Asynchronous Coordination of Distributed Mixed-Integer Linear Subsystems via Surrogate Lagrangian Relaxation

Mikhail Bragin [1], Peter Luh [2], and Bing Yan [2]

[1]University of Connecticut
[2]Affiliation not available

October 30, 2023

Abstract

With the emergence of 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 paper, a new "price-based" decomposition and coordination methodology is developed to efficiently coordinate subsystems such as machines and parts, which are described by Mixed-Integer Linear Programming (MILP) formulations, in a distributed and asynchronous way. To ensure low communication requirements, exchanges between the "coordinator" and subsystems are limited to "prices" (Lagrangian multipliers) broadcast by the coordinator, and to subsystem solutions sent to 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, convergence of our method is proved by innovatively extending 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.

Asynchronous Coordination of Distributed Mixed-Integer Linear Subsystems via Surrogate Lagrangian Relaxation

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

***Abstract*—With the emergence of 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 paper, a new "price-based" decomposition and coordination methodology is developed to efficiently coordinate subsystems such as machines and parts, which are described by Mixed-Integer Linear Programming (MILP) formulations, in a distributed and asynchronous way. To ensure low communication requirements, exchanges between the "coordinator" and subsystems are limited to "prices" (Lagrangian multipliers) broadcast by the coordinator, and to subsystem solutions sent to 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, convergence of our method is proved by innovatively extending 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 distributed and asynchronous method to coordinate distributed subsystems is developed. Under realistic assumptions of finite communication and solve times, method convergence is proved. 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, Surrogate Lagrangian Relaxation, Self-Optimizing Factories, Mixed-Integer Linear Programming Problems**

I. Introduction

With the emergence of 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, e.g., self-optimizing factories with interacting components and subsystems. Within these futuristic factories, 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, efficient communication scheme is required to prevent bandwidth overloading. This motivates the need for efficient coordinated operations of subsystems while ensuring high computational and communication efficiency.

Within manufacturing, machine and part subsystems are frequently formulated as mixed-integer linear programming (MILP) subproblems. Traditionally, to coordinate MILP subproblems, Lagrangian relaxation (LR) [7-11] has been used by exploiting problem separability in manufacturing problems such as job-shop scheduling [10] and in power systems problems such as unit commitment [11]. Within standard LR, multipliers (or "shadow prices") are updated using subproblem solutions based on levels of violation of relaxed constraint using subgradient methods [12-13]. Because of 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 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 LR 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 significant amount of time waiting for synchronization.

Some the above difficulties have been overcome within subgradient incremental methods [14, 15], Alternate Direction Method of Multipliers (ADMM) [16-21], surrogate subgradient method [22], and surrogate Lagrangian relaxation (SLR) [23-24, 49]. The distributed and asynchronous incremental subgradient method [15] for optimizing convex dual functions consisting of a large number of components, which arise within Lagrangian relaxation framework with a large number of subproblems, overcomes the synchronization difficulty. However, the method imposes the requirement that all

This work is supported by the National Science Foundation under grants ECCS-1509666, ECCS-1810108, and CNS-1647209. *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.*

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

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

subproblem solutions arrive to 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 Lagrangian Relaxation" (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, ADMM does not converge. Our recent SLR method [23, 24, 49] has overcome major convergence difficulties of standard Lagrangian Relaxation such as high computational effort, zigzagging of multipliers, and the need to know the optimal dual value for convergence. In [49], it has been demonstrated that the method is capable of efficiently coordinating thousands of subsystems. These methods will be reviewed in more detail in Section II.

In this paper, a novel distributed and asynchronous "price-based" decomposition and coordination method based upon the SLR method will be developed to efficiently coordinate distributed MILP subsystems within futuristic self-optimizing factories in Section III. Within the framework, multiple distributed subsystems and one coordinator have computation and communication capabilities. Information exchanges between the coordinator and subsystems are limited to "prices" (Lagrangian multipliers) broadcast by the coordinator and to subsystem solutions sent to the coordinator to avoid excessive data transfer within the system. 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 to 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 proved under a "freshness" assumption, whereby a coordinator can update multipliers without waiting for "slow" subproblems as long as all subproblem solutions arrive to the coordinator at least once within a finite number of iterations. Our novel 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 that this upper bound approaches zero.

In section IV, by simulating asynchronous update of multipliers, two examples are presented. The first small example is to show that Lyapunov functions within of the new method while non-monotonic, approach zero fast. The second example is based on generalized assignment problems, 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

In this section, standard Lagrangian Relaxation (LR) will be reviewed in subsection II.A. In subsection II.B, the distributed asynchronous incremental subgradient method as well as asynchronous ADMM, both are version of LR tailored for asynchronous coordination, will be reviewed and their limitations will be presented. In subsection II.C, our recent Surrogate Lagrangian relaxation will be reviewed as a promising approach for the development of an efficient asynchronous coordination method. Since this paper deals with coordination of MILP subsystems, branch-and-cut, an MILP method, will be reviewed in subsection II.D. Methods that do not support distributed coordination, such as heuristics methods, or the distributed methods that require continuity of problems will not be reviewed.

A. Standard Lagrangian Relaxation.

Traditionally, to solve MILP problems, Lagrangian relaxation [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]. In power systems, to solve unit commitment problems, system demand coupling constraints are relaxed to decompose the problem into individual unit subproblems [11]. Within standard LR, multipliers (or "shadow prices") are updated after receiving subproblem solutions based on levels of violation of relaxed constraint using subgradient methods [12-13]. Because of 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 non-smooth polyhedral concave [28, p. 670, Proposition 7.1.2]. Therefore, gradients may not exist and subgradients are used. As a result, multipliers may suffer from zigzagging across ridges of the dual function [23, p. 192, Fig. 1; 29, p. 594, Fig. 1]. Also, convergence proof as well 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, 199, Figs. 3-5, 7].

B. Distributed and Asynchronous Coordination Methods.

Distributed Asynchronous Incremental Subgradient Method. To optimize non-smooth 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 to the coordinator with the same "long-term"

frequency on average, and convergence was proved using the diminishing stepsizing rule. Moreover, convergence was proved 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 to satisfy the "long-term" frequency requirement.

Alternate Direction Method of Multipliers. ADMM, a decomposable version of the Method of Multipliers [25, 26] (frequently referred to as "Augmented Lagrangian Relaxation" (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 method, 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 coordinator iterations [21, 32]. The main difficulty of ADMM is that it can guarantee convergence for convex problems only [21, p. 419]. When solving non-convex problems, ADMM does not convergence [33, p. 73]. When coordinating MILP subproblems, which are non-convex, ADMM does not converge because stepsizes within the method do not approach zero. However, stepsizes are required to approach zero to guarantee convergence when optimizing non-smooth 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 [34], the minimum of penalties is typically not preserved through such linearization and performance of the method is degraded. Furthermore, penalties 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, 35] are needed.

C. Surrogate Lagrangian Relaxation Method

All major difficulties of standard LR such as high computational effort required to solve all subproblems, zigzagging of multipliers and the requirement of the knowledge of the optimal dual value, have been overcome within our recent surrogate Lagrangian relaxation (SLR) [23-24, 49]. Within the method, it is not necessary to spend the effort to optimally 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 proved 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. Additionally, a constructive stepsizing formula satisfying these criteria has been developed. When solving large-scale problems, such as unit commitment problem arising in power systems [49], 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 [36] is that if the convex hull[1] of an MILP is obtained, the problem reduces to solving a linear programming problem. Owing to linearity of the problem, the surface of the convex hull is polyhedral [41], and 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 [37, p. 6]. However, the convex hull generally cannot be obtained. After relaxing integrality requirements, branch-and-cut solves the LP-relaxed problem [37]. Attempting to obtain feasible solutions, branch-and-cut uses "cuts" to cut off 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 [38, 39] after a finite number of iterations when "tailing off" of cuts occurs [40, 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 subsection III.A, an MILP problem formulation of a system consisting of several distributed subsystems is presented. In subsection III.B, a Distributed and Asynchronous Surrogate Lagrangian Relaxation (DA-SLR) is developed. In subsection III.C, convergence of DA-SLR is proved.

