IEEE Control Systems Letters paper presented at
2023 American Control Conference (ACC)
San Diego, CA, USA. May 31 - June 2, 2023

Resolving large-scale control and optimization through network structure analysis and decomposition: A tutorial review

Wentao Tang[1], Andrew Allman[2], Ilias Mitrai[3] and Prodromos Daoutidis[3]

Abstract— **Decomposition is a fundamental principle of resolving complexity by scale, which is utilized in a variety of decomposition-based algorithms for control and optimization. In this paper, we aim to give a tutorial review of the following aspects: (i) how to decompose a network representing a control or optimization problem according to its latent block structure, (ii) how decomposition is determined for distributed control, and (iii) how optimization problems are solved under decomposition. Directions for further developing decomposition methods and decomposition-based control and optimization algorithms are also discussed.**

I. Introduction

Large-scale systems are ubiquitous in natural and engineered systems, including but not limited to metabolic networks [1], power grids [2], transportation networks [3], and social networks [4]. Chemical processes, due to the need of higher process efficiency and sustainability, are also increasingly being mass- and energy-integrated and intensified, resulting in process networks that are challenging to control and optimize [5]. Besides the size of the systems, the formulation of decision making problems as dynamic programming [6] or multi-stage stochastic programming problems [7] in prediction horizons leads to many-fold increase in the problem size.

In the early studies of optimal control of process networks (late 1970s), Morari, Arkun, and Stephanopoulos [8] noted that

> "*Decomposition is the underlying, guiding principle, leading to the classification of the control objectives and the partitioning of the process for the practical implementation of the control structures.*"

They classified decompositions into a vertical "multilayer" one and a horizontal "multiechelon" one.

- In the multilayer category, the objective of optimal control is decomposed into self-organization (target optimization), adaptation (disturbance rejection), optimization (dynamic optimization), and regulation layers. The frequencies of executing these tasks are dispersed into different time scales. This is a natural decomposition entailed in the entire "planning – scheduling – control – monitoring" decision making hierarchy for process systems. For integrated decision making across the hierarchy (e.g., the integrated scheduling and control problems for processes with fast transitions), this vertical decomposition is frequently used [9]–[11].
- In the multiechelon category, the plant's flowsheet is partitioned into multiple parts, each controlled by its corresponding regulators and dynamically optimized by its own solver routine. The optimizers are then coordinated for the optimum of the monolithic system. Such coordination typically makes use of the duality theory of optimization problems, i.e., uses Lagrangian multipliers as the leverage to iterate subsystem solutions.

For the coordination of multiple optimization solvers, primal-dual algorithms including Benders decomposition and Lagrangian decomposition were recognized as the most useful approaches for decomposition-based solution of process operation problems, especially those involving integer variables [12]. Studies in the electrical engineering community in parallel, focusing on continuous optimization problems in a setting of multi-agent decision making over communication networks, proposed the use of decentralized gradient-based approaches [13]–[15] and, more recently, algorithms based on the alternating direction method of multipliers (ADMM) [16]–[19].

The search for a high-quality decomposition for control and optimization-based decision making, however, lacks a common framework. This is first due to the different objectives for decomposition in different problem settings, e.g., stability in control problems, optimality in mathematical programming, and computational efficiency. Also, the decomposition problem is of combinatorial complexity, which makes it difficult to optimize. Especially for large-scale systems, the decomposition approach must be systematic, automated, and highly scalable. These requirements are hard to meet in conjunction with other objectives such as stability and optimality, due to the obscure relations between the decomposition configuration and these additional objectives.

With these considerations, in this tutorial review, we focus on the idea of representing the system or problem to decompose as a network and analyzing its block structure. This idea initially stemmed from our studies on distributed model predictive control (MPC) for integrated chemical processes [20] and sparse control for modular networks [21], [22], and was subsequently extended to optimization problems

*This work was supported by NC State University faculty startup funds (W. Tang), the NSF CAREER award (CBET #2237284, A. Allman) and NSF-CBET (Award #1926303, P. Daoutidis).

[1]Wentao Tang is with the Department of Chemical and Biomolecular Engineering, North Carolina State University, Raleigh, NC 27695, USA. Email: wtang23@ncsu.edu

[2]Andrew Allman is with the Department of Chemical Engineering, University of Michigan, Ann Arbor, MI 48109, USA. Email: allmanaa@umich.edu

[3]Ilias Mitrai and Prodromos Daoutidis are with the Department of Chemical Engineering and Materials Science, University of Minnesota, Minneapolis, MN 55455, USA. Emails: mitra047@umn.edu, daout001@umn.edu

in general [23]. The most significant advantages of this framework lie in its expressiveness of capturing interaction patterns of interest inside large networks and its effectiveness of leveraging network science algorithms to partition the system into statistically significant block structures. A software package has been developed for generating such decompositions automatically [24] and is being continually updated. The network-based decomposition has also been successfully applied to plantwide nonlinear MPC on an industrial scale [25].

