Distributed and Asynchronous Coordination of a Mixed-Integer Linear System via Surrogate Lagrangian Relaxation

Mikhail A. Bragin, *Member, IEEE*, Bing Yan, *Member, IEEE*, and Peter B. Luh, *Life Fellow, IEEE*

***Abstract*— With the emergence of the Internet of Things that allows communications and local computations and with the vision of Industry 4.0, a foreseeable transition is from centralized system planning and operation toward decentralization with interacting components and subsystems, e.g., self-optimizing factories. In this article, a new "price-based" decomposition and coordination methodology is developed to efficiently coordinate a system consisting of distributed subsystems such as machines and parts, which are described by mixed-integer linear programming (MILP) formulations, in an asynchronous way. The novel method is a dual approach, whereby the coordination is performed by updating Lagrangian multipliers based on economic principles of "supply and demand." To ensure low communication requirements within the method, exchanges between the "coordinator" and subsystems are limited to "prices" (Lagrangian multipliers) broadcast by the coordinator and to subsystem solutions sent at the coordinator. Asynchronous coordination, however, may lead to convergence difficulties since the order in which subsystem solutions arrive at the coordinator is not predefined as a result of uncertainties in communication and solving times. Under realistic assumptions of finite communication and solve times, the convergence of our method is proven by innovatively extending the Lyapunov stability theory. Numerical testing of generalized assignment problems through simulation demonstrates that the method converges fast and provides near-optimal results, paving the way for self-optimizing factories in the future. Accompanying CPLEX codes and data are included.**

***Note to Practitioners*—In view of a foreseeable transition toward self-optimizing factories whereby machines and parts have communication and computational capabilities, a novel "price-based" distributed and asynchronous method to coordinate a system consisting of distributed subsystems is developed. Under realistic assumptions of finite communication and solve times, method convergence is proven. Numerical testing of generalized assignment problems through simulation demonstrates that the method converges fast and provides near-optimal results, paving the way for self-optimizing factories in the future. Accompanying CPLEX codes and data are included.**

***Index Terms*— Distributed and asynchronous algorithms, mixed-integer linear programming (MILP) problems, self-optimizing factories, surrogate Lagrangian relaxation (LR).**

I. Introduction

With the emergence of the Internet of Things [1], [2] empowered by smart sensors together with advanced computation and communication technologies and with the vision of Industry 4.0 [3], [4], a foreseeable transition is from centralized system planning and operation toward decentralization. For example, within self-optimizing factories, a system will consist of multiple distributed and interacting components/subsystems that need to be coordinated. Within these futuristic factories, distributed subsystems, such as machines and parts, are coordinated through 5G networks to meet certain objectives, such as on-time delivery. In manufacturing, examples of operations optimization problems include planning, scheduling, and dispatching problems [5], [6]. Scheduling problems are solved before each shift and require short solving times, such as a few minutes, and online dispatching of a part to a machine may require a few seconds. Because of the many possible interconnections among parts, operations, and machines, an efficient communication scheme is required to prevent bandwidth overloading. This motivates the need for efficient, coordinated operations while ensuring high computational and communication efficiency.

A system consisting of subsystems are frequently formulated as mixed-integer linear programming (MILP) subproblems. For complicated systems, the complexity of MILP problems is a serious issue because of the presence of integer decision variables, and the goal of obtaining high-quality solutions within short times as delineated earlier, typically cannot be met. Nevertheless, the structures of these systems and the associated MILP problems are amenable to decomposition into individual MILP subproblems associated with corresponding subsystems with drastically reduced complexity. Traditionally, to coordinate subproblems, price-based decomposition and coordination Lagrangian relaxation (LR) method [7]–[11]

Manuscript received November 5, 2019; revised March 29, 2020; accepted May 18, 2020. This article was recommended for publication by Associate Editor R. Cordone and Editor S. A. Reveliotis upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation (NSF) under Grant ECCS-1509666, Grant ECCS-1810108, Grant CNS-1647209, and Grant ECCS-1831811. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. *(Corresponding author: Mikhail A. Bragin.)*

Mikhail A. Bragin and Peter B. Luh are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269-4157 USA (e-mail: mikhail.bragin@uconn.edu; peter.luh@uconn.edu).

Bing Yan is with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: bxyeee@rit.edu).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2020.2998048

has been used by exploiting problem separability in manufacturing problems, such as job-shop scheduling [10]. The LR method is a dual approach, whereby the coordination is performed by updating Lagrangian multipliers based on economic principles of "supply and demand." Multipliers (or "shadow prices") are updated based on levels of violation of relaxed constraints by using subgradient methods [12], [13]. Because of the exploitation of decomposability, the method is a good candidate for coordinating distributed subsystems whereby a coordinator updates multipliers and only needs to know solutions of subproblems associated with distributed subsystems. However, standard LR methods suffer from major convergence difficulties, such as high computational effort, zigzagging of multipliers, and the need to know the optimal dual values. Moreover, since standard LR requires solving all subproblems to update multipliers, the method is synchronous. When the number of subproblems is large, synchronous coordination may lead to inefficient time management since "fast" subproblem solvers will likely spend a significant amount of time waiting for synchronization.

Some the above-mentioned difficulties have been overcome within several versions of LR, such as incremental subgradient methods [14], [15], alternate direction method of multipliers (ADMM) [16]–[21], surrogate subgradient method [22], and surrogate Lagrangian relaxation (SLR) [23], [24], [48], to be reviewed in Section II. The distributed and asynchronous incremental subgradient method [15] for optimizing convex dual functions consisting of a large number of components, which arise within the LR framework with a large number of subproblems, overcomes the synchronization difficulty. However, the method may be slow when there are both "slow" and "fast" subsystems since the method requires that all subproblem solutions arrive at the coordinator with the same "long-term" frequency, on average. ADMM [16]–[21], a decomposable version of the method of multipliers (frequently referred to as "augmented LR" (ALR) [25], [26]), aims at accelerating convergence of traditional LR by penalizing constraint violations by using quadratic penalty terms and by decomposing relaxed problems arising in ALR to reduce computational effort. However, when it comes to coordination of MILP subproblems, neither synchronous nor asynchronous versions of ADMM converge.

Our recent SLR method [23], [24], [48] has overcome major convergence difficulties of standard LR, such as high computational effort, zigzagging of multipliers, and the need to know the optimal dual value for convergence. Moreover, as demonstrated in [24, Fig. 1, p. 537], the method outperforms another coordination method—ADMM. In [48], it has been demonstrated that the surrogate ALR method, which is built upon the SLR method, is capable of efficiently coordinating thousands of subsystems. The method has, thus, been demonstrated to be powerful, and the asynchronous functionality will be added to efficiently coordinate distributed subsystems to be discussed next.

In this article, a novel distributed and asynchronous price-based decomposition and coordination method based upon our recent SLR method [23] is developed in Section III to efficiently coordinate a system consisting of distributed MILP subsystems within futuristic self-optimizing factories while overcoming difficulties associated with other dual methods mentioned earlier. Within the new method, multiple distributed subsystems and one coordinator have computation and communication capabilities. To avoid excessive data transfer within the system, information exchanges between the coordinator and subsystems are limited to: 1) "prices" (Lagrangian multipliers) broadcast by the coordinator and to 2) subsystem solutions sent at the coordinator. While asynchronous coordination avoids the synchronization issue, it leads to major convergence difficulties: 1) because of uncertainties in solving, communication, and multiplier-updating times, the order in which subsystem solutions arrive at the coordinator is uncertain and 2) subsystem solutions are obtained based on multipliers of different vintages, and multipliers may not converge. To overcome these difficulties while ensuring fast speed, rather than requiring the "long-term" frequency requirement as in [15], convergence is proven under a "freshness" assumption, whereby a coordinator can update multipliers without waiting for "slow" subproblems as long as all subproblem solutions arrive at the coordinator at least once within a finite number of iterations. Our idea to establish convergence is through the novel use of the Lyapunov energy function defined as the square of the distance from the current prices to the optimum with the idea of forcing this function to approach zero thereby ensuring that prices approach their optimal values. Although not monotonically decreasing as required by traditional Lyapunov methods for convergence [27], an upper bound is innovatively established as an envelope of Lyapunov functions for all possible (uncertain) trajectories of multipliers ("prices") that result from uncertain sequences of subproblem solutions arriving at the coordinator. Based on the contraction mapping concept whereby distances between multipliers at consecutive iterations decrease, it is then proved in the main theorem and several supporting propositions that these upper bound approaches zero, thereby leading to convergence.

In Section IV, by simulating asynchronous updates of multipliers, two examples are presented. The first small example is to show that Lyapunov functions within the new method while nonmonotonic, approach zero fast. The second example is based on generalized assignment problems (GAPs), which can be viewed as simplified problems that arise within factories. These results demonstrate that the new method converges fast. With such effective distributed and asynchronous coordination, the method has valuable implications for future self-optimizing factories to coordinate machines or parts.

II. Literature Review

SLR is reviewed in Section II-A. In Section II-B, other existing price-based decomposition and coordination approaches, such as the distributed asynchronous incremental subgradient method, and asynchronous ADMM, both are versions of LR tailored for asynchronous coordination, are reviewed, and their limitations are presented. In Section II-C, our recent SLR is reviewed as a promising approach for the development of an efficient asynchronous coordination method. Since this article deals with the coordination of MILP subsystems that

use branch-and-cut to solve their subproblems, branch-and-cut is reviewed in Section II-D. Methods that do not support distributed coordination, such as heuristics methods, or the distributed methods that require continuity of problems are not reviewed.

A. Standard Lagrangian Relaxation

Traditionally, to solve MILP problems, LR [7]–[11] has been used to exploit problem separability. Specifically, in manufacturing, to solve job-shop scheduling, machine capacity coupling constraints are relaxed to decompose the problem into part subproblems [10]. The LR method is a dual approach whereby the coordination is performed by updating Lagrangian multipliers based on economic principles of "supply and demand." Within standard LR, multipliers (or "shadow prices") are updated after receiving subproblem solutions and based on levels of violation of relaxed constraints by using subgradient methods [12], [13]. Because of the exploitation of decomposability, the LR method is a good candidate for coordinating distributed subsystems whereby a coordinator updates multipliers and only needs to know solutions of subproblems associated with distributed subsystems. However, standard LR methods suffer from major convergence difficulties. Because of the presence of discrete variables, the dual function is nonsmooth polyhedral concave [28, p. 670, Proposition 7.1.2]. Therefore, gradients may not exist, and subgradients are used. Moreover, ridges of the dual function may be sharp. Since the method requires solving all subproblems, because of the sharp ridges, subgradient directions may change drastically from one iteration to the next. As a result, multipliers suffer from zigzagging across ridges of the dual function [23, p. 192, Fig. 1], [29, p. 594, Fig. 1]. In addition, the convergence proof and as practical implementations require the knowledge of the optimal dual value, which is unknown and is typically adaptively adjusted in practice as in "subgradient-level" methods [30] or incremental subgradient methods [31]. However, these adjustments are inefficient, and convergence is slow, as demonstrated in [23, pp. 195–196 and 199, Figs. 3–5 and 7].

B. Distributed and Asynchronous Coordination Methods

1) Distributed Asynchronous Incremental Subgradient Method: To optimize nonsmooth dual functions consisting of a large number of components, which arise within the LR framework, in a distributed and asynchronous manner, a distributed asynchronous incremental subgradient method was developed [15]. The method requires that all subproblem solutions arrive at the coordinator with the same "long-term" frequency on average, and convergence was proven using the diminishing stepsizing rule. Moreover, convergence was proven under the assumption that the subgradient is split into individual components, and each component is updated independently rather than updating the subgradient as a whole. Under this scheme, convergence may be slow in situations whereby there are "fast" and "slow" subsystems solvers because "fast" subsystems may spend significant amounts of time waiting to satisfy the "long-term" frequency requirement.