A. Distributed MILP 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, which couple individual subsystems and the MILP formulation of a system can be written as follows:

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

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

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

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, $\mathbb{Z}$ 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)-(2) is non-empty. 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 (1) [42].

B. Distributed and Asynchronous Surrogate Lagrangian Relaxation

As discussed in Sections 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 decomposing the resulting relaxed problem into individual subproblems:

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

It is assumed that subsystems have computational and communication capabilities. Namely, subsystems are capable of solving their own subproblem (3) and to send the resulting solution to the coordinator.

Within the SLR framework, as discussed in Section II, it is not necessary to spend the effort to fully optimize subproblems. 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 a 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, it is also assumed that the coordinator has capability to receive subproblem solutions, update multipliers

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

and broadcast them to all subproblems. Here

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

are "surrogate" subgradient directions that are obtained instead of subgradient directions after receiving solutions from one or few subproblems. If inequality constraints are present in the formulation, multipliers are updated according to (5) with subsequent projection onto the positive orthant.

Within the asynchronous framework, subproblems are assumed to be solved without waiting for other subproblems to finish, and coordinator updates multipliers asynchronously without waiting for all subproblem solutions to arrive. Multipliers (5) will be 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:

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

with

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

For notational convenience, superscripts k of multipliers and subproblems will denote coordinator-updating iterations and superscripts of subproblems will denote the most recent subproblem solution available to the coordinator at iteration k.

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:

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

This assumption is realistic for MILP problems defined over a closed and bounded set. Indeed, surrogate subgradients are essentially levels of constraints 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 subgradient. Therefore, this solution is not guaranteed to be optimal. In this case, an iteration is skipped without updating multipliers (5) and stepsizes (7)-(8).

Figure 1 illustrates 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, 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 third subproblem is solved, and its solution arrives at the coordinator, the coordinator once again updated 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 to 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. 1, 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 the following subsection, under realistic "freshness" assumption, convergence of the DA-SLR method will be proved.

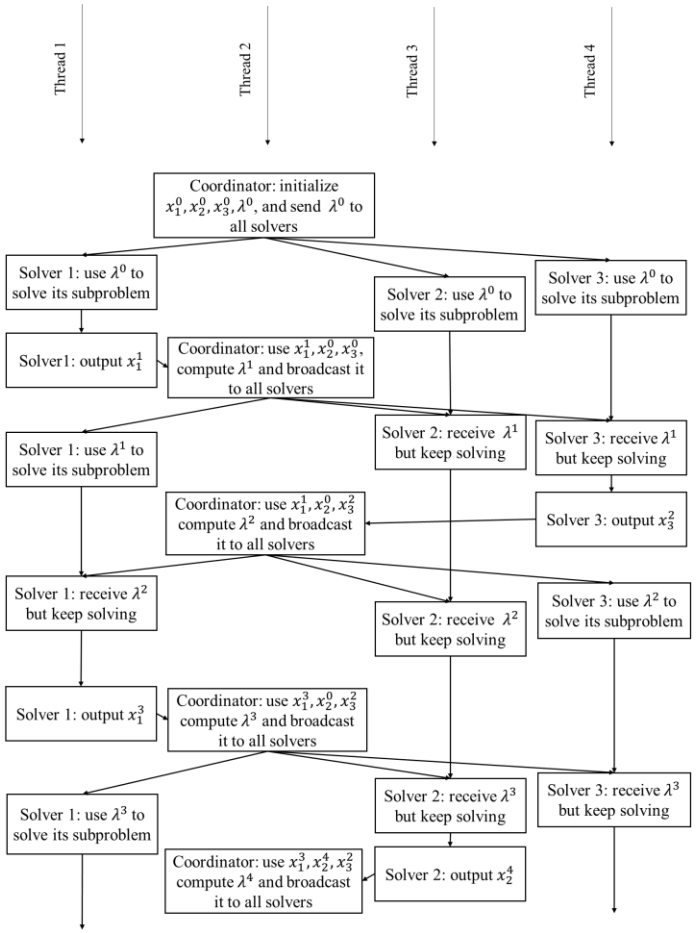

Fig. 1. Distributed and Asynchronous implementation of Surrogate Lagrangian Relaxation

C. Convergence of Distributed and Asynchronous Surrogate Lagrangian Relaxation

Convergence of Distributed and Asynchronous Surrogate Lagrangian Relaxation (DA-SLR) is proved in three stages. In Stage 1, it is proved that "surrogate" dual values approach dual values and multipliers approach the optimum "infinitely often" (Propositions 1-4). In Stage 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 5-6). In Stage 3 it is proved that the upper bound on Lyapunov functions approaches zero thereby leading to convergence of multipliers (Proposition 7).

Stage 1. Convergence of "surrogate" dual values to dual values

It is assumed that subproblem solving times as well as communication times are finite, which is equivalent to the following "freshness" Assumption:

Assumption 2. Freshness. There exists $D > 0$ such that within any consecutive D coordinator multiplier-updating iterations, all subproblem solutions arrive to 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.

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 this, Propositions 1-2 will first prove that subproblem solutions satisfying (4) converge their optimal values.

Proposition 1. Convergence to optimal subproblem solutions for fixed λ. Assuming that subproblem solutions satisfy the surrogate optimality condition (4), for each subproblem j and there exist finite K'_j such that the subproblem solution is optimal for multiplier values λ:

$$x_j^{K'_j} = x_j^*(\lambda). \tag{11}$$

Proof: As explained in subsection 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 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 proved 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}:

$$x_j^{K'_j} = x_j^*\left(\lambda^{K_j}\right). \tag{11}$$

Proof: As proved in Proposition 1, for K'_j and fixed λ^{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:

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

To prove (12), introduce the following operator:

$$A\left(f_j(x_j)+(\lambda)^T g_j(x_j)\right) = \arg\min_{x_j}\left\{f_j(x_j)+(\lambda)^T g_j(x_j)\right\}. \tag{13}$$

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

$$\left\|A\left(f_j(x_j)+(\lambda)^T g_j(x_j)\right)\right\| < \infty. \tag{14}$$

The operator A is thus bounded [43]. Therefore, there exists a finite constant $C'_A > 0$ such that the following inequality holds:

$$\left\|A\left(f_j(x_j)+(\lambda)^T g_j(x_j)\right)\right\| < C'_A\left\|\left(f_j(x_j)+(\lambda)^T g_j(x_j)\right)\right\|. \tag{15}$$

To establish (12), consider the following norm:

$$\left\|x_j^*\left(\lambda^{K'_j}\right) - x_j^*\left(\lambda^{K_j}\right)\right\|. \tag{16}$$

Using (13), equation (16) can be rewritten as:

$$\left\|A\left(f_j(x_j)+\left(\lambda^{K'_j}\right)^T g_j(x_j)\right) - A\left(f_j(x_j)+\left(\lambda^{K_j}\right)^T g_j(x_j)\right)\right\|. \tag{17}$$

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

$$\begin{aligned}&\left\|A\left(f_j(x_j)+\left(\lambda^{K'_j}\right)^T g_j(x_j)\right) - A\left(f_j(x_j)+\left(\lambda^{K_j}\right)^T g_j(x_j)\right)\right\| \le \\ &C_A\left\|\left(f_j(x_j)+\left(\lambda^{K'_j}\right)^T g_j(x_j)\right) - \left(f_j(x_j)+\left(\lambda^{K_j}\right)^T g_j(x_j)\right)\right\| = \\ &C_A\left\|\left(\left(\lambda^{K'_j}\right)^T - \left(\lambda^{K_j}\right)^T\right) g_j(x_j)\right\|.\end{aligned} \tag{18}$$