The remainder of this paper is organized as follows. In Section II, we introduce the preliminaries on defining and detecting block structures in networks. Then we discuss in two following sections (Section III and Section IV) the application of this principle to large-scale control and optimization problems, respectively. Perspectives on ongoing and future research directions are provided in Section V and conclusions are made in Section VI.

II. Block Structures in Networks

To determine a decomposition for large-scale control and optimization problems in an automatic and systematic way, one needs an efficient approach to handle the complex information contained in the interactions underlying the system. To this end, the perspective of *network science* is naturally useful. Regarded as a subfield of physics, network science studies the organization of large-scale networks by investigating its macroscopic and statistical topological features and the dynamics associated with them [26]. In particular, the pattern of interest to decomposing a network should be a mesoscale one [27] – neither associated with a few individual nodes or edges nor only exhibited over the entire network. We should specifically focus on block structures, i.e., subnetworks so that the interactions inside them and across them have some regularities.

A. Stochastic Block Model

To provide a definition of the block structures, we introduce the *stochastic block model* for networks. Let us consider a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1, 2, \ldots, n\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ stand for the set of nodes and edges, respectively. Generally the topology can be captured by an adjacency matrix A with order $n = |\mathcal{V}|$. For simplicity we will assume that this is an unweighted directed network, i.e., $(i, j) \in \mathcal{E}$ does not necessarily imply that $(j, i) \in \mathcal{E}$; in such a case, we may define $A_{ij} = 1$ if there is an edge from node i to node j. Nevertheless, the following framework is extensible to weighted, undirected, or bipartite networks as well.

We view the network to be analyzed as the observed result of a generation mechanism, which is called the stochastic block model (SBM) [28], [29]. In SBM:

- The n nodes are assumed to have latent affiliations to a number of blocks. We denote by $b_i = k$ ($i \in \{1, \ldots, n\}$, $k = 1, \ldots, B$, B being the total number of blocks) to indicate that the node i belongs to the k-th block.
- When the edges are created, the nodes in different blocks have different inclinations to connect, and such inclinations depend on the nodes' block affiliations. In particular, it is assumed that the number of edges from any node in block r to any node in block s is assumed to be Poisson distributed, with a parameter ω_{rs}.

Denoting the vector $b = (b_1, \ldots, b_n)$ and matrix $\omega = [\omega_{rs}] \in \mathbb{R}_{>0}^{B \times B}$, and assuming that all the edges are independently generated, we should have the probability of observing the observed network A:

$$P(A|\omega, b) = \prod_{\substack{i,j=1 \\ i \neq j}}^{n} \frac{\omega_{b_i b_j}^{A_{ij}}}{A_{ij}!} e^{-\omega_{b_i b_j}}. \tag{1}$$

We have excluded the terms for $i = j$ since we assume that there can be no self-edges.

As Karrer and Newman [30] pointed out, the SBM does not account for the heterogeneity of degrees (total number of edges) among individual nodes. In other words, if one "extroverted" node has more edges incident to it than another "introverted" node, then the former node should naturally have more edges with other nodes in all blocks. Another interpretation is that the degree should be considered as an attribute of the node irrelevant to its block affiliation. As we focus on directed networks, every node has an out-degree and in-degree, denoted as

$$k_i^+ = \sum_{j=1}^{n} A_{ij}, \quad k_i^- = \sum_{j=1}^{n} A_{ji}. \tag{2}$$

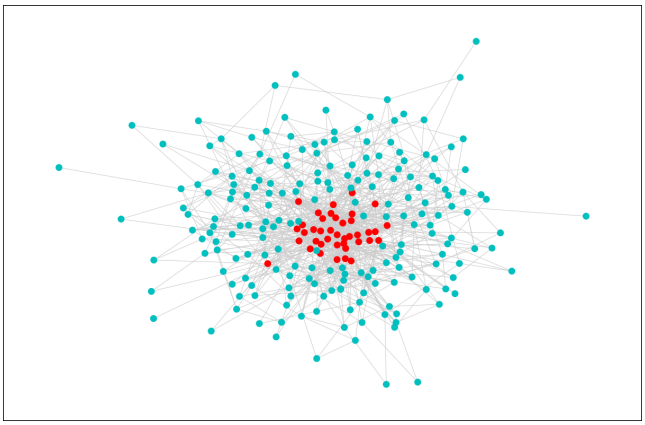
From this point of view, another group of parameters are added, i.e., the expected out-degrees and in-degrees of nodes, which we denote by $\theta = (\theta_1^+, \theta_1^-, \ldots, \theta_n^+, \theta_n^-)$. The Poisson parameter for the edge distribution from node i to node j, $\omega_{b_i b_j}$, is correspondingly modified to $\theta_i^+ \theta_j^- \omega_{b_i b_j}$. That is,

$$P(A|\omega, b) = \prod_{\substack{i,j=1 \\ i \neq j}}^{n} \frac{\theta_i^+ \theta_j^- \omega_{b_i b_j}^{A_{ij}}}{A_{ij}!} e^{-\theta_i^+ \theta_j^- \omega_{b_i b_j}}. \tag{3}$$