2) Alternate Direction Method of Multipliers: ADMM, a decomposable version of the method of multipliers [25], [26] (frequently referred to as ALR), aims at accelerating convergence of traditional LR by penalizing constraint violations by using quadratic penalty terms and by decomposing relaxed problems arising in ALR to reduce computational effort. Within the asynchronous ADMM, to alleviate the issues associated with synchronization, two conditions are used: 1) "partial barrier," which allows the coordinator to update multipliers after receiving solutions from one or few subsystems and 2) "bounded delay," which requires solutions from every subsystem to arrive at the coordinator at least once within a finite number of iterations [21], [32]. However, ADMM converges when solving convex problems only [21, p. 419], but when solving nonconvex problems, ADMM does not converge [33, p. 73]. This is because, within ADMM, stepsizes do not approach zero, which is the requirement to guarantee convergence when optimizing nonsmooth polyhedral concave dual functions [13], [23]. Moreover, quadratic penalties make the resulting relaxed problem nonlinear, which cannot be solved by MILP solvers. While penalty terms can be linearized [33], the minimum of penalties is typically not preserved, and the performance of the method is degraded. Furthermore, penalty terms are a part of each subproblem formulation, but these terms involve decision variables from multiple subproblems. Therefore, additional communication requirements are entailed. For example, in power systems, communication requirements among subsystems [21], [34] are needed.

C. Surrogate Lagrangian Relaxation Method

All major difficulties of standard LR, such as high computational effort, zigzagging of multipliers, and the requirement of the knowledge of the optimal dual value, have been overcome within our recent SLR [23], [24], [48]. Within the method, it is not necessary to optimally solve subproblems. Rather, it is sufficient to optimize subproblems subject to the simple "surrogate optimality condition" [23, p. 178, eq. 12] guaranteeing that "surrogate dual" values approach dual values [23, p. 181]. Convergence is proven without requiring the knowledge of the optimal dual value. This was achieved with a constructive process based on the contraction mapping concept whereby distances between Lagrange multipliers decrease at consecutive iterations, and as a result, multipliers converge to a unique limit. At the same time, stepsizes are kept sufficiently large to avoid premature algorithm termination. In addition, a constructive stepsizing formula satisfying these criteria has been developed. When solving large-scale problems, such as unit commitment problems arising in power systems [48], the method demonstrated high efficiency in the coordination of thousands of power generating units. SLR, thus, satisfies high computational efficiency requirement because of much-improved convergence over standard LR, and low communication requirements because subsystems are not required to communicate with each other. The method has been shown

to outperform other previous methods, including coordination methods, such as ADMM [24].

D. MILP Method: Branch-and-Cut

The main premise behind branch-and-cut [35] is that if the convex hull[1] of an MILP problem is obtained, the solution process is reduced to solving an LP problem. Owing to the linearity of the problem, the surface of the convex hull is polyhedral [40], and the vertices of the convex hull are feasible solutions to the original MILP problem. Because of finite numbers of variables and constraints, the number of vertices is finite, and linear programming methods, such as simplex methods converge to the optimal feasible solution within a finite number of iterations [36, p. 6]. However, the convex hull generally cannot be obtained. After relaxing integrality requirements, branch-and-cut solves the LP-relaxed problem. Attempting to obtain feasible solutions, branch-and-cut uses "cuts" to cutoff LP regions that contain fractional vertices without cutting off feasible solutions. While cuts generally require an infinite number of iterations to define facets of the convex hull, branch-and-cut resorts to branch-and-bound [37], [38] after a finite number of iterations when "tailing off" of cuts occurs [39, p. 349]. Since the number of fractional vertices that correspond to integer variables is finite, the number of branching operations required to obtain optimal feasible solutions is also finite.

III. Convergence of Distributed and Asynchronous Surrogate Lagrangian Relaxation

In Section III-A, an MILP problem formulation of a system consisting of several distributed subsystems is considered. In Section III-B, our recent price-based decomposition and coordination SLR method is presented. In Section III-C, a novel distributed and asynchronous SLR (DA-SLR) method is developed. In Section III-D, the convergence of DA-SLR is proven in the dual space. In Section III-E, the practical considerations of the new method are presented.

A. MILP System With Distributed Subsystems

Consider a system consisting of one coordinator and I distributed subsystems. Each subsystem is subject to its local linear constraints, which will not be considered for simplicity and ease of presentation. The entire system is subject to system-wide coupling constraints. These constraints couple individual subsystems and the MILP formulation of an overall system can be written as follows:

$$\min_{x} \sum_{i=1}^{I} f_i(x_i) \tag{1}$$

$$\text{subject to } \sum_{i=1}^{I} g_i(x_i) = 0 \tag{2}$$

where $x_i = (y_i, z_i) \in X_i \subset \mathbb{R}^{N_i^r} \times \mathbb{Z}^{N_i^z}$, X_i are closed and bounded sets, $x = (x_1, \ldots, x_I) = (y, z) \in X \subset \mathbb{R}^{N^r} \times \mathbb{Z}^{N^z}$, $y = (y_1, \ldots, y_I) \in \mathbb{R}^{N^r}$, $z = (z_1, \ldots, z_I) \in \mathbb{Z}^{N^z}$, with $\mathbb{R}$ denoting the set of real numbers, and $\mathbb{Z}$ denoting the set of integers. Functions f_i: $X_i \to \mathbb{R}$ and g_i: $X_i \to \mathbb{R}^m$ are linear. It is assumed that a set of feasible solutions that satisfy (1) and (2) is nonempty. To rule out possible irregularities, such as linear dependence of gradients, of active constraints in the continuous subspace $\mathbb{R}^{N^r}$, it is assumed that gradient vectors of active inequality constraints with respect to continuous variables y only are linearly independent at a local minimum $x^* = (y^*, z^*)$ of [41].

[1]The convex hull is the smallest convex set that encloses feasible solutions of a problem.

B. Surrogate Lagrangian Relaxation

As discussed in Section II, separability of the problem is exploited by relaxing coupling constraints (2) by introducing Lagrangian multipliers $\lambda^T = (\lambda_1, \ldots, \lambda_m) \in \mathbb{R}^m$ and by decomposing the resulting relaxed problem into individual subproblems

$$\min_{x_j} \left\{ f_j(x_j) + (\lambda)^T g_j(x_j) \right\}. \tag{3}$$

As discussed in Section II, it is not necessary to fully optimize subproblems within SLR. Rather, it is sufficient to stop optimization after the "surrogate" optimality condition for subproblems [23, eq. 57] is satisfied at iteration $k+1$

$$f_j\left(x_j^{k+1}\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^{k+1}\right) < f_j\left(x_j^k\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^k\right). \tag{4}$$

This condition is not the necessary requirement in the sense that if a subproblem is solved to optimality and the best solution found is the same as the most recent subproblem solution, i.e., $x_j^{k+1} = x_j^k$, then, although this solution does not satisfy (4), the algorithm can proceed. To coordinate subsystems, multipliers are updated in the following way:

$$\lambda^{k+1} = \lambda^k + s^k \tilde{g}\left(x^k\right), \quad k = 0, 1, \ldots \ . \tag{5}$$

Here,

$$\tilde{g}\left(x^k\right) = \sum_{i=1:i\neq j}^{I} g_i\left(x_i^{k-1}\right) + g_j\left(x_j^k\right) \tag{6}$$

are "surrogate" subgradient directions that are obtained instead of subgradient directions by using a solution from one subproblem. If inequality constraints are present in the formulation, multipliers are updated according to (5) with subsequent projection onto the positive orthant.

Multipliers (5) are updated using stepsizes s^k that satisfy [23, p. 180, eqs. (21a) and (21b)], which are derived based on the contraction mapping concept and are set as

$$s^k = \alpha_k \frac{s^{k-1} \left\| \tilde{g}\left(x^{k-1}\right) \right\|}{\left\| \tilde{g}\left(x^k\right) \right\|}, \quad 0 < \alpha_k < 1, \ k = 1, 2, \ldots \tag{7}$$

with

$$\alpha_k = 1 - \frac{1}{Mk^p}, \quad p = 1 - \frac{1}{k^r}, M > 1 \\ 0 < r < 1, k = 1, 2, 3, \ldots \ . \tag{8}$$

To ensure that stepsizes (7) are well defined, the following assumption is required.

Assumption 1 (Boundedness of Surrogate Subgradients): Surrogate subgradients satisfy the following condition:

$$\left\|\tilde{g}\left(x^k\right)\right\| < C < \infty. \tag{9}$$

This assumption is realistic for MILP problems defined over a closed and bounded set. Indeed, surrogate subgradients are levels of constraint violations. Since constraints are linear and each variable is defined over a closed and bounded set, constraint violations are finite.

Unlike the subgradient method, whereby zero subgradients imply that the optimum is obtained and the algorithm terminates with the optimal primal solution, within SLR, zero surrogate subgradient implies that there are no constraint violations and that a feasible solution is obtained, but it does not imply zero subgradients. Therefore, this solution is not guaranteed to be optimal. In this case, an iteration is skipped without updating multipliers (5) and stepsizes (7) and (8).

As proven in [23], multipliers (5) converge to their optimal values λ^* that maximize the following dual function:

$$q(\lambda) = \min_{x \in X} L(\lambda, x) \tag{10}$$

where

$$L(\lambda, x) = \sum_{i=1}^{I} f_i(x_i) + (\lambda)^T \sum_{i=1}^{I} g_i(x_i) \tag{11}$$

is the Lagrangian function. The problem of minimizing the Lagrangian function (11) within (10) is referred to as "the relaxed problem."

C. Distributed and Asynchronous Surrogate Lagrangian Relaxation

Within DA-SLR, it is assumed that subsystems have computational and communication capabilities. Namely, subsystems are capable of solving subproblems to obtain solutions that satisfy the surrogate optimality condition (4) and to send the resulting solution at the coordinator. To coordinate subsystems, it is also assumed that the coordinator has the capability to receive subproblem solutions, update multipliers, and send them to all subproblems. Throughout the rest of this article, superscript k will denote multiplier-updating iterations performed by the coordinator. Within the distributed and asynchronous framework, subproblems are assumed to perform their own "surrogate" optimization without waiting for other subproblems to finish, and the coordinator updates multipliers asynchronously without waiting for all subproblem solutions to arrive. For notational convenience, superscripts k of subproblems will denote the most recent subproblem solution available at iteration k.

Distributed Architecture of the Method: High-level architecture of the method is shown in Fig. 1.

As shown in Fig. 1, information exchanges are limited to multipliers λ that the coordinator broadcasts to all subsystems and subsystem solutions $x_1, x_2, \ldots, x_I$ that corresponding subsystems send at the coordinator. Each subproblem corresponds to a thread, and each thread is using branch-and-cut to solve the corresponding subproblem. The coordinator corresponds to a separate thread to update stepsizes, subgradient directions, and multipliers. The dynamic aspect of the coordination will be discussed next.

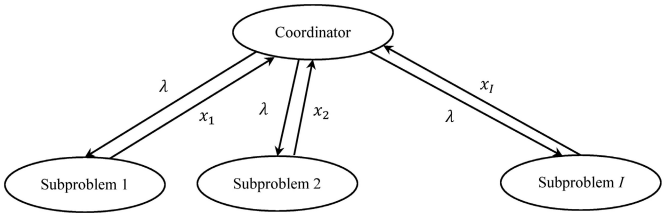

Fig. 1. Distributed architecture of the method.

By using a simple illustrative example, Fig. 2 demonstrates the asynchronous update by using one coordinator and three subproblems, and the difficulties caused by asynchronous updating of multipliers. After obtaining a solution to the first subproblem, the coordinator updates the multipliers without waiting for other solutions to arrive and broadcasts the updated multipliers to all subproblems. Subproblem 1 can then start solving the problem once receiving updated multipliers. Then, after the third subproblem is solved and its solution arrives at the coordinator, the coordinator once again updates multipliers and broadcasts their values to all subproblems, and so on. While asynchronous coordination avoids the synchronization issue, it leads to major convergence difficulties: 1) because of uncertainties in solving, communication and multiplier-updating times, the order in which subsystem solutions arrive at the coordinator is uncertain and 2) subsystem solutions are obtained based on multipliers of different vintages, and multipliers may not converge. For example, as demonstrated in Fig. 2, at coordinator iteration 4, x_1^3 is obtained using λ^2, x_2^4 is obtained using λ^0, and x_3^2 is obtained using λ^1. As a result, there may be convergence difficulties. In Section III-D, under realistic "freshness" assumption, the convergence of the DA-SLR method will be proved.