Here, C_A is a finite positive constant. Using the Cauchy-Schwarz inequality, equation (18) becomes:

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

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

$$\left\|x_j^*\left(\lambda^{K'_j}\right) - x_j^*\left(\lambda^{K_j}\right)\right\| \le C_A C\left\|\lambda^{K'_j} - \lambda^{K_j}\right\|. \tag{20}$$

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

$$s^{K'_j} < \frac{\varepsilon}{C_A C^2\left(K_j - K'_j\right)}. \tag{21}$$

Therefore,

$$\left\|\lambda^{K'_j} - \lambda^{K_j}\right\| \le \sum_{i=K'_j}^{K_j-1}\left\|\tilde{g}\left(x^i\right)\right\| s^i < \frac{C\left(K_j - K'_j\right)\varepsilon}{C_A C^2\left(K_j - K'_j\right)} = \frac{\varepsilon}{C_A C}. \tag{22}$$

From (20) and (22) is follows that

$$\left\|x_j^*\left(\lambda^{K'_j}\right) - x_j^*\left(\lambda^{K_j}\right)\right\| < \varepsilon. \tag{23}$$

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 (12) holds. Since it takes a finite number of iterations to obtain $x_j^*\left(\lambda^{K'_j}\right)$ without updating multipliers, it will also take a finite number of iterations to obtain $x_j^*\left(\lambda^{K_j}\right)$ when multipliers are updated. □

Proposition 3. Convergence of "surrogate" dual values to dual values. With stepsizing formula (7)-(8), Lagrange multipliers (5) converge to a unique fixed point

$$\lambda^k \to \bar{\lambda}, \tag{24}$$

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

$$\tilde{L}(\bar{\lambda}, x^k) \to q(\bar{\lambda}), \tag{25}$$

where

$$q(\bar{\lambda}) = \min_{x\in X} L(\bar{\lambda}, x), \tag{26}$$

is a dual value obtained by solving all subproblems optimally and

$$\tilde{L}(\bar{\lambda}, x^k) \equiv \sum_{i=1}^{I} f_i(x_i^k) + \left(\bar{\lambda}\right)^T \tilde{g}(x^k), \tag{27}$$

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

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

$$f_j\left(x_j^{k+1}\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^{k+1}\right) < f_j\left(x_j^k\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^k\right). \tag{28}$$

By using (5), inequality (28) can be rewritten as:

$$\begin{aligned}&f_j\left(x_j^{k+1}\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^{k+1}\right) < \\ &f_j\left(x_j^k\right) + \left(\lambda^k\right)^T g_j\left(x_j^k\right) + s^k\left\|g_j\left(x_j^k\right)\right\|^2.\end{aligned} \tag{29}$$

The inequality (29) can then be equivalently rewritten as:

$$\begin{aligned}&f_j\left(x_j^{k+1}\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^{k+1}\right) - f_j\left(x_j^k\right) - \left(\lambda^k\right)^T g_j\left(x_j^k\right) < \\ &s^k\left\|g_j\left(x_j^k\right)\right\|^2.\end{aligned} \tag{30}$$

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

$$\begin{aligned}&f_j\left(x_j^{k+1}\right) + \left(\lambda^{k+1}\right)^T g_j\left(x_j^{k+1}\right) - f_j\left(x_j^k\right) - \left(\lambda^k\right)^T g_j\left(x_j^k\right) < \\ &\varepsilon\left\|g_j\left(x_j^k\right)\right\|^2 < \varepsilon C^2.\end{aligned} \tag{31}$$

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

$$f_j\left(x_j^k\right) + \left(\lambda^k\right)^T g_j\left(x_j^k\right) \to f_j\left(\bar{x}_j\right) + \left(\bar{\lambda}\right)^T g_j\left(\bar{x}_j\right) < \infty. \tag{32}$$

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

$$\left(\lambda^k\right)^T g_j\left(x_j^k\right) \to \infty \tag{33}$$

is impossible. Multipliers cannot grow without bound because that would imply that there is always positive or always negative constraint violation,[2] implying infeasibility of (1)-(2), which is impossible. From Assumption 2, within any consecutive *D* iterations, all subproblem solutions arrive to the coordinator at least once. Moreover, by Propositions 1-2, 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:

$$\tilde{L}(\bar{\lambda}, x^k) \to q\left(\bar{\lambda}\right) \text{ as } s^k \to 0\,. \qquad \square \tag{34}$$

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

$$q^* - \tilde{L}(\lambda^k, x^k) \ge \nu \left\|\lambda^* - \lambda^k\right\|^2,\ k \in \aleph\,. \tag{35}$$

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

Proof: If condition (35) is not satisfied infinitely often for $\nu > 0$, then starting from iteration κ, the following inequality holds:

$$q^* - \tilde{L}(\lambda^k, x^k) < \nu \left\|\lambda^* - \lambda^k\right\|^2,\ k > \kappa\,. \tag{36}$$

There are three cases:

Case 1: The left-hand side of (36) 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 (36) is positive, and $q^* - \tilde{L}(\lambda^k, x^k) > \varepsilon$ for some $\varepsilon > 0$ and $k' > k > \kappa$, then there exists $\nu' = \dfrac{\varepsilon}{\left\|\lambda^* - \lambda^{k'}\right\|^2} > 0$, and the following condition holds:

$$q^* - \tilde{L}(\lambda^{k'}, x^{k'}) \ge \nu' \left\|\lambda^* - \lambda^{k'}\right\|^2\,. \tag{37}$$

There is a contradiction with (36) because in this case it is possible to find $\nu' > 0$ that satisfies (35).

Case 3: The left hand-side of (36) 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. □

Stage 2. Development of an upper bound for Lyapunov functions

In this Stage, the Lyapunov function is defined as

$$V\left(\lambda^k\right) = \left\|\lambda^* - \lambda^k\right\|^2, \tag{38}$$

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 5-6 as an envelope of all possible Lyapunov functions for any sequence of subproblems arriving to the coordinator. Two inequalities are derived based on whether condition (38) 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 proved in [23, p. 187], under condition (35) and assuming that stepsizes are "sufficiently small" $s^k \le 1/(2\nu)$ the following inequality holds:

$$\left\|\lambda^* - \lambda^{k+1}\right\|^2 \le \left\|\lambda^* - \lambda^k\right\|^2 \cdot (1 - 2s^k\nu) + (s^k)^2 \left\|\tilde{g}(x^k)\right\|^2\,. \tag{39}$$

If condition (35) is not satisfied or stepsizes are not "sufficiently small" $s^k > 1/(2\nu)$, then the following inequality holds:

$$\left\|\lambda^* - \lambda^{k+1}\right\|^2 \le \left\|\lambda^* - \lambda^k\right\|^2 \cdot \left(1 + s^k\beta^k \left\|\tilde{g}(x^k)\right\|\right) + (s^k)^2 \left\|\tilde{g}(x^k)\right\| \cdot \left(\frac{1}{\beta^k} + \left\|\tilde{g}(x^k)\right\|\right), \beta^k > 0. \tag{40}$$

Proof: Inequality (39) has been derived in [23, Proposition 2.5]. To derive inequality (40) consider

$$\left\|\lambda^* - \lambda^{k+1}\right\|^2 \le \left\|\lambda^* - \lambda^k\right\|^2 - 2s^k\left(\lambda^* - \lambda^k\right)\tilde{g}(x^k) + (s^k)^2\left\|\tilde{g}(x^k)\right\|^2. \tag{41}$$