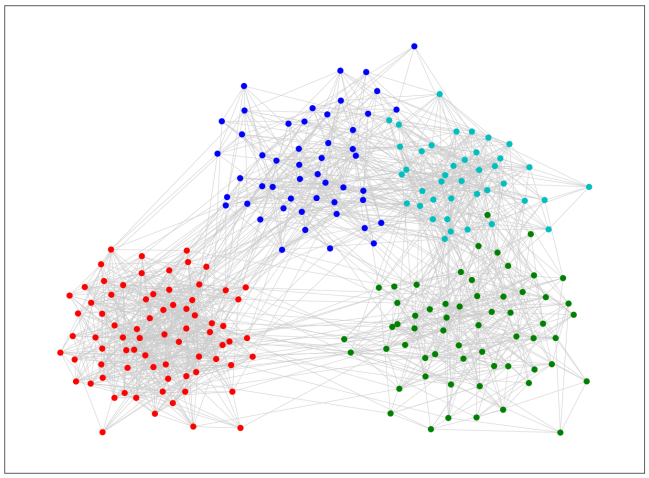
This is called the degree-corrected stochastic block model (DC-SBM). Another extension of SBM that can capture multiscale network structures is to assume that the network is generated by a nested sequence of SBMs. Under this assumption, one can compute the probability to observe the network itself and the entire hierarchy [31].

From SBM to DC-SBM or nested SBM, we can use, e.g., a maximum likelihood estimation [32] or a Bayesian estimation scheme [33] to infer the values of ω and b from the observed A, thus revealing the latent block structure of the network. The advantage of the Bayesian approach lies in its incorporation of a prior probability distribution, which can regulate the complexity of the model and improve the consistency [34], [35].

Fig. 1 illustrates two networks generated under the mechanism of stochastic block models, where the blocks of nodes are labeled by different colors. The pattern shown in Subfig. 1a is called a *core-periphery structure* [36]. The core refers to the small fraction of nodes (in red) which are internally strongly connected and concentrate the majority



(a) A network with core-periphery structure.



(b) A network with communities.

Fig. 1: Networks generated by stochastic block models.

of connections in the network; the periphery refers to the remaining part that makes up the most of the nodes but is loosely connected. In Subfig. 1b, the nodes are affiliated to multiple assortative groups, called *communities* [37]. Within these communities the connections are dense, while the interactions across the communities are much looser. Hybrid core-periphery/community and any other arbitrary structure can also be generated with SBM. In the following subsection, we will discuss in more detail how to detect the latent community structure within a network.

B. Community Detection

Community structure is a special type of latent block structure. For community detection, the formulation may be simplified from the general form (3). Let us assume that the connection propensities ω are known, thus simplifying the problem to inferring only the community affiliations b. In particular, since communities are assortative, let us say that the internal and external connections should have, correspondingly, two homogeneous propensities ω_{in} and ω_{out}. Following the proof of Newman [38], the maximum likelihood estimation will lead to the following problem

$$\max_{b} \; Q_\gamma(b) = \frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij} - \gamma\frac{k_i^+ k_j^-}{m}\right)\delta_{b_i b_j}, \qquad (4)$$

in which m is the total number of edges, and δ is the Kronecker's delta, i.e., $\delta_{b_i b_j} = 1$ if nodes i and j belong to the same community and 0 otherwise. The Q function of community assignments $b = (b_1, \ldots, b_n)$, which is to be maximized, is called the *modularity*. It was initially proposed by Newman and Girvan [39] (thus known as Newman-Girvan modularity) and extended to bipartite [40] and directed [41] networks.

Here the parameter γ is determined by ω_{in} and ω_{out} according to the relation

$$\gamma = \frac{\omega_{\text{in}} - \omega_{\text{out}}}{\ln \omega_{\text{in}} - \ln \omega_{\text{out}}}. \qquad (5)$$

It can be viewed as a *resolution parameter* [42] as it tunes the coarseness of the decomposition. When γ is extremely small, the maximization of modularity (4), which is dominated by the count of edges inside the communities, will naturally attract nodes into a single community. On the other hand, when γ is extremely large, the penalty term will dominate, thus pushing the nodes away from each other and forming communities of individual nodes. A particular case is $\gamma = 1$, which conforms to the original form of Newman and Girvan. In this case, the penalized term in parenthesis, $k_i^+ k_j^- / m$ can be interpreted as the expected number of edges from i to j when all the outgoing edges of i and incoming edges j are randomly redistributed. In other words, $Q_1(b)$ captures how much the network appears to be structured more than randomized, if the nodes $1, \ldots, n$ belong to the b_i-th community, respectively.

The maximization of modularity, apparently, is combinatorial, and actually has been proved to be an NP-hard problem [43]. Practically well-performing and computationally efficient heuristic approaches have been proposed to seek approximations to its solution. An exhaustive overview of the existing approaches was given in Fortunato and Hric [44], including the ones not under the umbrella of stochastic block model or modularity maximization. We would like to highlight two algorithms.

