networks (Routing)

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 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.

C. Regularized Linear Systems

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 $x^\star$.

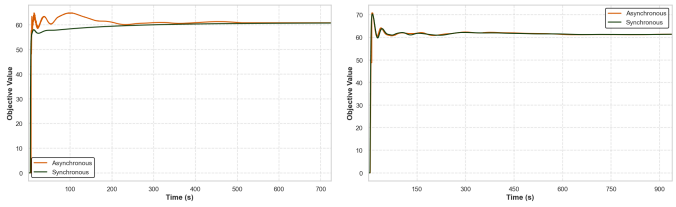
The first dataset simulates a diagonal sensing problem in grayscale image recovery. A 16×16 grayscale image is vectorized into $x^\star \in \mathbb{R}^{256}$, and a diagonal measurement matrix $A \in \mathbb{R}^{256\times 256}$ is constructed with entries drawn independently from a uniform distribution 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 magnitudes. The measurement vector $y = Ax^\star$, therefore, reflects a noisier and more variable scaling of the original image, posing greater

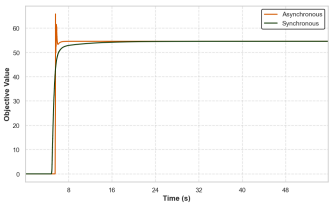
challenges for algorithmic recovery under unstable and zero-mean multiplicative transformations.

The third dataset is derived from the UCI Breast Cancer Wisconsin (Diagnostic) dataset [17] 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 $A \in \mathbb{R}^{10\times 30}$ 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.

2) Results and Discussion: Table IV and Figs. 6–10 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.



(a) Uniform (b) Gaussian



(c) Cancer Spars

Fig. 6: Async-Sync: Objectives

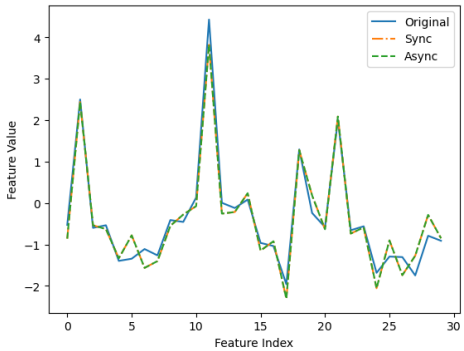


Fig. 7: Async-Sync: Compressed Measurements of Cancer Spars

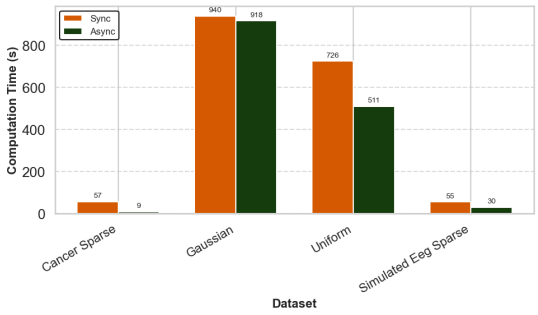
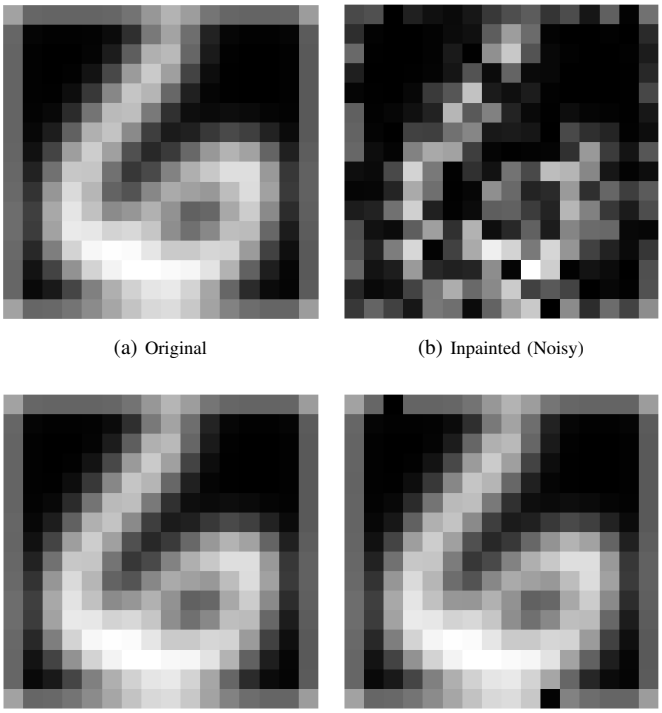


Fig. 8: Async-Sync: Run Time (secs)



(a) Original (b) Inpainted (Noisy)

(c) Synchronous (d) Asynchronous

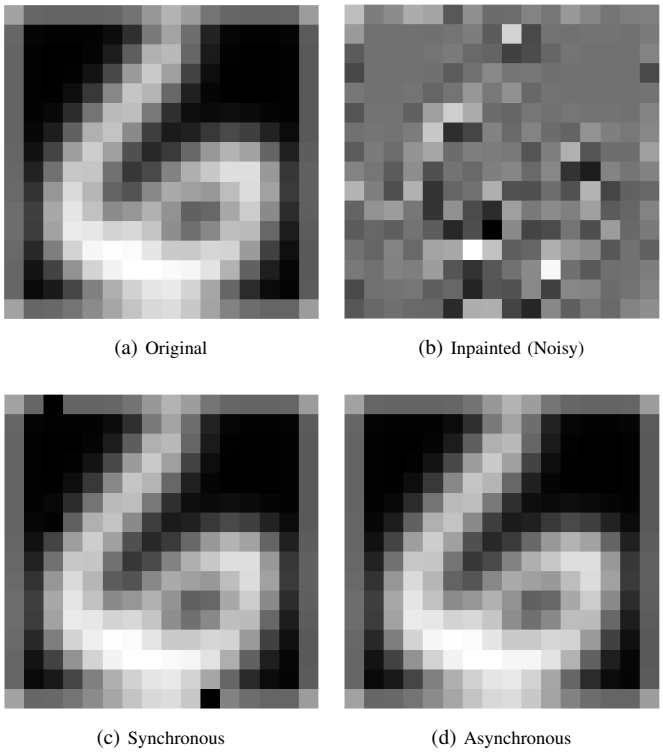
Fig. 9: Image Recovery under Gaussian Additive Noise