By using the Cauchy-Schwarz inequality, (41) becomes:

$$\left\|\lambda^* - \lambda^{k+1}\right\|^2 \le \left\|\lambda^* - \lambda^k\right\|^2 + 2s^k\left\|\lambda^* - \lambda^k\right\| \cdot \left\|\tilde{g}(x^k)\right\| + (s^k)^2\left\|\tilde{g}(x^k)\right\|^2. \tag{42}$$

The right-hand side of (42) contains the Lyapunov function $\left\|\lambda^* - \lambda^k\right\|^2$ at iteration *k* as well as its square root $\left\|\lambda^* - \lambda^k\right\|$. In order to express the inequality (42) in terms of the Lyapunov function, the basic inequality $2ab \le \beta a^2 + \frac{1}{\beta}b^2$ [15] is used and the inequality (42) becomes

$$\left\|\lambda^* - \lambda^{k+1}\right\|^2 \le \left\|\lambda^* - \lambda^k\right\|^2 + s^k\beta^k\left\|\lambda^* - \lambda^k\right\|^2 \cdot \left\|\tilde{g}(x^k)\right\| + \frac{1}{\beta^k}(s^k)^2\left\|\tilde{g}(x^k)\right\| + (s^k)^2\left\|\tilde{g}(x^k)\right\|^2. \tag{43}$$

Therefore,

$$\left\|\lambda^* - \lambda^{k+1}\right\|^2 \le \left\|\lambda^* - \lambda^k\right\|^2 \cdot \left(1 + s^k\beta^k \left\|\tilde{g}(x^k)\right\|\right) + (s^k)^2 \left\|\tilde{g}(x^k)\right\| \cdot \left(\frac{1}{\beta^k} + \left\|\tilde{g}(x^k)\right\|\right). \ \square \tag{44}$$

In Proposition 6 below, 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.

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

Proposition 6. Upper bound for Lyapunov functions. The following upper bound is valid for Lyapunov functions:

$$\begin{aligned} &V\left(\lambda^{k+1}\right) \le \left\|\lambda^* - \lambda^0\right\|^2 \prod_{i=0}^{k} P^i + \\ &\sum_{j=0}^{k-1}\left(\left(s^j\right)^2 \left\|\tilde{g}\left(x^j\right)\right\|\left(\frac{1}{\beta^j} + \left\|\tilde{g}\left(x^j\right)\right\|\right)\prod_{l=j+1}^{k} P^l\right) + \\ &\left(s^k\right)^2 \left\|\tilde{g}\left(x^k\right)\right\|\left(\frac{1}{\beta^k} + \left\|\tilde{g}\left(x^k\right)\right\|\right), k \ge 1, \end{aligned} \tag{45}$$

where $P^i = \left(1 - 2s^i\mu\right)$ if condition (35) holds at iteration i, and $P^i = \left(1 + s^i\beta^i\left\|\tilde{g}\left(x^i\right)\right\|\right)$ 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 (35) is satisfied, then inequality (39) holds for k = 0:

$$\left\|\lambda^* - \lambda^1\right\|^2 \le \left\|\lambda^* - \lambda^0\right\|^2\left(1 - 2s^0\mu\right) + \left(s^0\right)^2\left\|\tilde{g}\left(x^0\right)\right\|^2. \tag{46}$$

If (2356) is not satisfied, then inequality (40) holds for k = 0:

$$\begin{aligned} &\left\|\lambda^* - \lambda^1\right\|^2 \le \left\|\lambda^* - \lambda^0\right\|^2\left(1 + s^0\beta^0\left\|\tilde{g}\left(x^0\right)\right\|\right) + \\ &\left(s^0\right)^2\left\|\tilde{g}\left(x^0\right)\right\|\left(\frac{1}{\beta^0} + \left\|\tilde{g}\left(x^0\right)\right\|\right), \beta^0 > 0. \end{aligned} \tag{47}$$

Since the term $\left(s^0\right)^2\left\|\tilde{g}\left(x^0\right)\right\|\left(\frac{1}{\beta^0} + \left\|\tilde{g}\left(x^0\right)\right\|\right)$ which appears in (47) is greater than $\left(s^0\right)^2\left\|\tilde{g}\left(x^0\right)\right\|^2$ which appears in (46), the following expression is the upper bound for $\left\|\lambda^* - \lambda^1\right\|^2$:

$$\begin{aligned} &\left\|\lambda^* - \lambda^1\right\|^2 \le \left\|\lambda^* - \lambda^0\right\|^2 P^0 + \\ &\left(s^0\right)^2\left\|\tilde{g}\left(x^0\right)\right\|\left(\frac{1}{\beta^0} + \left\|\tilde{g}\left(x^0\right)\right\|\right), \beta^0 > 0, \end{aligned} \tag{48}$$

where $P^0 = \left(1 - 2s^0\mu\right)$ if condition (35) holds at k = 0, and $P^0 = \left(1 + s^0\beta^0\left\|\tilde{g}\left(x^0\right)\right\|\right)$ if condition (35) does not holds at k = 0. The inequality (48) is indeed an upper bound of $\left\|\lambda^* - \lambda^1\right\|^2$ because if condition (35) does not hold, then (48) reduces to (47), and if condition (35) holds, then (48) reduces to (46) plus a positive extra term $\frac{\left(s^0\right)^2\left\|\tilde{g}\left(x^0\right)\right\|}{\beta^0}$.

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

$$\begin{aligned} &\left\|\lambda^* - \lambda^2\right\|^2 \le \left\|\lambda^* - \lambda^1\right\|^2 P^1 + \left(s^1\right)^2\left\|\tilde{g}\left(x^1\right)\right\|\left(\frac{1}{\beta^1} + \left\|\tilde{g}\left(x^1\right)\right\|\right) = \\ &\left\|\lambda^* - \lambda^0\right\|^2 P^0 P^1 + \left(s^0\right)^2\left\|\tilde{g}\left(x^0\right)\right\|\left(\frac{1}{\beta^0} + \left\|\tilde{g}\left(x^0\right)\right\|\right)P^1 + \\ &\left(s^1\right)^2\left\|\tilde{g}\left(x^1\right)\right\|\left(\frac{1}{\beta^1} + \left\|\tilde{g}\left(x^1\right)\right\|\right), \end{aligned} \tag{49}$$

where $P^1 = \left(1 - 2s^1\mu\right)$ if condition (35) holds at k = 1, and $P^0 = \left(1 + s^1\beta^1\left\|\tilde{g}\left(x^1\right)\right\|\right)$ if condition (35) does not holds at k = 1. Inequality (49) is indeed the same as inequality (45) for k = 1.

What remains to prove is that assuming that (45) holds at iteration k, it also holds for k+1:

$$\begin{aligned} &\left\|\lambda^* - \lambda^{k+2}\right\|^2 \le \left\|\lambda^* - \lambda^0\right\|^2 \prod_{i=0}^{k+1} P^i + \\ &\sum_{j=0}^{k}\left(\left(s^j\right)^2 \left\|\tilde{g}\left(x^j\right)\right\|\left(\frac{1}{\beta^j} + \left\|\tilde{g}\left(x^j\right)\right\|\right)\prod_{l=j+1}^{k+1} P^l\right) + \\ &\left(s^{k+1}\right)^2 \left\|\tilde{g}\left(x^{k+1}\right)\right\|\left(\frac{1}{\beta^{k+1}} + \left\|\tilde{g}\left(x^{k+1}\right)\right\|\right). \end{aligned} \tag{50}$$

The validity of (50), will be derived using the same logic as that used in deriving (48). Consider the following inequality:

$$\begin{aligned} &\left\|\lambda^* - \lambda^{k+2}\right\|^2 \le \left\|\lambda^* - \lambda^{k+1}\right\|^2 P^{k+1} + \\ &\left(s^{k+1}\right)^2 \left\|\tilde{g}\left(x^{k+1}\right)\right\|\left(\frac{1}{\beta^{k+1}} + \left\|\tilde{g}\left(x^{k+1}\right)\right\|\right), \end{aligned} \tag{51}$$