D. Convergence of Distributed and Asynchronous Surrogate Lagrangian Relaxation

It is assumed that within the DA-SLR method, subproblem solving times and communication times are finite, which is equivalent to the following "freshness" assumption.

Assumption 2 (Freshness): There exists integer number $D > 0$ such that within any consecutive D iterations, all subproblem solutions arrive at the coordinator at least once. □

Indeed, if solving and communication times are bounded, then each subproblem solution should arrive at the coordinator at least once within a finite number of iterations. The convergence of DA-SLR is stated in the following theorem.

Main Theorem: Suppose that Assumptions 1 and 2 hold, the surrogate optimality condition (4) is satisfied by subproblem solutions to (3) that are obtained by using branch-and-cut, Lagrange multipliers are updated per (5), and stepsizes are updated per (7) and (8). Within the DA-SLR method for coordinating MILP problems with separable structure as in (1) and (2), multipliers converge to λ^*. □

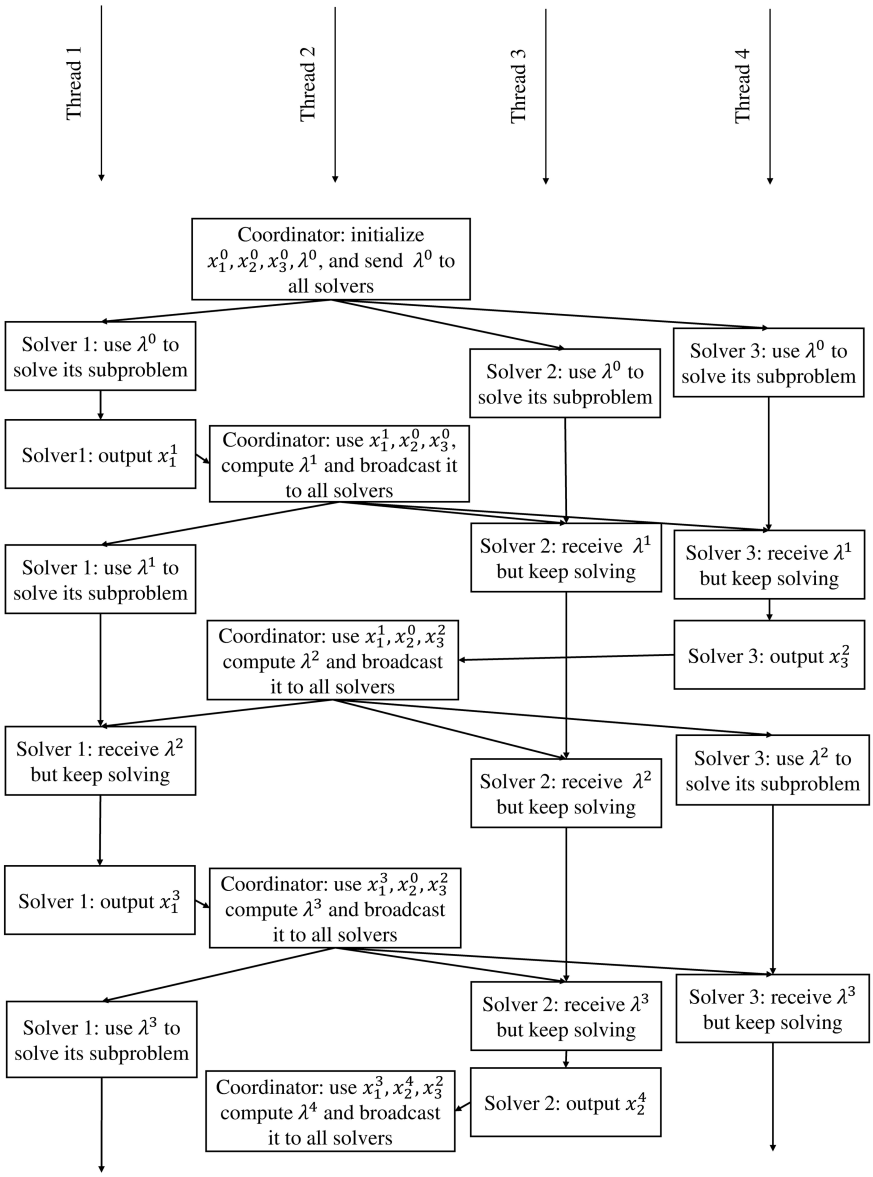

Fig. 2. Illustration of distributed and asynchronous SLR.

For the ease of understanding, the proof is split into three steps. In Step 1, it is proven that "surrogate" dual values approach dual values and multipliers approach the optimum "infinitely often" (Propositions 1–4). In Step 2, the Lyapunov function is introduced as the square of distances from multiplies to the optimum, and the upper bound on Lyapunov functions is developed (Propositions 5 and 6). In Step 3, with the help of the result obtained in Steps 1 and 2, in the Main Theorem, it is proven that the upper bound on Lyapunov functions approaches zero, thereby leading to convergence of multipliers.

Step 1 (Convergence of "Surrogate" Dual Values to Dual Values): Since subproblems are solved subject to the "surrogate" optimality condition (4), rather than obtaining dual values as within standard LR, "surrogate" dual values are obtained, which are generally above the dual surface. To prove that surrogate dual values approach dual values, Propositions 1 and 2 will first prove that the subproblem solutions satisfying (4) converge to their optimal values.

Proposition 1 (Convergence to Optimal Subproblem Solutions for Fixed λ): Assuming that subproblem solutions satisfy the surrogate optimality condition (4), then for each subproblem j, there exist finite K_j' such that the subproblem solution is optimal for multiplier values λ

$$x_j^{K_j'} = x_j^*(\lambda). \tag{12}$$

Proof: As explained in Section II-D, an optimal subproblem solution is obtained by branch-and-cut within a finite number of steps. A subproblem-feasible solution satisfying (4) is also obtained within a finite number of steps. Since multipliers are assumed to be constant here, (4) implies the decrease of subproblem objective function. Essentially, branch-and-cut will continue to search along with the unexplored nodes of

the branch tree, trying to find a lower objective function value until the subproblem-optimal solution is obtained. □

The limitation of Proposition 1 is that it is proven for a fixed set of multipliers. Within DA-SLR, multipliers are updated, and, therefore, the objective functions of subproblems (3) will change. In turn, this will affect the optimal solution of a subproblem. Proposition 2 removes this limitation.

Proposition 2 (Convergence to Optimal Subproblem Solutions): Assuming that subproblem solutions satisfy (4), then for each subproblem j, there exist finite K_j $(>K'_j)$ such that solution to subproblem j is optimal for multiplier values λ^{K_j}

$$x_j^{K_j} = x_j^*\left(\lambda^{K_j}\right). \tag{13}$$

Proof: As proven in Proposition 1, for K'_j and fixed $\lambda^{K'_j}$, a subproblem-optimal solution is $x_j^*(\lambda^{K'_j})$. What remains to prove is that when multipliers are updated, there exist K_j $(>K'_j)$ such that optimal solutions at $\lambda^{K'_j}$ and λ^{K_j} are the same

$$x_j^*\left(\lambda^{K'_j}\right) = x_j^*\left(\lambda^{K_j}\right). \tag{14}$$

To prove (12), introduce the following operator:

$$A\left(f_j(x_j) + (\lambda)^T g_j(x_j)\right) = \arg\min_{x_j}\left\{f_j(x_j) + (\lambda)^T g_j(x_j)\right\}. \tag{15}$$

Because subproblems are defined over bounded sets X_j, solutions are finite, and the following inequality holds:

$$\left\|A\left(f_j(x_j) + (\lambda)^T g_j(x_j)\right)\right\| < \infty. \tag{16}$$

The operator A is, thus, bounded [42]. Therefore, there exists a finite constant $C'_A > 0$ such that the following inequality holds:

$$\left\|A\left(f_j(x_j) + (\lambda)^T g_j(x_j)\right)\right\| < C'_A\left\|\left(f_j(x_j) + (\lambda)^T g_j(x_j)\right)\right\|. \tag{17}$$

To establish (12), consider the following norm:

$$\left\|x_j^*\left(\lambda^{K'_j}\right) - x_j^*\left(\lambda^{K_j}\right)\right\|. \tag{18}$$

Using (15), (18) can be rewritten as

$$\left\|A\left(f_j(x_j) + \left(\lambda^{K'_j}\right)^T g_j(x_j)\right) - A\left(f_j(x_j) + \left(\lambda^{K_j}\right)^T g_j(x_j)\right)\right\|. \tag{19}$$

Because X_j is bounded, subproblem objective functions (3) take on finite values, and therefore, the following inequality also holds:

$$\begin{aligned}&\left\|A\left(f_j(x_j) + \left(\lambda^{K'_j}\right)^T g_j(x_j)\right) - A\left(f_j(x_j) + \left(\lambda^{K_j}\right)^T g_j(x_j)\right)\right\| \\ &\le C_A\left\|\left(f_j(x_j) + \left(\lambda^{K'_j}\right)^T g_j(x_j)\right) - \left(f_j(x_j) + \left(\lambda^{K_j}\right)^T g_j(x_j)\right)\right\| \\ &= C_A\left\|\left(\left(\lambda^{K'_j}\right)^T - \left(\lambda^{K_j}\right)^T\right) g_j(x_j)\right\|.\end{aligned} \tag{20}$$

Here, C_A is a finite positive constant. Using the Cauchy–Schwarz inequality, (20) becomes

$$\begin{aligned}&\left\|A\left(f_j(x_j) + \left(\lambda^{K'_j}\right)^T g_j(x_j)\right) - A\left(f_j(x_j) + \left(\lambda^{K_j}\right)^T g_j(x_j)\right)\right\| \\ &\qquad \le C_A\left\|g_j(x_j)\right\|\left\|\lambda^{K'_j} - \lambda^{K_j}\right\|.\end{aligned} \tag{21}$$

Since $g_j(x_j)$ is a component of constraint violations, Assumption 1 is applicable, and therefore

$$\left\|x_j^*\left(\lambda^{K'_j}\right) - x_j^*\left(\lambda^{K_j}\right)\right\| \le C_A C\left\|\lambda^{K'_j} - \lambda^{K_j}\right\|. \tag{22}$$

Since stepsizes (7) and (8) approach zero [23], there exist iteration K'_j and K_j such that for any $\varepsilon > 0$, the following inequality holds:

$$s^{K'_j} < \frac{\varepsilon}{C_A C^2\left(K_j - K'_j\right)}. \tag{23}$$

Therefore

$$\left\|\lambda^{K'_j} - \lambda^{K_j}\right\| \le \sum_{i=K'_j}^{K_j-1}\left\|\tilde{g}(x^i)\right\| s^i < \frac{C\left(K_j - K'_j\right)\varepsilon}{C_A C^2\left(K_j - K'_j\right)} = \frac{\varepsilon}{C_A C}. \tag{24}$$

From (22) and (24), it is followed that:

$$\left\|x_j^*\left(\lambda^{K'_j}\right) - x_j^*\left(\lambda^{K_j}\right)\right\| < \varepsilon. \tag{25}$$

As reviewed in Section II, subproblem convex hulls contain a finite number of vertices, each corresponding to a feasible solution. Moreover, it can be assumed that distances between any two adjacent vertices are greater than ε. Therefore, optimal solutions at iterations K'_j and K_j are the same, and (14) holds. Since it takes a finite number of iterations to obtain $x_j^*(\lambda^{K'_j})$ without updating multipliers, it will also take a finite number of iterations to obtain $x_j^*(\lambda^{K_j})$ when multipliers are updated. □

Proposition 3 (Convergence of "Surrogate" Dual Values to Dual Values): With stepsizing formulas (7) and (8), Lagrange multipliers (5) converge to a unique fixed point

$$\lambda^k \to \bar{\lambda} \tag{26}$$

(not necessarily λ^*), and surrogate dual values approach dual values

$$\tilde{L}\left(\bar{\lambda}, x^k\right) \to q\left(\bar{\lambda}\right) \tag{27}$$

where

$$q\left(\bar{\lambda}\right) = \min_{x\in X} L\left(\bar{\lambda}, x\right) \tag{28}$$

is a dual value obtained by solving all subproblems optimally and

$$\tilde{L}\left(\bar{\lambda}, x^k\right) \equiv \sum_{i=1}^{I} f_i\left(x_i^k\right) + \left(\bar{\lambda}\right)^T \tilde{g}\left(x_i^k\right) \tag{29}$$

is a "surrogate" dual value obtained after solving one or a few subproblems subject to the surrogate optimality condition (4).