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 scenar-

TABLE IV: 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) Original (b) Inpainted (Noisy)

(c) Synchronous (d) Asynchronous

Fig. 10: Image Recovery under Gaussian Additive Noise

ios 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.

V. Conclusions and Future Work

This study has developed and tested Bersekas Augmented Lagrangian decomposition algorithms (Synchronous and Asynchronous) tailored to the simultaneous optimization of routing and bandwidth allocation in energy-aware networks and the solutions of regularized linear systems in large-scale machine learning contexts. The proposed techniques addressed the challenges posed by the mixed-integer nature of routing problems and the high dimensionality of the regularized linear systems. The resulting methods achieved efficient and scalable computation by employing decomposition and dualization strategies. Notably, asynchronous implementations demonstrated superior resilience to delays and significantly improved computational performance compared to their synchronous counterparts, without any substantial degradation in solution quality.

Beyond empirical validation, this work provides evidence for the robustness and adaptability of asynchronous optimization paradigms in large-scale and distributed systems. In the context of network design, our method overcomes traditional limitations of Lagrangian relaxation by producing feasible, high-quality routing and flow allocations. In regularized linear systems, the asynchronous update mechanisms offer considerable reductions in wall-clock time, especially when operating in heterogeneous computing environments.

Looking forward, this framework opens several avenues for further research. One direction involves extending the network model to allow K-path routing and incorporating stochastic or uncertain demand profiles, reflecting more realistic operating conditions. Another promising line of inquiry is refining the convergence theory of asynchronous methods, especially under relaxed assumptions such as unbounded delays or non-uniform update frequencies. Additionally, dynamic penalty parameter tuning and adaptive step-size mechanisms may accelerate convergence. Finally, deployment on physical networking hardware or distributed edge platforms would provide practical insights into the viability of these methods in real-time applications. These directions aim to advance the use of augmented Lagrangian methods as a foundation for efficient, scalable, and decentralized optimization in networked and data-rich environments.

References

[1] E. Gelenbe, "Energy packet networks: adaptive energy management for the cloud," in *Proceedings of the 2nd International Workshop on Cloud Computing Platforms*, ser. EuroSys '12. ACM, apr 2012, p. 1–5. [Online]. Available: http://doi.org/10.1145/2168697.2168698

[2] B. Wang and P.-H. Ho, "Energy-efficient routing and bandwidth allocation in OFDM-based optical networks," *Journal of Optical Communications and Networking*, vol. 8, no. 2, p. 71, jan 2016. [Online]. Available: http://doi.org/10.1364/JOCN.8.000071

[3] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surveys in Operations Research and Management Science*, vol. 17, no. 2, p. 97–106, 7 2012. [Online]. Available: http://doi.org/10.1016/j.sorms.2012.08.001

[4] I. E. Grossmann and Z. Kravanja, *Mixed-Integer Nonlinear Programming: A Survey of Algorithms and Applications*. Springer New York, 1997, p. 73–100. [Online]. Available: http://doi.org/10.1007/978-1-4612-1960-6_5
[5] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-posed Problems*. New York: Wiley, 1977.
[6] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. Philadelphia: SIAM, 1998.
[7] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall Inc., 1989.
[8] ——, "Some aspects of parallel and distributed algorithms: A survey," *Automatica*, vol. 27, no. 1, pp. 3–21, 1991.
[9] S. Low and D. Lapsley, "Optimization flow control. I. Basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, p. 861–874, 1999. [Online]. Available: https://ieeexplore.ieee.org/document/811451
[10] A. Karbowski, "Comments on "Optimization flow control. I. Basic algorithm and convergence"," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, p. 338–339, 4 2003. [Online]. Available: https://ieeexplore.ieee.org/document/1194828
[11] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, "Asynchronous distributed ADMM for large-scale optimization—part I: Algorithm and convergence analysis," in *IEEE Transactions on Signal Processing*, vol. 64, 2016, pp. 118–132.
[12] E. Wei and A. Ozdaglar, "On the O(1/k) convergence of asynchronous distributed alternating direction method of multipliers," 2013. [Online]. Available: https://arxiv.org/abs/1307.8254
[13] A. C. Nwachukwu and A. Karbowski, "Solution of the simultaneous routing and bandwidth allocation problem in energy-aware networks using Augmented Lagrangian-based algorithms and decomposition," *Energies*, vol. 17, no. 5, p. 1233, 3 2024. [Online]. Available: http://doi.org/10.3390/en17051233
[14] 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, 10 2022. [Online]. Available: https://doi.org/10.3390/en15207634
[15] "Networkx - network analysis in Python," https://networkx.org/, accessed: 2024-02-12.
[16] The Pandas Development Team, "pandas.merge_asof — pandas documentation," 2024, accessed: 2025-06-20. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.merge_asof.html
[17] W. H. Wolberg and O. L. Mangasarian, "Breast cancer Wisconsin (diagnostic)," 1993. [Online]. Available: https://doi.org/10.24432/C5DW2B