After substituting the expression for $\left\|\lambda^* - \lambda^{k+1}\right\|^2$ from (45) into (51) one obtains the following inequality:

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

The inequality (52) simplifies to the following:

$$\begin{aligned} &\left\|\lambda^* - \lambda^{k+2}\right\|^2 \le \left\|\lambda^* - \lambda^0\right\|^2 \prod_{i=0}^{k+1} P^i + \\ &\left(\sum_{j=0}^{k-1}\left(\left(s^j\right)^2 \left\|\tilde{g}\left(x^j\right)\right\|\left(\frac{1}{\beta^j} + \left\|\tilde{g}\left(x^j\right)\right\|\right)\prod_{l=j+1}^{k} P^l\right)\right) P^{k+1} + \\ &\left(s^k\right)^2 \left\|\tilde{g}\left(x^k\right)\right\|\left(\frac{1}{\beta^{k-1}} + \left\|\tilde{g}\left(x^k\right)\right\|\right) P^{k+1} + \\ &\left(s^{k+1}\right)^2 \left\|\tilde{g}\left(x^{k+1}\right)\right\|\left(\frac{1}{\beta^{k+1}} + \left\|\tilde{g}\left(x^{k+1}\right)\right\|\right). \end{aligned} \tag{53}$$

After further simplifications, the inequality (53) becomes:

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

The inequality (54) is the sought-for inequality (45).

Stage 3. Convergence of the upper bound to zero.

In this Stage, it is proved that the upper bound on the Lyapunov function defined in (45) asymptotically approaches zero, thereby leading to convergence of multipliers to λ^*.

Proposition 7. Proof of Convergence. If stepsizes are updated per (7)-(8), then $\lambda^k \to \lambda^*$ as $k \to \infty$.

Proof: In order to prove that $\lambda^k \to \lambda^*$, it is necessary to prove that the upper bound on Lyapunov function (right-hand side of (45)) approaches zero. This leads the Lyapunov function to converge to zero and to the convergence of multipliers.

Since λ^* that maximizes the dual function (26), is assumed to exist, the term $\left\|\lambda^* - \lambda^0\right\|^2$ is finite. Therefore, it is sufficient to prove that the following expression approaches zero:

$$\prod_{i=0}^{k-1} P^i = \prod_{i=0:i\in\mathbb{N}/\aleph}^{k-1}\left(1+s^i\beta^i\left\|\tilde{g}(x^k)\right\|\right)\prod_{i=0:i\in\aleph}^{k-1}\left(1-2s^i\mu\right), \tag{55}$$

where $\aleph$ is the set is iteration numbers whereby inequality (39) holds, and $\mathbb{N}$ is the set of natural numbers.

To prove that (55) approaches zero, the stepsizing formula (7)-(8) is plugged in first, then the resulting function is upper-bounded by using standard functions and their asymptotical representation, then, though algebraic manipulations, the condition for β^i is derived to ensure that (55) approaches zero. By exploiting the fact that set $\aleph$ is a proper subset of natural numbers $\aleph \subset \mathbb{N}$ and that each term $\left(1+s^i\beta^i\left\|\tilde{g}(x^i)\right\|\right)$ is greater than 1, the following inequality holds:

$$\begin{aligned}&\prod_{i=0:i\in\mathbb{N}/\aleph}^{k-1}\left(1+s^i\beta^i\left\|\tilde{g}(x^i)\right\|\right)\prod_{i=0:i\in\aleph}^{k-1}\left(1-2s^i\nu\right)\le\\&\prod_{i=0:i\in\mathbb{N}}^{k-1}\left(1+2\left(\prod_{j=1}^{i}\alpha_j\right)s^0\left\|\tilde{g}(x^0)\right\|\beta^i\right)\prod_{i=0:i\in\aleph}^{k-1}\left(1-2s^i\nu\right).\end{aligned} \tag{56}$$

Assuming that condition (35) is satisfied at least every N ($< \infty$) iterations the entire expression (56) is upper-bounded as:

$$\begin{aligned}&\prod_{i=0:i\in\mathbb{N}/\aleph}^{k-1}\left(1+s^i\beta^i\left\|\tilde{g}(x^i)\right\|\right)\prod_{i=0:i\in\aleph}^{k-1}\left(1-2s^i\nu\right)\le\\&\prod_{i=0:i\in\mathbb{N}}^{k-1}\left(1+2\left(\prod_{j=1}^{i}\alpha_j\right)s^0\left\|\tilde{g}(x^0)\right\|\beta^i\right)\times\\&\prod_{i=0}^{\lfloor(k-1)/N\rfloor}\left(1-2\left(\prod_{j=1}^{iN}\alpha_j\right)\frac{s^0\left\|\tilde{g}(x^0)\right\|\nu}{\left\|\tilde{g}(x^i)\right\|}\right).\end{aligned} \tag{57}$$

If such N does not exist and condition (35) does not hold infinitely often, then there is a contradiction with Proposition 3. To prove that the right-hand side of (57) approaches zero, consider α_k from (8) which asymptotically behaves as $1-\frac{1}{Mk}$ as $k \to \infty$ [23], therefore, asymptotically, the right hand-side of (57) becomes

$$\begin{aligned}&\prod_{i=0:i\in N}^{\infty}\left(1+2\prod_{j=1}^{i}\left(1-\frac{1}{Mj}\right)s^0\left\|\tilde{g}(x^0)\right\|\beta^i\right)\times\\&\prod_{i=0}^{\infty}\left(1-2\prod_{j=1}^{iN}\left(1-\frac{1}{Mj}\right)\frac{s^0\left\|\tilde{g}(x^0)\right\|\nu}{\left\|\tilde{g}(x^i)\right\|}\right).\end{aligned} \tag{58}$$

The product $\prod_{j=1}^{i}\left(1-\frac{1}{Mj}\right)$ can be expressed in terms of a "Pochhammer function," [44] which asymptotically behaves as $\left(\gamma\left(\frac{M-1}{M}\right)i^{\frac{1}{M}}\right)^{-1}$ [44] where γ is the Euler's gamma function. Therefore, asymptotically, (58) approaches the following expression:

$$\prod_{i=0}^{\infty}\left(1+\frac{2s^0\left\|\tilde{g}(x^0)\right\|\beta^i}{\gamma\left(\frac{M-1}{M}\right)i^{\frac{1}{M}}}\right)\prod_{i=0}^{\infty}\left(1-\frac{2s^0\left\|\tilde{g}(x^0)\right\|\nu}{\left\|\tilde{g}(x^{iN})\right\|\gamma\left(\frac{M-1}{M}\right)(iN)^{\frac{1}{M}}}\right). \tag{59}$$

After regrouping terms, (59) becomes

$$\prod_{j=1}^{\infty}\left(\begin{array}{l}\prod_{i=1-N+jN}^{jN}\left(1+\frac{2s^0\left\|\tilde{g}(x^0)\right\|\beta^i}{\gamma\left(\frac{M-1}{M}\right)i^{\frac{1}{M}}}\right)\times\\\left(1-\frac{2s^0\left\|\tilde{g}(x^0)\right\|\nu}{\left\|\tilde{g}(x^{jN})\right\|\gamma\left(\frac{M-1}{M}\right)(jN)^{\frac{1}{M}}}\right)\end{array}\right). \tag{60}$$

After expanding the inner product, and ignoring involving $(j)^{-2/M}$ and higher order terms, (60) becomes

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

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

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

To ensure that every terms is less than 1, consider

$$\beta^i < \frac{(1-N+jN)^{\frac{1}{M}}\nu}{N\left\|\tilde{g}(x^{jN})\right\|(jN)^{\frac{1}{M}}}, \quad i = 1-N+jN,\ldots,jN,\ j=1,2,\ldots. \tag{63}$$

The second term of the right-hand side of (45) also approaches zero, because it involves similar products as in (55), and the proof follows exactly the same logic. The last term in the right-hand side of (45) approaches zero because stepsizes approach zero. □

IV. Numerical Testing

The purpose of this section is to demonstrate performance of the Distributed and Asynchronous Surrogate Lagrangian Relaxation (DA-SLR) method. In Example 1, a small integer linear problem is considered to demonstrate that the Lyapunov function approaches zero fast. In Example 2, a generalized assignment problem with 20 machines and 1600 jobs is considered to demonstrate 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 Literature Review, subsection II.B and space limitations, comparison of DA-SLR is performed against its sequential version – SLR [23] only, which, in turn, has been shown to outperform other previous coordination methods in [24]. The DA-SLR method is implemented using IBM ILOG CPLEX Optimization Studio Version: 12.7.1.0 on a PC with 3.10GHz Intel(R) Xeon(R) CPU and 32G RAM.