1) Newman's spectral algorithm [45] – All nodes start in a single community. Large communities are recursively divided into two (bisectioned) as long as the bisectioning results in a modularity increase. To determine the partition of, e.g., the entire network $\mathcal{G}$, into two subgraphs (denoted as $\mathcal{G}_+$ and $\mathcal{G}_-$), it can be shown that the corresponding modularity change is

$$\Delta Q = \frac{1}{2}\left(s^\top C s - e^\top C e\right), \qquad (6)$$

where $s_i = +1$ if $i \in \mathcal{G}_+$ and -1 if $i \in \mathcal{G}_-$, e is a vector of all ones, and C is the matrix of $a_{ij} - k_i^+ k_j^- / m$. A rough estimate is

$$s = \text{sign}\left(v_1(C)\right). \qquad (7)$$

Here $v_1(\cdot)$ refers to a nonzero eigenvector associated with the largest eigenvalue.

2) Louvain algorithm by Blondel et al. [46] – All nodes start as a singleton community. The nodes then attempt to escape from their own communities and relocate to their neighbors' communities guided by the modularity ascent. When the modularity reaches a stationary point, the formed communities are aggregated into a single node, and the edges are correspondingly aggregated by summation. Thus a higher-level network is formed. The aforementioned processes are repeated, until modularity can no longer increase.

Before network science was established as a field, clustering and partitioning algorithms were extensively discussed in the literature of graph theory [47]. Although not endowed with statistical interpretations, they can be complementary to community detection. For example, the Kernighan-Lin algorithm [48] addresses the problem of dividing a graph into two subgraphs so as to minimize the total cost of external connections. After the cut is initiated, pairs of nodes from the two subgraphs are selected to maximize the resulting incremental cost decrease, and then the adjustments of subgraphs are carried out in a greedy manner. This can be used as a routine to fine-tune the bisection determined by the spectral algorithm.

III. Decomposition for Distributed Control

A. Decentralized Control and Decomposition

The query for a decomposition in control problems dates back at least to the 1970s in the studies pertinent to the stability analysis of decentralized control [49], [50]. Based on the idea that in order to guarantee closed-loop stability, mutually impacting variables must be grouped together in decentralized control, graph-theoretic approaches using strongly connected components and block-triangular structures were emphasized [51]. These methods are restrictive on system structures and not suitable for highly integrated systems such as chemical plants that are generally well connected as a whole. In a different vein, in the process control domain, works on interaction analysis focused on multi-loop control configurations and aimed to develop interaction measures to capture the relations between inputs and outputs, so that input–output pairings are chosen in an optimal sense [52]. For example, the classical relative gain array (RGA) approach [53] considers the static gains between inputs and outputs, G_{ij} as the interaction measure, and then calculates a matrix

$$\text{RGA} = G \circ (G^{-1})^\top, \tag{8}$$

where $\circ$ stands for element-wise multiplication, to assess the quality of a diagonal pairing. Especially with the development of robust control theory after the 1980s, interaction analysis was combined with decentralized stability analysis [54], [55].

From a process design point-of-view, it is necessary to decide on the selection of input and output variables, as well as their pairing configuration, from the conceptual design stage. Studies that bore the name of "plantwide control" provided guidelines for such control structure selection [56]–[58]. For example, in Seider et al. [59, Chap. 20], multiple steps in sequence are suggested to determine the "subsystems":

1) Determine the control structure of energy flows to eliminate temperature runaway in reaction systems and decouple material and energy aspects;
2) Control the feed and production rates to ensure the overall material balance;
3) Control the variables closely related to quality, safety, environmental, and operational constraints;
4) Control the recycle and inventories;
5) Control individual units.

Such guidelines, apparently, were developed with a heuristic understanding of the process characteristics and relative priority of control objectives, using a hybridization of unit boundaries and mass/energy balance laws. Essentially, the "control structure selection" in plantwide control targets a decentralized, single-input-single-output (SISO) control architecture that is suitable for proportional-integral-differential (PID) controllers. Despite the rapid development of model predictive control throughout the 1980s and 1990s, the problem of performing a multi-echelon decomposition in optimal control using subsystem regulators and optimizers seemed to fade away.

B. Distributed MPC

The application of the decomposition principle in optimal control formulations has returned as a popular research topic after distributed MPC was proposed [60] and its stability properties were discussed [61]. The optimal control problem refers to the following dynamic optimization one, where the cost associated with the future trajectory is to be minimized under the discrete-time dynamical model and state or input constraints:

$$\begin{aligned} J(x_k) = \min_{\hat{x}_t, \hat{u}_t} \ & \sum_{t=k}^{k+N-1} \ell(\hat{x}_t, \hat{u}_t) + \ell_{\text{f}}(\hat{x}_{k+N}) \\ \text{s.t.} \ & \hat{x}_{t+1} = f(\hat{x}_t, \hat{u}_t), \ \ t = k, \ldots, k+N-1 \\ & \phi(\hat{x}_t, \hat{u}_t) \le 0, \ \ t = k, \ldots, k+N-1 \\ & \hat{x}_k = x_k. \end{aligned} \tag{9}$$

In the distributed MPC setting, the inputs and states are partitioned into n subsystems, i.e., $\hat{x}_t = (\hat{x}_t^{[1]}, \ldots, \hat{x}_t^{[n]})$ and $\hat{u}_t = (\hat{u}_t^{[1]}, \ldots, \hat{u}_t^{[n]})$. To solve the above problem across multiple optimizers, sequential and parallel iterations of control actions were proposed in the works of Christofides and coworkers [62], [63] as well as Rawlings and coworkers [64], [65]. The number of algorithms for distributed MPC has rapidly expanded since the early 2010s [66].