Proof: As proven in [23], stepsizes (7) and (8) approach zero. To prove that surrogate dual values approach dual values, consider first the surrogate optimality condition for one subproblem j

$$f_j\left(x_j^{k+1}\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^{k+1}\right) < f_j\left(x_j^k\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^k\right). \tag{30}$$

By using (5), inequality (30) can be rewritten as

$$f_j\left(x_j^{k+1}\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^{k+1}\right) < f_j\left(x_j^k\right) + \left(\lambda^k\right)^T g_j\left(x_j^k\right) + s^k \left\| g_j\left(x_j^k\right)\right\|^2. \quad (31)$$

The inequality (31) can then be equivalently rewritten as

$$f_j\left(x_j^{k+1}\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^{k+1}\right) - f_j\left(x_j^k\right) - \left(\lambda^k\right)^T g_j\left(x_j^k\right) < s^k \left\| g_j\left(x_j^k\right)\right\|^2. \quad (32)$$

As stepsizes approach zero, there exists κ so that for all $k > \kappa$ and all $\varepsilon > 0$, the following inequality holds:

$$f_j\left(x_j^{k+1}\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^{k+1}\right) - f_j\left(x_j^k\right) - \left(\lambda^k\right)^T g_j\left(x_j^k\right) < \varepsilon \left\| g_j\left(x_j^k\right)\right\|^2 < \varepsilon C^2. \quad (33)$$

Therefore, $\{f_j(x_j^k) + (\lambda^k)^T g_j(x_j^k)\}$ forms a convergent sequence

$$f_j\left(x_j^k\right) + \left(\lambda^k\right)^T g_j\left(x_j^k\right) \to f_j\left(\bar{x}_j\right) + \left(\bar{\lambda}\right)^T g_j\left(\bar{x}_j\right) < \infty. \quad (34)$$

Indeed, as proven in Propositions 1 and 2, subproblem solutions approach a limit, which is here denoted as $\bar{x}_j$. Moreover, the situation whereby

$$\left(\lambda^k\right)^T g_j\left(x_j^k\right) \to \infty \quad (35)$$

is impossible. Multipliers cannot grow without bound because it would imply that there is always positive or always negative constraint violation,[2] implying infeasibility of (1) and (2), which is impossible. From Assumption 2, within any consecutive D iterations, all subproblem solutions arrive at the coordinator at least once. Moreover, by Propositions 1 and 2, subproblem-feasible solutions are obtained within a finite number of iterations. Therefore, optimal solutions to all subproblems are obtained within a finite number of iterations, implying that "surrogate" dual values approaches dual values

$$\tilde{L}\left(\bar{\lambda}, x^k\right) \to q\left(\bar{\lambda}\right) \quad \text{as } s^k \to 0. \quad (36)$$

□

Proposition 4 (Rate of Convergence [23, p. 187]): When stepsizes are updated per (7) and (8), there exists $\nu > 0$, and the following condition is satisfied "infinitely often":

$$q^* - \tilde{L}\left(\lambda^k, x^k\right) \geq \nu \left\| \lambda^* - \lambda^k \right\|^2, \quad k \in \aleph. \quad (37)$$

Here, $\aleph$ is an infinite subset of natural numbers.

Proof: If condition (37) is not satisfied infinitely often for $\nu > 0$, then starting from iteration κ, the following inequality holds:

$$q^* - \tilde{L}\left(\lambda^k, x^k\right) < \nu \left\| \lambda^* - \lambda^k \right\|^2, \quad k > \kappa. \quad (38)$$

There are three cases.

Case 1: The left-hand side of (38) is negative. Surrogate dual values are greater than q^* for all $k > \kappa$, which contradicts Proposition 3 that states that surrogate dual values approach dual values.

Case 2: The left-hand side of (38) is positive, and $q^* - \tilde{L}(\lambda^k, x^k) > \varepsilon$ for some $\varepsilon > 0$ and $k' > k > \kappa$, then there exists $\nu' = (\varepsilon/(\|\lambda^* - \lambda^{k'}\|^2)) > 0$, and the following condition holds:

$$q^* - \tilde{L}\left(\lambda^{k'}, x^{k'}\right) \geq \nu' \left\| \lambda^* - \lambda^{k'} \right\|^2. \quad (39)$$

There is a contradiction with (38) because, in this case, it is possible to find $\nu' > 0$ that satisfies (37).

Case 3: The left-hand side of (38) is positive, but infinitesimally small, $q^* - \tilde{L}(\lambda^k, x^k) < \varepsilon$ for all $\varepsilon > 0$, then surrogate dual values approach q^*. Since, by Proposition 3, surrogate dual values approach dual values, then dual values approach the optimal dual value, and convergence to the optimum is immediate. □

Step 2 (Development of an Upper Bound for Lyapunov Functions): In this step, the Lyapunov function is defined as

$$V\left(\lambda^k\right) = \left\| \lambda^* - \lambda^k \right\|^2 \quad (40)$$

which is the square of the distance from current to optimal multipliers. Because subproblem solving times and subproblem-coordinator communication times are uncertain, different sequences of subproblem solutions arriving at the coordinator lead to different trajectories of multipliers. As a result, the exact representation of the Lyapunov function is unknown. To resolve this issue, an upper bound of the Lyapunov function is derived in Propositions 5 and 6 as an envelope of all possible Lyapunov functions for any sequence of subproblems arriving at the coordinator. Two inequalities are derived based on whether the condition (37) holds or not in Proposition 5. In Proposition 6, these inequalities are combined to derive an upper bound on all possible Lyapunov functions.

Proposition 5: As proven in [23, p. 187], under condition (37) and assuming that stepsizes are "sufficiently small" $s^k \leq 1/(2\nu)$, the following inequality holds:

$$\left\| \lambda^* - \lambda^{k+1} \right\|^2 \leq \left\| \lambda^* - \lambda^k \right\|^2 \cdot \left(1 - 2s^k\nu\right) + \left(s^k\right)^2 \left\| \tilde{g}\left(x^k\right)\right\|^2. \quad (41)$$

If condition (37) is not satisfied or stepsizes are not "sufficiently small" $s^k > 1/(2\nu)$, then the following inequality holds:

$$\left\| \lambda^* - \lambda^{k+1} \right\|^2 \leq \left\| \lambda^* - \lambda^k \right\|^2 \cdot \left(1 + s^k\beta^k \left\| \tilde{g}\left(x^k\right)\right\|\right) + \left(s^k\right)^2 \left\| \tilde{g}\left(x^k\right)\right\| \cdot \left(\frac{1}{\beta^k} + \left\| \tilde{g}\left(x^k\right)\right\|\right), \quad \beta^k > 0. \quad (42)$$

Proof: Inequality (41) has been derived in [23, Proposition 2.5]. To derive inequality (42) consider

$$\left\| \lambda^* - \lambda^{k+1} \right\|^2 \leq \left\| \lambda^* - \lambda^k \right\|^2 - 2s^k\left(\lambda^* - \lambda^k\right)\tilde{g}\left(x^k\right) + \left(s^k\right)^2 \left\| \tilde{g}\left(x^k\right)\right\|^2. \quad (43)$$

By using the Cauchy–Schwarz inequality, (43) becomes

$$\left\| \lambda^* - \lambda^{k+1} \right\|^2 \leq \left\| \lambda^* - \lambda^k \right\|^2 + 2s^k \left\| \lambda^* - \lambda^k \right\| \cdot \left\| \tilde{g}\left(x^k\right)\right\| + \left(s^k\right)^2 \left\| \tilde{g}\left(x^k\right)\right\|^2. \quad (44)$$

The right-hand side of (44) contains the Lyapunov function $\|\lambda^* - \lambda^k\|^2$ at iteration k and its square root $\|\lambda^* - \lambda^k\|$.

[2]If there are inequality constraints, then the violation would be positive.

In order to express the inequality (44) in terms of the Lyapunov function, the basic inequality $2ab \leq \beta a^2 + (1/\beta)b^2$ [15] is used, and the inequality (44) becomes

$$\left\|\lambda^* - \lambda^{k+1}\right\|^2 \leq \left\|\lambda^* - \lambda^k\right\|^2 + s^k\beta^k\left\|\lambda^* - \lambda^k\right\|^2 \cdot \left\|\tilde{g}\left(x^k\right)\right\| + \frac{1}{\beta^k}\left(s^k\right)^2\left\|\tilde{g}\left(x^k\right)\right\| + \left(s^k\right)^2\left\|\tilde{g}\left(x^k\right)\right\|^2. \quad (45)$$

Therefore

$$\left\|\lambda^* - \lambda^{k+1}\right\|^2 \leq \left\|\lambda^* - \lambda^k\right\|^2 \cdot \left(1 + s^k\beta^k\left\|\tilde{g}\left(x^k\right)\right\|\right) + \left(s^k\right)^2\left\|\tilde{g}\left(x^k\right)\right\| \cdot \left(\frac{1}{\beta^k} + \left\|\tilde{g}\left(x^k\right)\right\|\right). \quad (46)$$

□

In Proposition 6, an upper bound on Lyapunov functions at iteration $k+1$ in terms of Lyapunov functions at iteration 0 is derived by induction taking into account all possible realizations of Lyapunov functions.

Proposition 6 (Upper Bound for Lyapunov Functions): The following upper bound is valid for Lyapunov functions:

$$V\left(\lambda^{k+1}\right) \leq \left\|\lambda^* - \lambda^0\right\|^2 \prod_{i=0}^{k} P^i + \sum_{j=0}^{k-1}\left(\left(s^j\right)^2\left\|\tilde{g}\left(x^j\right)\right\|\left(\frac{1}{\beta^j} + \left\|\tilde{g}\left(x^j\right)\right\|\right)\prod_{l=j+1}^{k} P^l\right) + \left(s^k\right)^2\left\|\tilde{g}\left(x^k\right)\right\|\left(\frac{1}{\beta^k} + \left\|\tilde{g}\left(x^k\right)\right\|\right), \quad k \geq 1 \quad (47)$$

where $P^i = (1 - 2s^i\nu)$ if condition (37) holds at iteration i, and $P^i = (1 + s^i\beta^i\|\tilde{g}(x^i)\|)$ otherwise.

Proof: Proof will follow by induction by first proving that the equation is true when $k = 1$, then assuming it is true for k, showing it is true for $k+1$.

Before starting the induction, consider the situation whereby $k = 0$. If condition (37) is satisfied, then inequality (41) holds for $k = 0$

$$\left\|\lambda^* - \lambda^1\right\|^2 \leq \left\|\lambda^* - \lambda^0\right\|^2\left(1 - 2s^0\nu\right) + \left(s^0\right)^2\left\|\tilde{g}\left(x^0\right)\right\|^2. \quad (48)$$

If (37) is not satisfied, then inequality (42) holds for $k = 0$

$$\left\|\lambda^* - \lambda^1\right\|^2 \leq \left\|\lambda^* - \lambda^0\right\|^2\left(1 + s^0\beta^0\left\|\tilde{g}\left(x^0\right)\right\|\right) + \left(s^0\right)^2\left\|\tilde{g}\left(x^0\right)\right\|\left(\frac{1}{\beta^0} + \left\|\tilde{g}\left(x^0\right)\right\|\right), \quad \beta^0 > 0. \quad (49)$$

Since the term $(s^0)^2\|\tilde{g}(x^0)\|((1/\beta^0) + \|\tilde{g}(x^0)\|)$ which appears in (49) is greater than $(s^0)^2\|\tilde{g}(x^0)\|^2$ which appears in (48), the following expression is the upper bound for $\|\lambda^* - \lambda^1\|^2$:

$$\left\|\lambda^* - \lambda^1\right\|^2 \leq \left\|\lambda^* - \lambda^0\right\|^2 P^0 + \left(s^0\right)^2\left\|\tilde{g}\left(x^0\right)\right\|\left(\frac{1}{\beta^0} + \left\|\tilde{g}\left(x^0\right)\right\|\right), \quad \beta^0 > 0 \quad (50)$$

where $P^0 = (1 - 2s^0\nu)$ if condition (37) holds at $k = 0$, and $P^0 = (1 + s^0\beta^0\|\tilde{g}(x^0)\|)$ if condition (37) does not holds at $k = 0$. The inequality (50) is indeed an upper bound of $\|\lambda^* - \lambda^1\|^2$ because if condition (37) does not hold, then (50) reduces to (49), and if condition (37) holds, then (50) reduces to (48) plus a positive extra term $(((s^0)^2\|\tilde{g}(x^0)\|)/\beta^0)$.