Example 1. A Small Integer Programming Problem. Consider the following integer optimization problem

$$\min_{\{x_1,x_2,x_3,x_4,x_5,x_6\}\in\mathbb{Z}} \left\{x_1 + 2x_2 + 3x_3 + x_4 + 2x_5 + 3x_6\right\} \tag{64}$$

$$\begin{aligned} &s.t. \\ &x_1 + 3x_2 + 5x_3 + x_4 + 3x_5 + 5x_6 - 26 \geq 0, \\ &2x_1 + 1.5x_2 + 5x_3 + 2x_4 + 0.5x_5 + x_6 - 16 \geq 0, \\ &0 \leq x_i \leq 3, i = 1,...,6. \end{aligned} \tag{65}$$

After constraints (65) are relaxed by using multipliers μ_1 and μ_2, the Lagrangian function becomes

$$\begin{aligned} &L(x_1,x_2,x_3,x_4,x_5,x_6,\mu_1,\mu_2) = \\ &x_1 + 2x_2 + 3x_3 + x_4 + 2x_5 + 3x_6 + \\ &\mu_1(-x_1 - 3x_2 - 5x_3 - x_4 - 3x_5 - 5x_6 + 26) + \\ &\mu_2(-2x_1 - 1.5x_2 - 5x_3 - 2x_4 - 0.5x_5 - x_6 + 16). \end{aligned} \tag{66}$$

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

$$\begin{aligned} &\min_{x_1\in\mathbb{Z}}\{x_1 - \mu_1 x_1 - 2\mu_2 x_1\}, \\ &s.t. 0 \leq x_1 \leq 3, \\ &\min_{x_2\in\mathbb{Z}}\{2x_2 - 3\mu_1 x_2 - 1.5\mu_2 x_2\}, \\ &s.t. 0 \leq x_2 \leq 3, \\ &\min_{x_3\in\mathbb{Z}}\{3x_3 - 5\mu_1 x_3 - 5\mu_2 x_3\}, \\ &s.t. 0 \leq x_3 \leq 3, \\ &\min_{x_4\in\mathbb{Z}}\{x_4 - \mu_1 x_4 - 2\mu_2 x_4\}, \\ &s.t. 0 \leq x_4 \leq 3, \\ &\min_{x_5\in\mathbb{Z}}\{2x_5 - 3\mu_1 x_5 - 0.5\mu_2 x_5\}, \\ &s.t. 0 \leq x_5 \leq 3, \\ &\min_{x_6\in\mathbb{Z}}\{3x_6 - 5\mu_1 x_6 - \mu_2 x_6\}, \\ &s.t. 0 \leq x_6 \leq 3. \end{aligned} \tag{67}$$

Derivation of dual function and optimal multipliers. Since the purpose of this example is to demonstrate 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 (66) by using software Mathematica [45], which allows symbolic manipulations. Because of technical limitations that do not allow performing symbolic minimization with respect to 6 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:

$$q(\mu_1,\mu_2) = \min_{\{x_1,x_2,x_3,x_4,x_5,x_6\}} L(x_1,x_2,x_3,x_4,x_5,x_6,\mu_1,\mu_2) =$$

$$\begin{cases} 26\mu_1 + 16\mu_2, \text{ if } \mu_1 + \mu_2 \leq 0.6,\ \mu_1 + 2\mu_2 \leq 1 \\ 6 + 20\mu_1 + 4\mu_2, \text{ if } \mu_1 + \mu_2 \leq 0.6,\ \mu_1 + 2\mu_2 > 1 \\ 21 - 4\mu_1 - 15.5\mu_2, \text{ if } 3\mu_1 + 1.5\mu_2 > 2,\ 5\mu_1 + \mu_2 \leq 3 \\ 18 + 2\mu_1 - 2\mu_2, \text{ if } 5\mu_1 + \mu_2 \leq 3,\ \mu_1 + 2\mu_2 > 1,\ 3\mu_1 + 0.5\mu_2 \leq 2 \\ 30 - 19\mu_1 - 18.5\mu_2, \text{ if } 5\mu_1 + \mu_2 > 3,\ \mu_1 + 2\mu_2 > 1,\ 3\mu_1 + 0.5\mu_2 \leq 2 \\ 9 + 11\mu_1 + \mu_2, \text{ if } \mu_1 + \mu_2 > 0.6,\ 5\mu_1 + \mu_2 \leq 3,\ \mu_1 + 2\mu_2 \leq 1 \\ 18 - 4\mu_1 - 2\mu_2, \text{ if } 5\mu_1 + \mu_2 > 3,\ 3\mu_1 + 0.5\mu_2 \leq 2 \\ 15 + 5\mu_1 - 11\mu_2, \text{ if } \mu_1 + \mu_2 > 0.6,\ \mu_1 + 2\mu_2 > 1,\ 3\mu_1 + 0.5\mu_2 \leq 2 \\ 24 - 13\mu_1 - 6.5\mu_2, \text{ if } 3\mu_1 + 1.5\mu_2 > 2,\ \mu_1 + 2\mu_2 \leq 1,\ 3\mu_1 + 0.5\mu_2 \leq 2 \\ 30 - 22\mu_1 - 8\mu_2, \text{ if } 3\mu_1 + 0.5\mu_2 > 2,\ \mu_1 + 2\mu_2 \leq 1 \\ 0, \text{ otherwise.} \end{cases} \tag{68}$$

By maximizing the dual function (68) over μ_1 and μ_2 in Mathematica, the optimal dual value and optimal multipliers are obtained as:

$$q(\mu_1^*,\mu_2^*) = 15.6, \text{ with } \mu_1^* = 0.6 \text{ and } \mu_2^* = 0. \tag{69}$$

Initialization. The stepsize is initialized by using [23, eq. (76), p. 190], whereby the optimal dual value q^* from (69), 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 an updating times are based on real times obtained by SLR first.

According to the SLR results, subproblem solving times range from 2 millisecond (ms) 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 MS Excel [48]. 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 data, absolute arrival times (time when one subproblem solver finishes solving one subproblem + communication time) of subproblem solutions are computed. Based on these absolute time stamps, a sequence of subproblem solution arrivals to 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 robustness of the method DA-SLR, 10 testing cases are generated following the above procedure. To demonstrate

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

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 ms to 450 ms. 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 roof of Lyapunov functions, for DA-SLR (average, minimum and maximum over 10 cases) and SLR are shown in Fig. 2. The results for the case with a "slow" subsystem are shown in Fig. 3.