With the rapid development of distributed optimization (optimization algorithms based on operator splitting [67]) in the recent years, we are motivated to consider distributed MPC as a distributed optimization problem (instead of allowing any algorithm that in principle does not guarantee the optimum of the monolithic problem) [68]. For distributed

optimization, the MPC problem (9) can be reformulated as a n-block problem with linear equality constraints. Specifically, let $f^{[i]}$ be the components of the dynamical model f corresponding to $x^{[i]}$, and $s^{[i]}$ be the vector of variables that either (i) belong to the subsystem i and appear in $f^{[j]}$ for some $j \neq i$, or (ii) appear in $f^{[i]}$ but are not components of $x^{[i]}$ or $u^{[i]}$, i.e., $s^{[i]}$ represents the shared variables of subsystem i. Then we can rewrite

$$\hat{x}^{[i]}_{t+1} = f^{[i]}(\hat{x}^{[i]}_t, \hat{u}^{[i]}_t, \hat{s}^{[i]}_t), \;\; i = 1, \ldots, n. \tag{10}$$

Since all $s^{[i]}$ should contain some components of x and u, we can put all components of $s^{[i]}$, $i = 1, \ldots, n$ together as a vector of shared variables ξ_0, and write $s^{[i]} = B_i \xi_0$ for properly defined matrices B_i. By stacking all $x^{[i]}_t$, $u^{[i]}_t$ and $s^{[i]}_t$ as a vector ξ_i (standing for the variables to be solved by the i-th solver) and assuming the separability of the stage cost ℓ and terminal cost ℓ_{f}, we assert that the MPC problem can be rewritten as

$$\begin{aligned} \min_{\xi_0, \xi_1, \ldots, \xi_n} \;\; & \sum_{i=1}^{n} J_i(\xi_i) \\ \text{s.t.} \;\; & A_i \xi_i + B_i \xi_0 = 0, \;\; i = 1, \ldots, n \\ & \xi_i \in \Xi_i, \;\; i = 1, \ldots, n \end{aligned} \tag{11}$$

for some matrices A_i and sets Ξ_i that represent the dynamical model and control constraints, $i = 1, \ldots, n$.

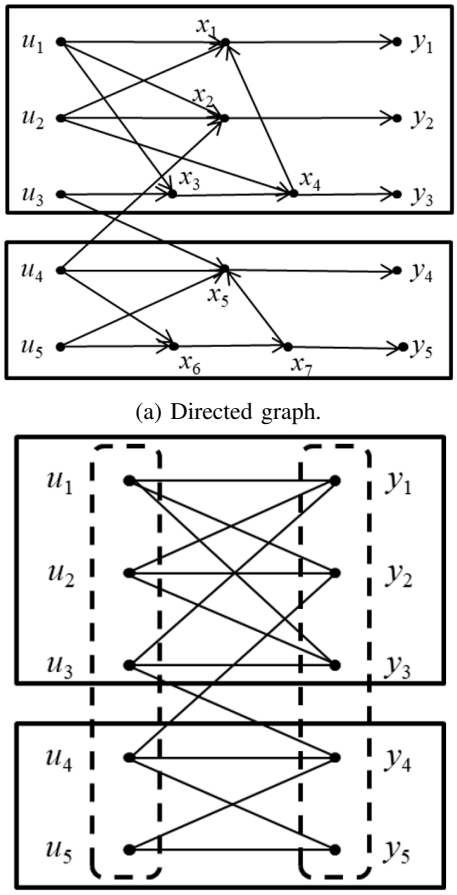
Such a linear equality constrained problem (11), if convex (i.e., in a linear MPC setting), is amenable to ADMM algorithms [69]. In the presence of nonconvex constraints (i.e., in nonlinear MPC), recent advances in distributed optimization have provided modified ADMM algorithms [70]. Our research has refined such modified ADMM and incorporated acceleration schemes for improved computational efficiency [71]. Furthermore, based on the reasoning of the closed-loop stability guarantee of nonlinear MPC, we pointed out that (11) may be terminated before converging to small enough tolerances, yet still preserving the closed-loop stability [72]. This algorithm uses the concept of *Lyapunov envelope*, which accounts for the effect of early termination on the closed-loop performance by penalizing the violations to inter-subsystem constraints, based on the assumption that all subsystems are incrementally dissipative, so that the propagation of errors will not devastate the descent of a robust upper bound of the Lyapunov function.