Following the same logic, the following holds for $k = 1$:

$$\begin{aligned}\left\|\lambda^* - \lambda^2\right\|^2 &\leq \left\|\lambda^* - \lambda^1\right\|^2 P^1 + \left(s^1\right)^2\left\|\tilde{g}\left(x^1\right)\right\|\left(\frac{1}{\beta^1} + \left\|\tilde{g}\left(x^1\right)\right\|\right)\\ &= \left\|\lambda^* - \lambda^0\right\|^2 P^0 P^1 + \left(s^0\right)^2\left\|\tilde{g}\left(x^0\right)\right\|\left(\frac{1}{\beta^0} + \left\|\tilde{g}\left(x^0\right)\right\|\right)P^1\\ &\quad + \left(s^1\right)^2\left\|\tilde{g}\left(x^1\right)\right\|\left(\frac{1}{\beta^1} + \left\|\tilde{g}\left(x^1\right)\right\|\right)\end{aligned} \quad (51)$$

where $P^1 = (1 - 2s^1\nu)$ if condition (37) holds at $k = 1$, and $P^0 = (1 + s^1\beta^1\|\tilde{g}(x^1)\|)$ if condition (37) does not holds at $k = 1$. Inequality (51) is indeed the same as inequality (47) for $k = 1$.

What remains to prove is that assuming that (47) holds at iteration k, it also holds for $k+1$

$$\begin{aligned}\left\|\lambda^* - \lambda^{k+2}\right\|^2 &\leq \left\|\lambda^* - \lambda^0\right\|^2\prod_{i=0}^{k+1} P^i + \sum_{j=0}^{k}\left(\left(s^j\right)^2\left\|\tilde{g}\left(x^j\right)\right\|\left(\frac{1}{\beta^j} + \left\|\tilde{g}\left(x^j\right)\right\|\right)\prod_{l=j+1}^{k+1} P^l\right)\\ &\quad + \left(s^{k+1}\right)^2\left\|\tilde{g}\left(x^{k+1}\right)\right\|\left(\frac{1}{\beta^{k+1}} + \left\|\tilde{g}\left(x^{k+1}\right)\right\|\right).\end{aligned} \quad (52)$$

The validity of (52) is derived using the same logic as that used in deriving (50). Consider the following inequality:

$$\left\|\lambda^* - \lambda^{k+2}\right\|^2 \leq \left\|\lambda^* - \lambda^{k+1}\right\|^2 P^{k+1} + \left(s^{k+1}\right)^2\left\|\tilde{g}\left(x^{k+1}\right)\right\|\left(\frac{1}{\beta^{k+1}} + \left\|\tilde{g}\left(x^{k+1}\right)\right\|\right). \quad (53)$$

After substituting the expression for $\|\lambda^* - \lambda^{k+1}\|^2$ from (47) into (53), one obtains the following inequality:

$$\begin{aligned}\left\|\lambda^* - \lambda^{k+2}\right\|^2 &\leq \left(\begin{array}{l}\left\|\lambda^* - \lambda^0\right\|^2\prod_{i=0}^{k} P^i\\ + \sum_{j=0}^{k-1}\left(\left(s^j\right)^2\left\|\tilde{g}\left(x^j\right)\right\|\left(\frac{1}{\beta^j} + \left\|\tilde{g}\left(x^j\right)\right\|\right)\prod_{l=j+1}^{k} P^l\right)\\ + \left(s^k\right)^2\left\|\tilde{g}\left(x^k\right)\right\|\left(\frac{1}{\beta^k} + \left\|\tilde{g}\left(x^k\right)\right\|\right)\end{array}\right)P^{k+1}\\ &\quad + \left(s^{k+1}\right)^2\left\|\tilde{g}\left(x^{k+1}\right)\right\|\left(\frac{1}{\beta^{k+1}} + \left\|\tilde{g}\left(x^{k+1}\right)\right\|\right).\end{aligned} \quad (54)$$

The inequality (54) simplifies to the following:

$$\begin{aligned}\left\|\lambda^* - \lambda^{k+2}\right\|^2 &\leq \left\|\lambda^* - \lambda^0\right\|^2\prod_{i=0}^{k+1} P^i + \left(\sum_{j=0}^{k-1}\left(\left(s^j\right)^2\left\|\tilde{g}\left(x^j\right)\right\|\left(\frac{1}{\beta^j} + \left\|\tilde{g}\left(x^j\right)\right\|\right)\prod_{l=j+1}^{k} P^l\right)\right)P^{k+1}\\ &\quad + \left(s^k\right)^2\left\|\tilde{g}\left(x^k\right)\right\|\left(\frac{1}{\beta^{k-1}} + \left\|\tilde{g}\left(x^k\right)\right\|\right)P^{k+1}\\ &\quad + \left(s^{k+1}\right)^2\left\|\tilde{g}\left(x^{k+1}\right)\right\|\left(\frac{1}{\beta^{k+1}} + \left\|\tilde{g}\left(x^{k+1}\right)\right\|\right).\end{aligned} \quad (55)$$

After further simplifications, the inequality (55) becomes

$$\begin{aligned}\left\|\lambda^* - \lambda^{k+2}\right\|^2 &\le \left\|\lambda^* - \lambda^0\right\|^2 \prod_{i=0}^{k+1} P^i \\ &+ \sum_{j=0}^{k}\left(\left(s^j\right)^2 \left\|\tilde{g}\left(x^j\right)\right\|\left(\frac{1}{\beta^j} + \left\|\tilde{g}\left(x^j\right)\right\|\right)\prod_{l=j+1}^{k+1} P^l\right) \\ &+ \left(s^{k+1}\right)^2 \left\|\tilde{g}\left(x^{k+1}\right)\right\|\left(\frac{1}{\beta^{k+1}} + \left\|\tilde{g}\left(x^{k+1}\right)\right\|\right). \end{aligned} \tag{56}$$

The inequality (56) is the sought-for inequality (47).

Step 3 (Convergence of the Upper Bound to Zero): In this step, the Main Theorem is proven, mainly, it is proven that the upper bound on the Lyapunov function defined in (47) asymptotically approaches zero, thereby leading to the convergence of multipliers to λ^*.

Proof of the Main Theorem: In order to prove that $\lambda^k \to \lambda^*$, it is necessary to prove that the upper bound on the Lyapunov function [right-hand side of (47)] approaches zero. This leads the Lyapunov function to converge to zero and to the convergence of multipliers.

Since λ^* that maximizes the dual function (10) is assumed to exist, the term $\|\lambda^* - \lambda^0\|^2$ is finite. Therefore, it is sufficient to prove that the following expression approaches zero:

$$\prod_{i=0}^{k-1} P^i = \prod_{i=0:i\in\mathbb{N}/\aleph}^{k-1}\left(1 + s^i\beta^i\left\|\tilde{g}\left(x^k\right)\right\|\right)\prod_{i=0:i\in\aleph}^{k-1}\left(1 - 2s^i\nu\right) \tag{57}$$

where $\aleph$ is the set is iteration numbers whereby inequality (41) holds and $\mathbb{N}$ is the set of natural numbers.

To prove that (57) approaches zero, the stepsizing formulas (7) and (8) are plugged in first, then the resulting function is upper bounded by using standard functions and their asymptotical representation, and then, through algebraic manipulations, the condition for β^i is derived to ensure that (57) approaches zero. By exploiting the fact that set $\aleph$ is a proper subset of natural numbers $\aleph \subset \mathbb{N}$ and that each term $(1 + s^i\beta^i\|\tilde{g}(x^i)\|)$ is greater than 1, the following inequality holds:

$$\begin{aligned}&\prod_{i=0:i\in\mathbb{N}/\aleph}^{k-1}\left(1 + s^i\beta^i\left\|\tilde{g}\left(x^i\right)\right\|\right)\prod_{i=0:i\in\aleph}^{k-1}\left(1 - 2s^i\nu\right) \\ &\le \prod_{i=0:i\in\mathbb{N}}^{k-1}\left(1 + 2\left(\prod_{j=1}^{i}\alpha_j\right)s^0\left\|\tilde{g}\left(x^0\right)\right\|\beta^i\right)\prod_{i=0:i\in\aleph}^{k-1}\left(1 - 2s^i\nu\right).\end{aligned} \tag{58}$$

Assuming that condition (37) is satisfied at least every N ($<\infty$) iterations, the entire expression (58) is upper bounded as

$$\begin{aligned}&\prod_{i=0:i\in\mathbb{N}/\aleph}^{k-1}\left(1 + s^i\beta^i\left\|\tilde{g}\left(x^i\right)\right\|\right)\prod_{i=0:i\in\aleph}^{k-1}\left(1 - 2s^i\nu\right) \\ &\le \prod_{i=0:i\in\mathbb{N}}^{k-1}\left(1 + 2\left(\prod_{j=1}^{i}\alpha_j\right)s^0\left\|\tilde{g}\left(x^0\right)\right\|\beta^i\right) \\ &\times \prod_{i=0}^{\lfloor(k-1)/N\rfloor}\left(1 - 2\left(\prod_{j=1}^{iN}\alpha_j\right)\frac{s^0\left\|\tilde{g}\left(x^0\right)\right\|\nu}{\left\|\tilde{g}\left(x^i\right)\right\|}\right).\end{aligned} \tag{59}$$

If such N does not exist and condition (37) does not hold infinitely often, then there is a contradiction with Proposition 3. To prove that the right-hand side of (59) approaches zero, consider α_k from (8) which asymptotically behaves as $1 - (1/Mk)$ as $k \to \infty$ [23], therefore, asymptotically, the right-hand side of (59) becomes

$$\begin{aligned}&\prod_{i=0:i\in N}^{\infty}\left(1 + 2\prod_{j=1}^{i}\left(1 - \frac{1}{Mj}\right)s^0\left\|\tilde{g}\left(x^0\right)\right\|\beta^i\right) \\ &\times \prod_{i=0}^{\infty}\left(1 - 2\prod_{j=1}^{iN}\left(1 - \frac{1}{Mj}\right)\frac{s^0\left\|\tilde{g}\left(x^0\right)\right\|\nu}{\left\|\tilde{g}\left(x^i\right)\right\|}\right).\end{aligned} \tag{60}$$

The product $\prod_{j=1}^{i}(1 - (1/Mj))$ can be expressed in terms of a "Pochhammer function" [43], which asymptotically behaves as $(\gamma(((M-1)/M))i^{(1/M)})^{-1}$ [43], [44] where γ is the Euler's gamma function. Therefore, asymptotically, (60) approaches the following expression:

$$\prod_{i=0}^{\infty}\left(1 + \frac{2s^0\left\|\tilde{g}\left(x^0\right)\right\|\beta^i}{\gamma\left(\frac{M-1}{M}\right)i^{\frac{1}{M}}}\right)\prod_{i=0}^{\infty}\left(1 - \frac{2s^0\left\|\tilde{g}\left(x^0\right)\right\|\nu}{\left\|\tilde{g}\left(x^{iN}\right)\right\|\gamma\left(\frac{M-1}{M}\right)(iN)^{\frac{1}{M}}}\right). \tag{61}$$

After regrouping terms, (61) becomes

$$\prod_{j=1}^{\infty}\left(\prod_{i=1-N+jN}^{jN}\left(1 + \frac{2s^0\left\|\tilde{g}\left(x^0\right)\right\|\beta^i}{\gamma\left(\frac{M-1}{M}\right)i^{\frac{1}{M}}}\right)\times\left(1 - \frac{2s^0\left\|\tilde{g}\left(x^0\right)\right\|\nu}{\left\|\tilde{g}\left(x^{jN}\right)\right\|\gamma\left(\frac{M-1}{M}\right)(jN)^{\frac{1}{M}}}\right)\right). \tag{62}$$