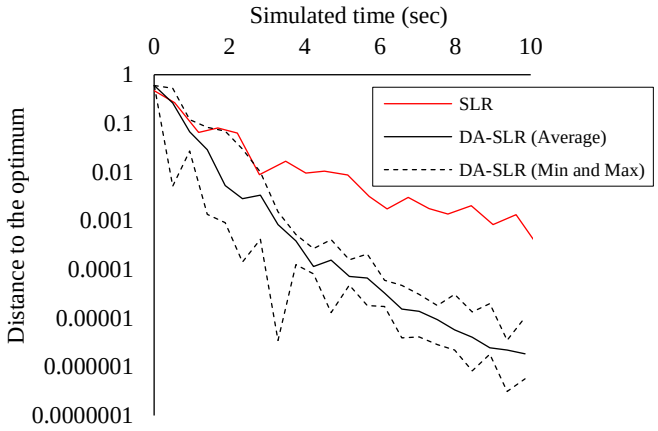

Fig. 2. Distances from multipliers to the optimum (square root of Lyapunov function) within DA-SLR and SLR

As demonstrated in Fig. 2, average as well as minimum and maximum values of Lyapunov functions within DA-SLR while non-monotonic, approach zero fast. Moreover, distances to the optimum within DA-SLR approach zero faster, as compared to those within SLR.

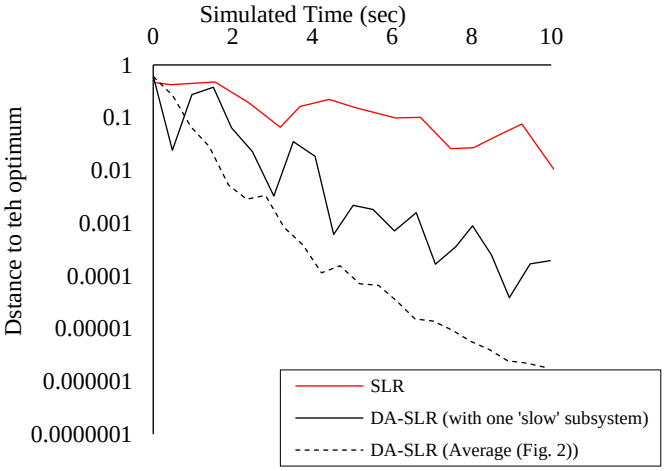

Fig. 3. Distances from multipliers to the optimum (square root of Lyapunov functions) within DA-SLR and SLR for a system with one "slow" subsystem; comparison with results of Fig 2.

As demonstrated in Fig. 3, 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 SLR.

Example 2. Generalized Assignment Problems [23, 24, 46, 47]. The Generalized Assignment Problem (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 generalized assignment problem is formulated in the following way:

$$\min_{x_{i,j}} \sum_{i=1}^{I}\sum_{j=1}^{J} g_{i,j}x_{i,j}\ , \qquad (70)$$
$$x_{i,j}\in\{0,1\},\ g_{i,j}\ge 0,\ a_{i,j}\ge 0,\ b_j\ge 0\ ,$$

$$s.t.\ \sum_{i=1}^{I} a_{i,j}x_{i,j}\le b_j,\ j=1,...,J\ , \qquad (71)$$

$$\sum_{j=1}^{J} x_{i,j}=1, i=1,...,I\ , \qquad (72)$$

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 (71) 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 (71) ensure that each job is to be performed on one and one machine only.

Relaxed problem. After relaxing assignment constraints (72), the relaxed problem is formulated in a separable form as follows [23]:

$$\min_{x_{i,j}} L(x,\lambda)=\min_{x_{i,j}}\sum_{i=1}^{I}\sum_{j=1}^{J}\left(g_{i,j}+\lambda_i\right)x_{i,j}-\sum_{i=1}^{J}\lambda_i\ ,$$
$$s.t.\ \sum_{i=1}^{I} a_{i,j}x_{i,j}\le b_j,\ j=1,\ ...,\ J \qquad (73)$$
$$x_{i,j}\in\{0,1\},\ g_{i,j}\ge 0,\ a_{i,j}\ge 0,\ b_j\ge 0\ .$$

Subproblems. The above relaxed problem (73) is decomposed into J individual machine subproblems, and subproblem j is formulated as follows:

$$\min_{x_{i,j}}\sum_{i=1}^{I}\left(g_{i,j}+\lambda_i\right)x_{i,j}\ ,\ s.t.\ \sum_{i=1}^{I}a_{i,j}x_{i,j}\le b_j, \qquad (74)$$
$$x_{i,j}\in\{0,1\},\ g_{i,j}\ge 0,\ a_{i,j}\ge 0,\ b_j\ge 0\ .$$

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 (70)-(72) after relaxing integrality requirements. Initial values of multipliers are obtained based on heuristic initialization rules following [47], 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 SLR and are plotted in Fig. 4

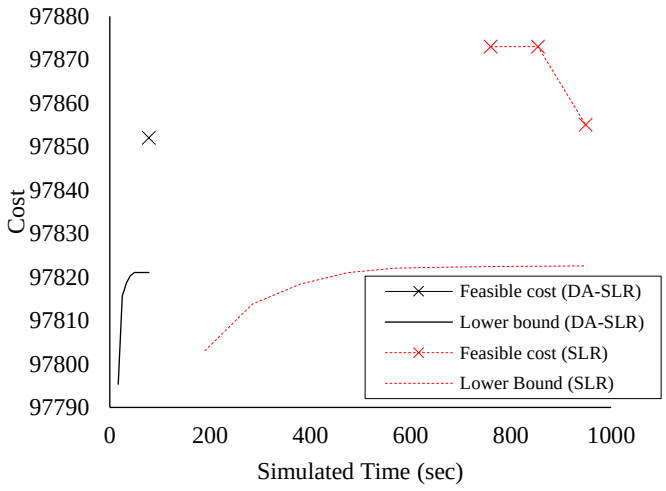

Fig. 4. Performance of DA-SLR and comparison against SLR using parameters M = 75 and r = 0.05 for solving the GAP d201600 instance

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

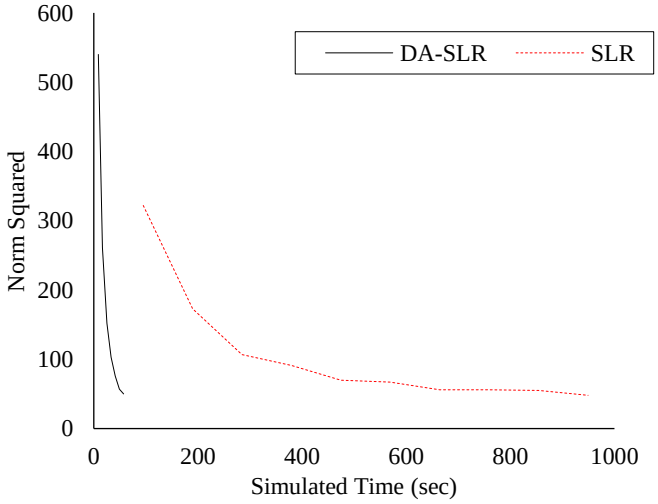

Fig. 5. Performance of DA-SLR and comparison against SLR using parameters M = 75 and r = 0.05 for solving the GAP d201600 instance

As demonstrated in Fig. 5, within DA-SLR surrogate subgradient norms reduce fast, and faster than within its sequential SLR version.