Although distributed MPC algorithms have been extensively studied, for a significant period after distributed MPC was proposed, the decomposition of large-scale systems in the sense of partitioning into several MPC subsystems, had not been addressed. As stated in [73]:

> "*There is no general framework for computing optimal decompositions for DMPC. ... Research in this direction should go hand-in-hand with the development of optimal communication strategies between the distributed controllers.*"

C. Network Representations for Decomposition

For a general framework of decomposing systems for control, our research has proposed a versatile range of network representations of dynamical systems, which flexibly capture the interactions among process variables under different characterizations, so that the latent block structures (communities) can be detected from the networks to generate subsystem configurations. These network representations include



(a) Directed graph.

(b) Input-output bipartite graph.

Fig. 2: Network representation for dynamical systems.

1) *Directed graph* (digraph), which captures the interactions among manipulated inputs, states, and controlled outputs [74]. That is, the set of nodes is $\mathcal{V} = \mathcal{U} \cup \mathcal{X} \cup \mathcal{Y}$, where the set of input nodes $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, state nodes $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, and output nodes $\mathcal{Y} = \{y_1, y_2, \ldots, y_p\}$. The set of edges is such that any edge is either from an input to a state, from a state to another state, or from a state to any output, i.e., $\mathcal{E} \subseteq (\mathcal{U} \times \mathcal{X}) \cup (\mathcal{X} \times \mathcal{X}) \cup (\mathcal{X} \times \mathcal{Y})$. For the model

$$\dot{x} = f(x) + g(x)u, \quad y = h(x), \tag{12}$$

(i) if $g_i j \neq 0$, then $(u_j, x_i) \in \mathcal{E}$; (ii) if $\partial f_i / \partial x_j \neq 0$, then $(x_j, x_i) \in \mathcal{E}$; (iii) if $\partial h_i / \partial x_j \neq 0$, then $(x_j, y_i) \in \mathcal{E}$. The directed graph can be weighted, e.g., by assigning the values of $|\partial f_i / \partial x_j|$, $|\partial h_i / \partial x_j|$, and $|\partial h_i / \partial x_j|$ at a reference steady state to the (u, x), (x, x) and (x, y) edges, respectively [75]. An alternative weighting can be defined by first calculating the shortest

path length between every pair of nodes, on which every edge is weighted inversely proportionally to the above coefficients' absolute values [76]. The weighted directed graph was shown to result in better control performance when there are large disparities among the edge weights (e.g., when the system contains a large recycle) [77].

2) *Input-output bipartite graph*, where the nodes stand for input and output variables ($\mathcal{V} = \mathcal{U} \cup \mathcal{Y}$) and every edge links an input and an output ($\mathcal{E} \subseteq \mathcal{U} \times \mathcal{Y}$). Any input u_i is connected to an output y_j if u_i has an effect on y_j, and in order to characterize the intensity of such an effect, edge weights can be assigned. In [78], the weight for the (u_j, y_i) edge, w_{ij}, is defined as the shortest path length from u_j to y_i on a weighted directed graph, so as to account for both the *relative degree* (topological closeness) and the sensitivity coefficients (response significance). In [79], the edge weight matrix $W = [w_{ij}]$ is defined according to a relative time-averaged gain array:

$$W = \max\left\{0, G(1/\tau) \circ \left(G(1/\tau)^{-1}\right)^{\top}\right\}, \qquad (13)$$

where $G(s)$ is the transfer function matrix, and τ is a time scale of interest. As such, W captures the short-time interactions between inputs and outputs.

3) *Variable-constraint graph*, which is constructed directly based on the optimization formulation of the optimal control problem at hand. In the variable-constraint graph, every node stands for an optimization variable or an algebraic constraint, and a variable node is connected to a constraint node if this node is involved in the constraint. In [80], this network representation was used to capture the MPC problem structure and was found to result in outstanding performance when it is used to decompose the optimization problem directly (instead of on the process variables). Naturally, variable-constraint graphs can serve as a basis for decomposing optimization problems in general [24], [81], which we discuss in the next section.

The directed and bipartite network representations for dynamical systems are shown in Fig. 2. Based on these representations, community detection can generate high-quality decompositions that outperform intuitive or heuristically determined ones, as shown in a series of studies by Pourkargar et al. [77], [82]–[84], where the algorithm of [63] was used for distributed MPC. In several other works that proposed new algorithms of distributed MPC, e.g., layered subsystems using hierarchical communication [85] and decomposition in the Karush-Kuhn-Tucker (KKT) conditions [86], directed graphs were adopted and community detection was carried out to generate subsystems. Different network representations can result in different performance metrics, which, on the other hand, largely depend on how distributed control is formulated and computed.

D. Community Detection and its Modifications

Without a deeper understanding of the fundamental relation between community structures and control, community detection in a network representation of large-scale systems seems at most a novel heuristic for decomposition. In fact, since community structure is a typical block structure existing in many biological networks [37], it is natural to hypothesize that the existence, or evolutionary emergence, of communities, plays a beneficiary role in their control. As pointed out in a number of studies [21], [22], [87], [88], community structures in a *modular network* allow the adoption of *modular controllers*, which largely promotes feedback sparsity (i.e., reduces complexity) while preserving the control performance. This justifies the use of community detection as a systematic framework of large-scale system decomposition.