After expanding the inner product, and ignoring involving $(j)^{-2/M}$ and higher order terms, (62) becomes

$$\prod_{j=1}^{\infty}\left(1 + \sum_{i=1-N+jN}^{jN}\frac{2s^0\left\|\tilde{g}\left(x^0\right)\right\|\beta^i}{\gamma\left(\frac{M-1}{M}\right)i^{\frac{1}{M}}} - \frac{2s^0\left\|\tilde{g}\left(x^0\right)\right\|\nu}{\left\|\tilde{g}\left(x^{jN}\right)\right\|\gamma\left(\frac{M-1}{M}\right)(jN)^{\frac{1}{M}}}\right). \tag{63}$$

To ensure that products involve terms less than 1 each, consider

$$\prod_{j=1}^{\infty}\left(1 + \frac{2s^0\left\|\tilde{g}\left(x^0\right)\right\|}{\gamma\left(\frac{M-1}{M}\right)} \times\left(\frac{N\max_{i=1-N+jN,\ldots,jN}\beta^i}{(1-N+jN)^{\frac{1}{M}}} - \frac{\nu}{\left\|\tilde{g}\left(x^{jN}\right)\right\|(jN)^{\frac{1}{M}}}\right)\right). \tag{64}$$

To ensure that every term is less than 1, consider

$$\beta^i < \frac{(1-N+jN)^{\frac{1}{M}}\nu}{N\left\|\tilde{g}\left(x^{jN}\right)\right\|(jN)^{\frac{1}{M}}} \qquad i = 1-N+jN,\ldots,\quad jN, j = 1,2,\ldots. \tag{65}$$

The second term of the right-hand side of (47) also approaches zero, because it involves similar products as in (57), and the proof follows exactly the same logic. The last term in the right-hand side of (47) approaches zero because stepsizes approach zero. □

E. Practical Implementations of the Method

In practical implementation, the following considerations are important. The coordinator needs to initialize stepsizes and multipliers at the beginning of the algorithm. In addition, the coordinator needs to obtain feasible costs and to provide the quality of the feasible solutions during or after the iterative process. These considerations are discussed next.

Initialization: The coordinator initializes multipliers and stepsizes. Multiplier initialization is typically problem-dependent; specific initialization for GAPs tested in this article will be explained in Section IV. One possible way to initialize stepsizes is [23, eq. 76, p. 190]

$$s^0 = \frac{\hat{q} - \tilde{L}(\lambda^0, x^0)}{\left\| \tilde{g}(x^0) \right\|^2} \tag{66}$$

where $\hat{q}$ is an estimate of the optimal dual value. Specific ways to obtain this estimate are discussed in Section IV.

Criteria to Search for Feasible Solutions: To find the feasible cost to the original problem, feasible solutions need to be searched for within the method. Several criteria can be used to start searching for feasible solutions: 1) search with a predetermined frequency, e.g., once every 100 iterations or 2) search after surrogate subgradient norm is below a certain threshold. There could be other criteria, such as the CPU time limit. Once some of these criteria are satisfied, feasible solutions are searched, as explained next.

Obtaining of Feasible Solutions: The process of obtaining feasible solutions is generally problem-dependent. Solutions to subproblems are feasible with respect to subproblems, but these solutions, when put together, may not satisfy relaxed constraints. To satisfy these constraints, some subproblem solutions are adjusted. This can be performed by selecting a few subproblems and fixing decision variables associated with other subproblems within the original problem (1) and (2) at x^k, the most recent values obtained by solving subproblems, and the resulting problem is solved by B&C. If a solution feasible with respect to the original problem is obtained, then the feasible costis computed; otherwise, multipliers are adjusted for a few more iterations, and a feasible solution is searched again.

Calculation of the Lower Bound: It is assumed that subsystems can solve subproblems (3) optimally. Dual values provide valid lower bounds, which are obtained by minimizing the Lagrangian function (10), which is equivalent to solving all subproblems optimally without updating the multipliers.

Stopping Criteria: The algorithm is terminated after the duality gap, which is the relative difference between the feasible cost and the lower bound value and is below a predetermined threshold.

Flowchart of the Algorithm: The algorithm is summarized in the flowchart, as shown in Fig. 3.

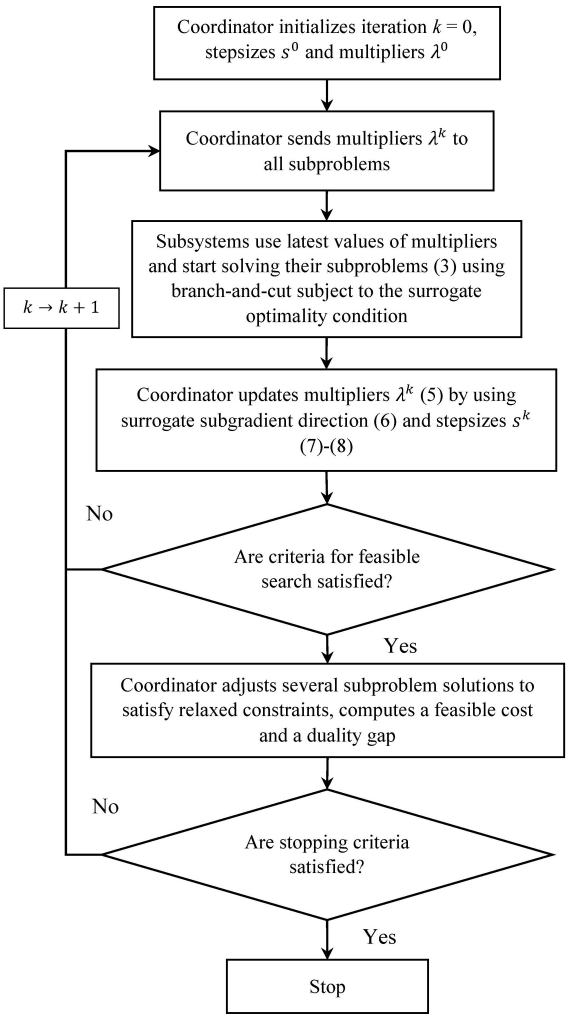

Fig. 3. Flowchart of the DA-SLR method.

IV. Numerical Testing

The purpose of this section is to demonstrate the performance of the new method. In Example 1, a small integer linear problem is considered to demonstrate that the Lyapunov function approaches zero fast. In Example 2, a GAP with 20 machines and 1600 jobs is considered to demonstrate the capability of DA-SLR to solve large-scale optimization problems fast with near-optimal performance. Because of difficulties associated with other methods as reviewed in Section II-B of Literature Review, a comparison of DA-SLR is performed against its sequential version—SLR [23], which, in turn, has been shown to outperform other previous coordination methods in [24], such as ADMM. Moreover, another variation of SLR—distributed and synchronous SLR (DS-DLR)—is used for further comparison. The new method is implemented using IBM ILOG CPLEX Optimization Studio Version: 12.7.1.0 on a PC with 3.10-GHz Intel Xeon CPU and 32-GB RAM.

Example 1 (Small Integer Programming Problem): Consider the following integer optimization problem

$$\min_{\{x_1,x_2,x_3,x_4,x_5,x_6\}\in\mathbb{Z}} \{x_1 + 2x_2 + 3x_3 + x_4 + 2x_5 + 3x_6\} \tag{67}$$

$$\text{s.t. } x_1 + 3x_2 + 5x_3 + x_4 + 3x_5 + 5x_6 - 26 \geq 0$$
$$2x_1 + 1.5x_2 + 5x_3 + 2x_4 + 0.5x_5 + x_6 - 16 \geq 0$$
$$0 \leq x_i \leq 3, \quad i = 1, \ldots, 6. \tag{68}$$

After constraints (68) are relaxed by using multipliers μ_1 and μ_2, the Lagrangian function becomes

$$\begin{aligned} &L(x_1, x_2, x_3, x_4, x_5, x_6, \mu_1, \mu_2) \\ &\quad = x_1 + 2x_2 + 3x_3 + x_4 + 2x_5 + 3x_6 \\ &\qquad + \mu_1(-x_1 - 3x_2 - 5x_3 - x_4 - 3x_5 - 5x_6 + 26) \\ &\qquad + \mu_2(-2x_1 - 1.5x_2 - 5x_3 - 2x_4 - 0.5x_5 - x_6 + 16). \end{aligned} \quad (69)$$

The relaxed problem is then decomposed into six individual subproblems, one for each variable

$$\begin{aligned} &\min_{x_1 \in \mathbb{Z}} \{x_1 - \mu_1 x_1 - 2\mu_2 x_1\} \\ &\text{s.t. } 0 \le x_1 \le 3 \\ &\min_{x_2 \in \mathbb{Z}} \{2x_2 - 3\mu_1 x_2 - 1.5\mu_2 x_2\} \\ &\text{s.t. } 0 \le x_2 \le 3 \\ &\min_{x_3 \in \mathbb{Z}} \{3x_3 - 5\mu_1 x_3 - 5\mu_2 x_3\} \\ &\text{s.t. } 0 \le x_3 \le 3 \\ &\min_{x_4 \in \mathbb{Z}} \{x_4 - \mu_1 x_4 - 2\mu_2 x_4\} \\ &\text{s.t. } 0 \le x_4 \le 3 \\ &\min_{x_5 \in \mathbb{Z}} \{2x_5 - 3\mu_1 x_5 - 0.5\mu_2 x_5\} \\ &\text{s.t. } 0 \le x_5 \le 3 \\ &\min_{x_6 \in \mathbb{Z}} \{3x_6 - 5\mu_1 x_6 - \mu_2 x_6\} \\ &\text{s.t. } 0 \le x_6 \le 3. \end{aligned} \quad (70)$$

Derivation of Dual Function and Optimal Multipliers: Since the purpose of this example is to demonstrate the convergence of multipliers to their optimal values, the knowledge of the dual function and optimal multipliers is needed. The dual function is obtained by minimizing the Lagrangian function (6669) by using software Mathematica [44], which allows symbolic manipulations. Because of technical limitations that do not allow performing symbolic minimization with respect to six integer variables, the dual function is obtained iteratively. The Lagrangian function is minimized over $\{x_1, x_2, x_3\}$ and the resulting function is minimized over $\{x_4, x_5, x_6\}$. The analytical expression for the dual function then becomes

$$\begin{aligned} &q(\mu_1, \mu_2) \\ &= \min_{\{x_1, x_2, x_3, x_4, x_5, x_6\}} L(x_1, x_2, x_3, x_4, x_5, x_6, \mu_1, \mu_2) \\ &= \begin{cases} 26\mu_1 + 16\mu_2, & \text{if } \mu_1 + \mu_2 \le 0.6,\ \mu_1 + 2\mu_2 \le 1 \\ 6 + 20\mu_1 + 4\mu_2, & \text{if } \mu_1 + \mu_2 \le 0.6,\ \mu_1 + 2\mu_2 > 1 \\ 21 - 4\mu_1 - 15.5\mu_2, & \text{if } 3\mu_1 + 1.5\mu_2 > 2,\ 5\mu_1 + \mu_2 \le 3 \\ 18 + 2\mu_1 - 2\mu_2, & \text{if } 5\mu_1 + \mu_2 \le 3,\ \mu_1 + 2\mu_2 > 1, \\ & \quad 3\mu_1 + 0.5\mu_2 \le 2 \\ 30 - 19\mu_1 - 18.5\mu_2, & \text{if } 5\mu_1 + \mu_2 > 3,\ \mu_1 + 2\mu_2 > 1, \\ & \quad 3\mu_1 + 0.5\mu_2 \le 2 \\ 9 + 11\mu_1 + \mu_2, & \text{if } \mu_1 + \mu_2 > 0.6,\ 5\mu_1 + \mu_2 \le 3, \\ & \quad \mu_1 + 2\mu_2 \le 1 \\ 18 - 4\mu_1 - 2\mu_2, & \text{if } 5\mu_1 + \mu_2 > 3,\ 3\mu_1 + 0.5\mu_2 \le 2 \\ 15 + 5\mu_1 - 11\mu_2, & \text{if } \mu_1 + \mu_2 > 0.6,\ \mu_1 + 2\mu_2 > 1, \\ & \quad 3\mu_1 + 0.5\mu_2 \le 2 \\ 24 - 13\mu_1 - 6.5\mu_2, & \text{if } 3\mu_1 + 1.5\mu_2 > 2,\ \mu_1 + 2\mu_2 \le 1, \\ & \quad 3\mu_1 + 0.5\mu_2 \le 2 \\ 30 - 22\mu_1 - 8\mu_2, & \text{if } 3\mu_1 + 0.5\mu_2 > 2,\ \mu_1 + 2\mu_2 \le 1 \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \quad (71)$$

By maximizing the dual function (71) over μ_1 and μ_2 in Mathematica, the optimal dual value and optimal multipliers are obtained as

$$q(\mu_1^*, \mu_2^*) = 15.6, \quad \text{with } \mu_1^* = 0.6 \ \text{ and } \mu_2^* = 0. \quad (72)$$

Initialization: The stepsize is initialized by using [23, eq. (76), p. 190], whereby the optimal dual value q^* from (72), rather than its estimate, is used. Multipliers are initialized at zero.