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 distributed and asynchronous Surrogate Lagrangian Relaxation is developed and convergence is proved 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," in *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347-2376, Fourthquarter 2015. doi: 10.1109/COMST.2015.2444095
2. S. Li, L.D. Xu, and S. Zhao, "The internet of things: a survey," *Inf. Syst. Front.*, vol. 17, no. 2, pp. 243–259, 2015. https://doi.org/10.1007/s10796-014-9492
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, 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, 2016.
7. M. L. Fisher, "Optimal solution of scheduling problems using Lagrange multipliers, Part I," *Operations Res.*, vol. 21, pp. 1114-1127, 1973.
8. M. L. Fisher, "Lagrangian relaxation method for solving integer programming problems," *Manag. Sci.*, vol. 27, pp. 1-18, 1981.
9. M. L. Fisher, B. J. Lageweg, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Surrogate duality relaxation for job shop scheduling," *Discrete Appl. Math.*, vol. 5, pp. 65-75, Jan. 1983.
10. D. J. Hoitomt, P. B. Luh, K. R. Pattipati, "A Practical Approach to Job Shop Scheduling Problems," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 1, pp. 1-13, February 1993.
11. X. Guan, P. B. Luh, H. Yan and P. M. Rogan, "Optimization-based Scheduling of Hydrothermal Power Systems with Pumped-storage Units," *IEEE Trans. Power Syst.*, vol. 9, no. 2, pp. 1023-1031, 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," *Econ. Math. Methods*, vol. 12, no. 2, pp. 337–356, 1976 (in Russian)
14. A. Nedić and D. Bertsekas, "Convergence Rate of Incremental Subgradient Algorithms," in *Stochastic Optimization: Algorithms and Applications*, pp. 223-264, Springer, Boston, MA, 2001
15. A. Nedić, D. P. Bertsekas and V. S. Borkar, "Distributed Asynchronous Incremental Subgradient Methods," *Studies in Computational Mathematics*, vol. 8, 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 Transactions on Automatic Control*, vol. 61, no. 4, pp. 892-904, 2016
17. E. Wei and A. Ozdaglar, "On the O(1/k) Convergence of Asynchronous Distributed Alternating Direction Method of Multipliers," *In Global conference on signal and information processing* (GlobalSIP), pp. 551-554, 2013
18. S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of

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

Multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1-122, 2010
19. R. Zhang and J. T. Kwok, "Asynchronous distributed ADMM for consensus optimization," in *Proc. 31th ICML*, Beijing, China, Jun. 21–26, 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 Transactions on Smart Grid*, vol. 8, no. 6, pp. 1–11, 2016
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, pp. 419–452, 2018.
22. X. Zhao, P. B. Luh and J. Wang, "Surrogate Gradient Algorithm for Lagrangian Relaxation," *Journal of Optimization Theory and Applications*, vol. 100, no. 3, pp. 699–712, 1999
23. M. A. Bragin, P. B. Luh, J. H. Yan, N. Yu and G. A. Stern, "Convergence of the Surrogate Lagrangian Relaxation Method," *Journal of Optimization Theory and Applications*, vol. 164, no. 1, pp. 173-201, 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, Jun. 2018 doi: 10.1109/TASE.2018.2835298
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), Doctoral dissertation, Univ. Kharkov 1892 English translations: (1) Stability of Motion, Academic Press, New-York & London, 1966 (2) The General Problem of the Stability of Motion, (A. T. Fuller trans.) Taylor & Francis, London 1992.
28. D. P. Bertsekas, Nonlinear Programming, 3rd Edition, Athena Scientific, 2016.
29. P. B. Luh, D. Zhang, R. N. Tomastik, "An Algorithm for Solving the Dual Problem of Hydrothermal Scheduling," *IEEE Transactions on Power Systems*, мol. 13, то. 2, pp. 593-600, May 1998.
30. J.-L. Goffin and K. Kiwiel, "Convergence of a simple subgradient level method," *Math. Program.*, vol. 85, no. 11, pp. 207–211, 1999.
31. A. Nedic, and D. P. Bertsekas, "Convergence rate of incremental subgradient algorithms," In: Uryasev, S., Pardalos, P.M. (eds.) Stochastic Optimization: Algorithms and Applications, pp. 263–304. Kluwer Academic, New York, 2000.
32. R. Zhang and J. T. Kwok, "Asynchronous Distributed ADMM for Consensus Optimization," *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp.1701-1709, 2014.
33. 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.
34. J. Yang and X. Yuan, "Linearized Augmented Lagrangian and Alternating Direction Methods for Nuclear Norm Minimization," *Math. Computation*, vol. 82, pp. 301-329, 2013.
35. 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.
36. J. E. Mitchell, "Branch-and-cut," in Wiley Encyclopedia of Operations Research and Management Science. Hoboken, NJ, USA: Wiley, 2010.
37. G. B. Dantzig, "Expected number of steps of the simplex method for a linear program with a convexity constraint," *Technical Report SOL 80-3*, Stanford University, 1980.
38. M. Brusco and S. Stahl, Branch-and-Bound Applications in Combinatorial Data Analysis. Springer, 2005.
39. A. H. Land and A. Doig, "An automatic method of solving discrete programming problems" *Econometrica*, vol. 28, pp. 497-520, July 1960.
40. M. Padberg, "Classical cuts for mixed-integer programming and branch-andcut," *Ann. Oper. Res*., vol. 139, pp. 321–352, 2006
41. R. Misener and A. F. Christodoulos "Global Optimization of Mixed-integer Quadratically Constrained Quadratic Programs (MIQCQP) through Piecewise-linear and Edge-concave Relaxations," *Mathematical Programming Journal on Computing*, vol. 136, no. 1, pp. 155-182, May 2012.
42. B. W. Wah, Y. X. Chen, "Subgoal Partitioning and Global Search for Solving Temporal Planning Problems in Mixed Space," *International Journal of Artificial Intelligence Tools*, vol. 13, no. 4, pp. 767-790, 2004
43. S. G. Kreĭn, and N. ĬA. Vilenkin, Functional analysis, Foreign Technology Division, Wright-Patterson Air Force Base, Ohio, 1967. (Translation from Russian)
44. R. Dıaz, and E. Pariguan, "On Hypergeometric Functions and Pochhammer k-symbol," *Divulgaciones Matemticas*, vol. 15, no. 2, pp. 179-192, 2007.
45. Wolfram Research, Inc., "Mathematica, Version 11.3," Wolfram Research, Inc., Champaign, Illinois, 2018
46. 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, 2006
47. M. L. Fisher, R. Jaikumar and L. N. Van Wassenhove, "A Multiplier Adjustment Method for the Generalized Assignment Problem," *Management Science*, vol. 32, no. 9, pp. 1095-1103, 1986
48. Discrete random number generator in Excel, https://stackoverflow.com/questions/43226094/discrete-random-number-generator-in-excel
49. 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, issue 5, September 2018, pp. 5297-5308

Mikhail A. Bragin (S'11-M'17) received his B.S. and M.S. degrees in Mathematics from the Voronezh State University, Russia, in 2004, the M.S. degree in Physics and Astronomy from the University of Nebraska-Lincoln, USA, in 2006, and the M.S. and Ph.D. degree in Electrical and Computer Engineering from the University of Connecticut, USA, in 2014 and 2016, respectively. He is an Assistant Research Professor in electrical and computer engineering at 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 (S'11-M'17) received the B.S. degree from Renmin University of China in 2010, M.S. and Ph.D. degrees from University of Connecticut in 2012 and 2016, respectively. She is currently an Assistant Professor in the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology. Before joining Rochester Institute of Technology, she was an Assistant Research Professor in the Department of Electrical and Computer Engineering, University of Connecticut. Her research interests include power system optimization, manufacturing system scheduling, mathematical optimization, formulation tightening, operation optimization of microgrids and distributed energy systems, and grid integration of renewables (wind and solar).

Peter B. Luh (S'77–M'80–SM'91–F'95-LF'16) received his B.S. degree from National Taiwan University, M.S. degree from M.I.T., and Ph.D. degree from Harvard University. He has been with the University of Connecticut since 1980, and is the SNET Professor of communications & information technologies. His interests include smart power systems – smart grid, design of auction methods for electricity markets, effective renewable (wind and solar) integration to the grid, electricity load and price forecasting with demand response, and micro grid. He is a fellow of IEEE, was the vice president of publication activities for the IEEE Robotics and Automation Society.