In the above-mentioned works on control-oriented decomposition, Newman's spectral algorithm [45] and the Louvain (fast unfolding) algorithm [46] were used for community detection. The difference between the two algorithms lies in the path to search for the partition. The former algorithm recursively partitions a larger community into two smaller ones, starting from the entire network as a single community and terminated when further partition does not increase modularity. In contrast, the latter algorithm is initiated from singletons and recursively agglomerates smaller communities into larger ones. Compared to the spectral method, the Louvain algorithm is usually more efficient to find decompositions with a higher modularity value. We note that:

- In the context of distributed MPC, the subsystems are usually at most one order of magnitude smaller than the whole network but may contain hundreds of singletons, i.e., a top-down approach follows a shorter path to the solution.
- A bottom-up procedure as in Louvain algorithm starts from small increase in modularity while larger increases appear at later stages, i.e., the major steps are dependent on less important steps.
- In the Louvain algorithm, it is hard to rule out the generation of extremely small communities with very little gain in modularity.

Due to the above reasons, it appears appropriate to consider Newman's spectral algorithm as a more suitable approach in general for the purpose of decomposing control problems.

The network science algorithms, although highly efficient and theoretically well established, may not always guarantee meaningful and acceptable decomposition results. In an industrial implementation in the process industry [25], we proposed the following practical refinements of the spectral algorithm:

- The groups of variables that should not be separated are aggregated into single nodes before handled by decomposition. These groups often arise as "artifacts" of control engineering, e.g., (i) intermediate variables created for a nonlinear transformation, (ii) setpoints (SP), outputs (OP), and process variables (PV) inside

the same PID loop secondary to the MPC system.
- The bisectioning is held back if the modularity increase is not significant enough above a thresholding value. This helps to prevent the generation of extremely small communities.
- The internal connectedness of the communities are ensured by a depth-first search for connected components and recombination of the connected components inside communities.
- If the communities detected are imbalanced in size, the small communities are each merged with a matched larger community.

In addition, multiple modifications to the community detection approach have been proposed in the literature, so that the subsystems are not only meaningful from a network topology point of view, but also possess certain control-theoretic properties. In [89], the subsystems are adjusted until controllability conditions are satisfied. Yin and Liu [90] proposed to perform stabilizability and observability tests during community detection. Masooleh et al. proposed to use a multi-objective metaheuristic algorithm (whale optimization) instead of modularity maximization [91], based on which candidate decompositions can be ranked and selected for observability during the community detection procedure [92]. In Wang et al. [93], a mean gap metric is defined to measure the change in the dynamical system caused by decomposition, and is used as a check during community detection.

IV. Decomposition Solution Approaches for Optimization Problems

Most realistic optimization problems for decision making in process systems, such as those encountered in model predictive control as well as supply chain management, process operations and scheduling, and process design, are inherently nonconvex (mixed integer) nonlinear programs. While the optimization community has developed many impressive advances supporting deterministic global optimization of these problems, including state-of-the-art off the self solvers such as BARON [94], MAiNGO [95], ANTIGONE [96], DICOPT [97], and others [98], these approaches are not always easily scalable to problems of practical size due to the inherent NP-hardness of nonconvex, mixed integer problems. In many cases, utilizing a decomposition solution approach, whereby smaller subproblems are solved iteratively and coordinated to arrive at the solution to the original large optimization problem, can be faster than applying an off-the-shelf solver monolithically [99]–[102]. In this section, we provide a brief tutorial review of three of the most commonly used decomposition solution approaches for solving optimization problems that are particularly well suited for exploiting network structure in the underlying optimization problem.

The following subsections consist of various generally written optimization formulations. To avoid confusion, we provide the following preliminary definitions for notation used consistently over the multiple subsections. The symbols x, y, and z denote vectors of decision variables in the optimization problem. Variables with a subscript, i.e. x_1, denote a scalar element of the corresponding vector. The symbols f, g, and h, sometimes with subscripts, denote arbitrary nonlinear functions of these decision variables. These functions may be vector or scalar valued, depending on the context: typically, functions in the objective will be scalar valued, while constraint functions can be vector valued functions of arbitrary size. Most other lowercase symbols refer to vectors of known constant parameters, i.e. the vector c might appear multiplied by x in an objective, denoting some sort of cost. Upper case symbols typically refer to matrices of known constant parameters. Definitions of additional symbols (i.e. Greek letters) and exceptions to these general guidelines will be explicitly noted when such symbols appear in the text.

A. Graph Representation of Optimization Problems

As we will see in this section, decomposition solution methods for solving optimization problems have been developed and in use for over 60 years. Traditionally, identifying if a particular problem was well suited for decomposition was reliant on the intuition of an optimization expert. However, recent advances in network theory, such as the methods discussed in section II, enable a systematic approach for identifying exploitable structure within an optimization problem of interest. To enable the use of network theory, a graph representation of the optimization problem is required. Recent work has demonstrated that optimization problems can naturally be represented as graphs, enabling easy visualization and structure detection of the problem [24], [103].