Simulation: Because of the lack of distributed computing and communicating facilities, asynchronous coordination is simulated by simulating subproblem-solving, multiplier-updating, and communication times. Simulated solving and updating times are based on real times obtained by SLR first. According to the SLR results, subproblem solving times range from 2 to 115 ms with an average value of 5.36 ms. The multiplier-updating time is either 0 or 1 ms with an average value of 0.036 ms (the updating time is very short, and the time resolutions within OPL CPLEX is 1 ms). Subproblem-solving and multiplier-updating times, thus, follow empirical distributions, which for simulation purposes are used to generate solving and updating times using discrete random number generators in the MS Excel [47]. Communication time between the coordinator and subproblem solvers is randomly generated following a uniform distribution $U[0.95, 1.05]$ as the average wireless 5G speed is 1 ms.[3] Based on the above-mentioned data, absolute arrival times (the time when one subproblem solver finishes solving one subproblem + communication time) of subproblem solutions are computed. Based on these absolute timestamps, a sequence of subproblem solution arrivals at the coordinator is obtained. Given solution arrival times, the sequence, and the multiplier-updating time, the set of latest subproblem solutions used to update multipliers at each coordinator iteration is determined. Then, the time of multiplier arrivals to each subproblem solver is obtained. Given the time when one subproblem solver starts solving, appropriate multipliers to be used are also determined based on multiplier arrival times. In simulations, subproblems are solved, and multipliers are updated based on simulated sequences, which are, in turn, based on empirical distributions as described above. To test the robustness of DA-SLR, ten testing cases are generated following the above-mentioned procedure. To demonstrate the convergence of DA-SLR when there is a "slow" subsystem, another testing case with one "slow" subproblem solver is also considered. The solving time of the "slow" subproblem solver is assumed to range from 20 to 450 ms. The other five subproblem solver remain the same. For comparison purposes, one more testing case with a "slow" subsystem is also generated for sequential SLR.

Results: Distances from multipliers to the optimum, which are square a square root of Lyapunov functions, for DA-SLR (average, minimum, and maximum over ten cases) and sequential SLR are shown in Fig. 4. The results for the case with a "slow" subsystem are shown in Fig. 5.

As demonstrated in Fig. 4, the average and minimum and maximum values of Lyapunov functions within DA-SLR while nonmonotonic, approach zero fast. Moreover, distances to the

[3] https://5g.co.uk/guides/how-fast-is-5g/

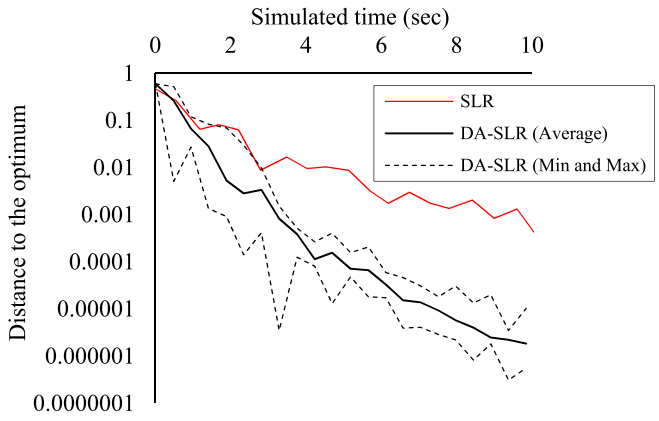

Fig. 4. Distances from multipliers to the optimum (square root of Lyapunov function) within DA-SLR and SLR.

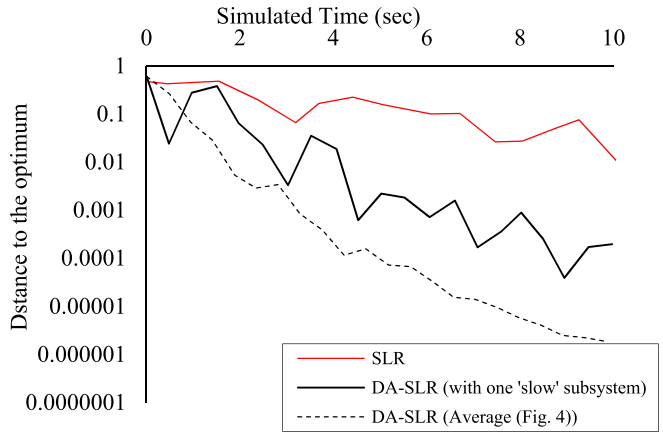

Fig. 5. Distances from multipliers to the optimum (square root of Lyapunov functions) within DA-SLR and sequential SLR for a system with one "slow" subsystem; comparison with results of Fig. 4.

optimum within DA-SLR approach zero faster, as compared with those within SLR.

As demonstrated in Fig. 5, when there is a "slow" subsystem, distances to the optimum within DA-SLR also approach zero. While in this case, the Lyapunov function approaches zero slower than within the system without "slow" subsystems, and still faster than within sequential SLR.

Example 2 (Generalized Assignment Problems [23], [24], [45], [46]): The GAP can be viewed as a futuristic and albeit simplified optimization problem that arises within "factories of tomorrow," whereby each machine or a job will have computational and communicational capabilities. The DA-SLR method will then serve as a foundation for self-optimization to efficiently coordinate machines and jobs.

Problem Formulation: Mathematically, the GAP is formulated in the following way:

$$\min_{x_{i,j}} \sum_{i=1}^{I}\sum_{j=1}^{J} g_{i,j}x_{i,j}$$
$$x_{i,j} \in \{0,1\}, \quad g_{i,j} \geq 0, \ a_{i,j} \geq 0, \ b_j \geq 0 \tag{73}$$
$$\text{s.t.} \sum_{i=1}^{I} a_{i,j}x_{i,j} \leq b_j, \quad j = 1,\ldots,J \tag{74}$$
$$\sum_{j=1}^{J} x_{i,j} = 1, \quad i = 1,\ldots,I \tag{75}$$

where I is the number of jobs and J is the number of machines, $a_{i,j}$ is the time required by machine j to perform job i, and $g_{i,j}$ is the cost of assigning job i to machine j. Capacity constraints (74) ensure that the total amount of time required by the jobs to be performed on machine j does not exceed its available time b_j. Assignment constraints (75) ensure that each job is to be performed on one and one machine only.

Relaxed Problem: After relaxing assignment constraints (75), the relaxed problem is formulated in a separable form as follows [23]:

$$\min_{x_{i,j}} L(x,\lambda) = \min_{x_{i,j}} \sum_{i=1}^{I}\sum_{j=1}^{J} (g_{i,j}+\lambda_i)x_{i,j} - \sum_{i=1}^{J}\lambda_i$$
$$\text{s.t.} \sum_{i=1}^{I} a_{i,j}x_{i,j} \leq b_j, \quad j = 1,\ldots,J$$
$$x_{i,j} \in \{0,1\}, \quad g_{i,j} \geq 0, \ a_{i,j} \geq 0, \ b_j \geq 0. \tag{76}$$

Subproblems: The above-mentioned relaxed problem (76) is decomposed into J individual machine subproblems, and subproblem j is formulated as follows:

$$\min_{x_{i,j}} \sum_{i=1}^{I} (g_{i,j}+\lambda_i)x_{i,j}$$
$$\text{s.t.} \sum_{i=1}^{I} a_{i,j}x_{i,j} \leq b_j$$
$$x_{i,j} \in \{0,1\}, \ g_{i,j} \geq 0, \ a_{i,j} \geq 0, \ b_j \geq 0. \tag{77}$$

These subproblems are solved using branch-and-cut implemented in CPLEX. The simulation follows the same process as that explained in Example 1. The resulting subproblem solving times follow uniform distributions $U[0.15, 0.20]$, and updating times follow $U[0.01, 0.02]$. Communication times follow the same 5G assumption with uniform distribution $U[0.95, 1.05]$.

Initialization: The stepsize is initialized by using [23, eq. (76), p 190], whereby an estimate of the optimal dual value q^* is used. This estimate is obtained by solving (73)–(75) after relaxing integrality requirements. Initial values of multipliers are obtained based on heuristic initialization rules following [46], whereby the second-highest cost of assigning a job is used.

Results: Because this example is complicated, optimal multipliers are difficult to obtain. Therefore, Lyapunov functions are not plotted. Rather, dual values and feasible costs obtained by using DA-SLR and sequential (SLR) and distributed and synchronous (DS-SLR) versions and are plotted in Fig. 6.

Fig. 6 demonstrates the performance of DA-SLR for the GAP d201600 instance with 20 machines and 1600 jobs. The dual value is obtained every 500 iterations by solving

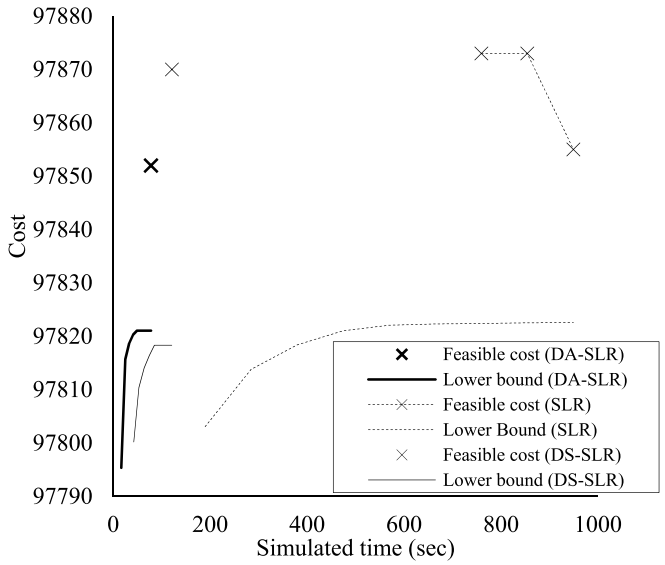

Fig. 6. Performance of DA-SLR and comparison against SLR and DS-SLR for the GAP d201600 instance.

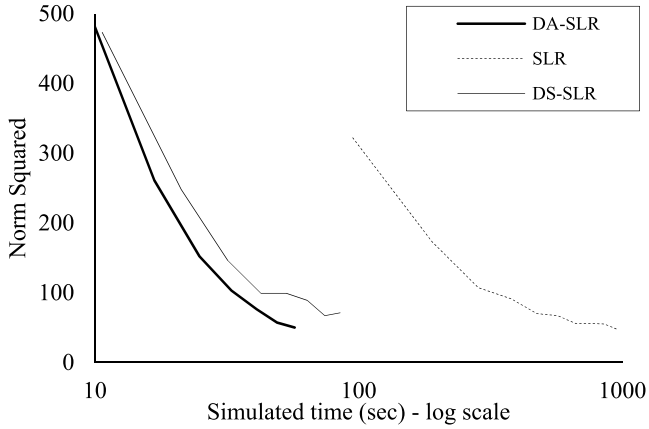

Fig. 7. Norm squared reduction within DA-SLR and comparison against SLR and DS-SLR for the GAP d201600 instance.

all subproblems to optimality.[4] As shown in Fig. 6, with asynchronous coordination, a feasible cost 97 852 is obtained with a duality gap of 0.0316% after 78 s. This demonstrates that DA-SLR converges and finds high-quality solutions significantly fast. As shown in Fig. 6, within sequential SLR, the best feasible cost 97 855 is obtained with a duality gap of 0.0332% after 950 s; within DS-SLR, the best feasible cost 97 870 is obtained with a duality gap of 0.0528% after 121 s.

As demonstrated in Fig. 7, within DA-SLR, surrogate subgradient norms reduce fast. The norm-squared reduction is faster than within DS-SLR, which translates into a better feasible cost shown in Fig. 6, and much faster than within sequential SLR, which leads to the overall drastic CPU time reduction, also shown in Fig. 6.

[4]It is expected that surrogate dual value approach dual values at convergence, but for demonstration purposes, dual values are obtained every 500 iterations.

V. Conclusion

In anticipation of trends toward self-optimizing factories, there is a need for efficient asynchronous price-based coordination of distributed subproblems. The novel DA-SLR is developed, and convergence is proven based on the novel use of Lyapunov energy function without requiring its strict monotonic decrease for convergence. Numerical results demonstrate that the novel approach converges fast. With this effective distributed and asynchronous coordination, the method has a strong potential to be used in future self-optimizing factories to coordinate machines and in future power systems to efficiently coordinate distributed energy resources.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015, doi: 10.1109/COMST.2015.2444095.

[2] S. Li, L. Xu, and S. Zhao, "The Internet of Things: A survey," *Inf. Syst. Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[3] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manuf. Lett.*, vol. 3, pp. 18–23, Jan. 2015.

[4] T. Stock and G. Seliger, "Opportunities of sustainable manufacturing in industry 4.0," *Procedia CIRP*, vol. 40, pp. 536–541, Jan. 2016.

[5] A. Giret, D. Trentesaux, and V. Prabhu, "Sustainability in manufacturing operations scheduling: A state of the art review," *J. Manuf. Syst.*, vol. 37, pp. 126–140, Oct. 2015.

[6] C. Gahm, F. Denz, M. Dirr, and A. Tuma, "Energy-efficient scheduling in manufacturing companies: A review and research framework," *Eur. J. Oper. Res.*, vol. 248, no. 3, pp. 744–757, Feb. 2016.

[7] M. L. Fisher, "Optimal solution of scheduling problems using Lagrange multipliers: Part I," *Oper. Res.*, vol. 21, pp. 1114–1127, Oct. 1973.

[8] M. L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," *Manag. Sci.*, vol. 27, pp. 1–18, Jan. 1981.

[9] M. L. Fisher, B. J. Lageweg, J. K. Lenstra, and A. H. G. R. Kan, "Surrogate duality relaxation for job shop scheduling," *Discrete Appl. Math.*, vol. 5, no. 1, pp. 65–75, Jan. 1983.

[10] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A practical approach to job-shop scheduling problems," *IEEE Trans. Robot. Autom.*, vol. 9, no. 1, pp. 1–13, Feb. 1993.

[11] X. Guan, P. B. Luh, H. Yen, and P. Rogan, "Optimization-based scheduling of hydrothermal power systems with pumped-storage units," *IEEE Trans. Power Syst.*, vol. 9, no. 2, pp. 1023–1031, May 1994.

[12] N. Z. Shor, "On the rate of convergence of the generalized gradient method," *Cybernetics*, vol. 4, no. 3, pp. 79–80, 1968.

[13] N. Z. Shor, "Generalized gradient methods for non-smooth functions and their applications to mathematical programming problems," (in Russian), *Econ. Math. Methods*, vol. 12, no. 2, pp. 337–356, 1976.

[14] A. Nedić and D. Bertsekas, "Convergence rate of incremental subgradient algorithms," in *Stochastic Optimization: Algorithms and Applications*. Boston, MA, USA: Springer, 2001, pp. 223–264.

[15] A. Nedić, D. P. Bertsekas, and V. S. Borkar, "Distributed asynchronous incremental subgradient methods," *Stud. Comput. Math.*, vol. 8, no. C, pp. 381–407, 2001.

[16] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Explicit convergence rate of a distributed alternating direction method of multipliers," *IEEE Trans. Autom. Control*, vol. 61, no. 4, pp. 892–904, Apr. 2016.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[18] E. Wei and A. Ozdaglar, "On the O(1/k) convergence of asynchronous distributed alternating direction method of multipliers," in *Proc. Global Conf. Signal Inf. Process. (GlobalSIP)*, 2013, pp. 551–554.

[19] R. Zhang and J. T. Kwok, "Asynchronous distributed ADMM for consensus optimization," in *Proc. 31th ICML*, Beijing, China, Jun. 2014, pp. 1–9.

[20] Y. Wang, L. Wu, and S. Wang, "A fully-decentralized consensus-based ADMM approach for DC-OPF with demand response," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2637–2647, Nov. 2017.
[21] Y. Wang, L. Wu, and J. Li, "A fully distributed asynchronous approach for multi-area coordinated network-constrained unit commitment," *Optim. Eng.*, vol. 19, no. 2, pp. 419–452, Jun. 2018.
[22] X. Zhao, P. B. Luh, and J. Wang, "Surrogate gradient algorithm for Lagrangian relaxation," *J. Optim. Theory Appl.*, vol. 100, no. 3, pp. 699–712, Mar. 1999.
[23] M. A. Bragin, P. B. Luh, J. H. Yan, N. Yu, and G. A. Stern, "Convergence of the surrogate Lagrangian relaxation method," *J. Optim. Theory Appl.*, vol. 164, no. 1, pp. 173–201, Jan. 2015.
[24] 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 Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 531–541, Apr. 2019.
[25] M. R. Hestenes, "Multiplier and gradient methods," *J. Optim. Theory Appl.*, vol. 4, no. 5, pp. 303–320, 1969.
[26] M. J. D. Powell, "A method for nonlinear constraints in minimization problems," in *Optimization*, R. Fletcher, Ed. New York, NY, USA: Academic, 1969.
[27] A. M. Lyapunov, "The general problem of the stability of motion," (in Russian), Ph.D. dissertation, Univ. Kharkov, Kharkiv, Ukraine, 1892.
[28] D. P. Bertsekas, *Nonlinear Programming*, 3rd ed. Belmont, MA, USA: Athena Scientific, 2016.
[29] P. B. Luh, D. Zhang, and R. N. Tomastik, "An algorithm for solving the dual problem of hydrothermal scheduling," *IEEE Trans. Power Syst.*, vol. 13, no. 2, pp. 593–600, May 1998.
[30] J.-L. Goffin and K. C. Kiwiel, "Convergence of a simple subgradient level method," *Math. Program.*, vol. 85, no. 1, pp. 207–211, May 1999.
[31] A. Nedic and D. P. Bertsekas, "Convergence rate of incremental subgradient algorithms," in *Stochastic Optimization: Algorithms and Applications*, S. Uryasev and P. M. Pardalos, Eds. New York, NY, USA: Kluwer, 2000, pp. 263–304.
[32] R. Zhang and J. T. Kwok, "Asynchronous distributed ADMM for consensus optimization," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 1701–1709.
[33] J. Yang and X. Yuan, "Linearized augmented Lagrangian and alternating direction methods for nuclear norm minimization," *Math. Comput.*, vol. 82, no. 281, pp. 301–329, 2013.
[34] W. T. Elsayed and E. F. El-Saadany, "A fully decentralized approach for solving the economic dispatch problem," *IEEE Trans. Power Syst.*, vol. 30, no. 4, pp. 2179–2189, Jul. 2015.
[35] J. E. Mitchell, "Branch-and-cut," in *Wiley Encyclopedia of Operations Research and Management Science*. Hoboken, NJ, USA: Wiley, 2010.
[36] G. B. Dantzig, "Expected number of steps of the simplex method for a linearprogram with a convexity constraint," Stanford Univ., Stanford, CA, USA, Tech. Rep. SOL 80-3, 1980.
[37] M. Brusco and S. Stahl, *Branch-and-Bound Applications in Combinatorial Data Analysis*. New York, NY, USA: Springer, 2005.
[38] A. H. Land and A. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, pp. 497–520, Jul. 1960.
[39] M. Padberg, "Classical cuts for mixed-integer programming and branch-and-cut," *Ann. Oper. Res.*, vol. 139, no. 1, pp. 321–352, 2006.
[40] R. Misener and C. A. Floudas, "Global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCQP) through piecewise-linear and edge-concave relaxations" *Math. Program. J. Comput.*, vol. 136, no. 1, pp. 155–182, May 2012.
[41] B. W. Wah and Y. X. Chen, "Subgoal partitioning and global search for solving temporal planning problems in mixed space," *Int. J. Artif. Intell. Tools*, vol. 13, no. 4, pp. 767–790, 2004.
[42] S. G. Kreĭn and N. I. Vilenkin, "Functional analysis," (in Russian), Foreign Technol. Division, Wright-Patterson Air Force Base, OH, USA, Tech. Rep. FTD-MT-65-573, 1967.
[43] R. Dıaz and E. Pariguan, "On hypergeometric functions and pochhammer k-symbol," *Divulgaciones Matemticas*, vol. 15, no. 2, pp. 179–192, 2007.
[44] *Mathematica, Version 11.3*, Wolfram Res., Champaign, IL, USA, 2018.
[45] M. Yagiura, T. Ibaraki, and F. Glover, "A path relinking approach with ejection chains for the generalized assignment problem," *Eur. J. Oper. Res.*, vol. 169, no. 2, pp. 548–569, Mar. 2006.
[46] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove, "A multiplier adjustment method for the generalized assignment problem," *Manage. Sci.*, vol. 32, no. 9, pp. 1095–1103, Sep. 1986.
[47] *Discrete Random Number Generator in Excel*. Accessed: Oct. 15, 2019. [Online]. Available: https://stackoverflow.com/questions/43226094/discrete-random-number-generator-in-excel
[48] X. Sun, P. B. Luh, M. A. Bragin, Y. Chen, J. Wan, and F. Wang, "A decomposition and coordination approach for large-scale security constrained unit commitment problems with combined cycle units," *IEEE Trans. Power Syst.*, vol. 33, no. 5, pp. 5297–5308, Sep. 2018.

Mikhail A. Bragin (Member, IEEE) received the B.S. and M.S. degrees in mathematics from Voronezh State University, Voronezh, Russia, in 2004, the M.S. degree in physics and astronomy from the University of Nebraska-Lincoln, Lincoln, NE, USA, in 2006, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Connecticut, Storrs, CT, USA, in 2014 and 2016, respectively.

He is currently an Assistant Research Professor in electrical and computer engineering with the University of Connecticut. His research interests include operations research, mathematical optimization, including power system optimization, grid integration of renewables (wind and solar), energy-based operation optimization of distributed energy systems, scheduling of manufacturing systems, and machine learning through deep neural networks.

Bing Yan (Member, IEEE) received the B.S. degree from the Renmin University of China, Beijing, China, in 2010, the M.S. and Ph.D. degrees from the University of Connecticut, Storrs, CT, USA, in 2012 and 2016, respectively.

She was an Assistant Research Professor with the Department of Electrical and Computer Engineering, University of Connecticut. She is currently an Assistant Professor with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY, USA. Her research interests include manufacturing system scheduling, power system optimization, mathematical optimization, formulation tightening, and operation optimization of microgrids and distributed energy systems.

Peter B. Luh (Life Fellow, IEEE) received the B.S. degree from National Taiwan University, Taipei, Taiwan, the M.S. degree from the Massachusetts Institute of Technology, Cambridge, MA, USA, and the Ph.D. degree from Harvard University, Cambridge, MA, USA.

He has been with the University of Connecticut, Storrs, CT, USA, since 1980, where he is currently a Board of Trustees Distinguished Professor and the SNET Professor of communications and information technologies. His research interests include intelligent manufacturing, energy-smart buildings, and smart grid.

Dr. Luh was the Chair of the IEEE Technical Activities Board (TAB) Periodicals Committee from 2018 to 2019. He is the Chair of the IEEE TAB Periodicals Review and Advisory Committee for 2020 to 2021. He is the Founding Editor-in-Chief of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.