While one can build any number of graphs or hypergraphs corresponding to a problem, for the purposes of identifying structure amenable to decomposition, the *variable-constraint graph* is a good starting point. In this graph, a bipartite set of nodes corresponding to decision variables and constraints, respectively, in the optimization problem is constructed. A variable node is linked with an edge to a constraint node if and only if the variable appears within the respective constraint function. Letting $f(x)$ be the vector of constraint functions of decision variables x and $\hat{A}$ be the adjacency matrix of the variable-constraint graph, the following mathematical expression ensures this relationship:

$$\hat{A}_{ij} = \begin{cases} 0, & \frac{\partial f_i}{\partial x_j} = 0 \\ 1, & \text{otherwise} \end{cases} \tag{14}$$

Note that for problems with integer variables, the partial derivative resulting from the continuous relaxation is used. The variable-constraint graph is useful for initial visualization of the problem structure, but it alone is not the best suited for identifying structure in the optimization problem suitable for a decomposition solution algorithm. For this, the unipartite variable or constraint graphs should be used. These are obtained from simple right or left projections of the variable-constraint graph. Letting $\hat{X}$ and $\hat{F}$ be the adjacency matrices of the variable and constraint graphs, respectively, these are obtained from $\hat{A}$ using $\hat{X} = \hat{A}^T\hat{A}$ and $\hat{F} = \hat{A}\hat{A}^T$. Using this definition, the constraint graph is one

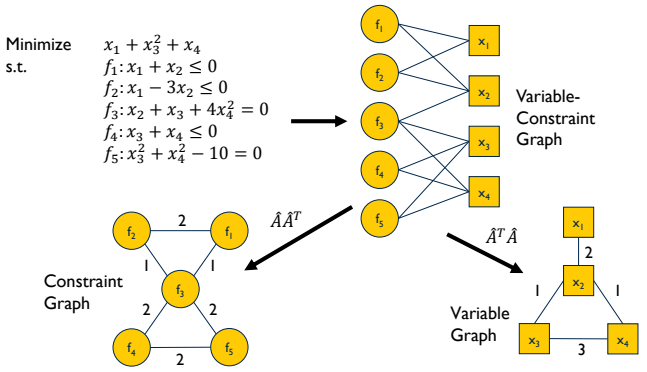


Fig. 3: Example generation of variable-constraint, constraint, and variable graphs from a simple optimization problem.

where nodes correspond to constraints and weighted edges correspond to the number of variables a pair of constraints share. Analogously, the variable graph is one where nodes correspond to variables and weighted edges correspond to the number of constraints that a pair of variables jointly appear in. A simple example of variable-constraint, constraint, and variable graphs are shown in Fig. 3. As we will see in the following subsections, community or core-periphery structure in the variable or constraint graphs directly corresponds with structure amenable for exploitation by three of the most well known decomposition solution algorithms.

B. Column Generation

Perhaps one of the earliest applications of decomposition to optimization problems is the classic 1960 paper of Dantzig and Wolfe [104]. This work proposed a met that makes use of the observation that many large scale linear programs (LPs) of practical interest can be formulated in a way such that the optimal solution was inherently sparse; that is, even though the number of variables in a problem could be very large, the number of variables with *nonzero* values in the optimal solution is typically significantly smaller. This observation motivated an iterative approach whereby a "restricted master problem" (RMP) is solved which only allows varying a subset of variables from the original problem to vary from zero. Based on the results of the RMP, a subproblem is then solved to determine, which, if any, variables not considered in the RMP could potentially improve the objective value. These variables are then added to the RMP, and the process repeats. Since in the linear programming literature, variables correspond to columns in the constraint matrix and are sometimes called columns, this approach is often referred to as "column generation."

To observe how the column generation algorithm is typically applied, consider the following linear program:

$$\min_{x,y,z} \quad c_x^T x + c_y^T y + c_z^T z \tag{15a}$$
$$\text{s.t.} \quad A_x x + A_y y \leq b \tag{15b}$$
$$D_y y + D_z z \leq e \tag{15c}$$

In column generation, constraints (15b) are typically referred to as complicating constraints that, if removed, would render the problem significantly easier to solve, usually due to problem structure. Moreover, the number of variables y appearing in both sets of constraints can be very large. Nonetheless, for fully bounded linear programs (i.e., those where no variable may feasibly take a value of $\pm\infty$), y can be represented as a convex combination of all vertices of a simplex defined by the problem's constraints. Moreover, the number of vertices to consider will be finite (although potentially very large). Let the set of these points be $\mathcal{Y}$, the set $\mathcal{K} = \{1, ..., |\mathcal{Y}|\}$, y_k refer to a point in $\mathcal{Y}$, and f_k be the "column cost" of y_k, such that $f_k = \min_z \{c_y^T y_k + c_z^T z : D_y y_k + D_z z \leq e\}$. Then, the following "master problem" is equivalent to problem (15):

$$\min_{x,\lambda} \quad c_x^T x + \sum_{k\in\mathcal{K}} f_k \lambda_k \tag{16a}$$
$$\text{s.t.} \quad c_x^T x + c_y^T \left(\sum_{k\in\mathcal{K}} y_k \lambda_k \right) \tag{16b}$$
$$\sum_{k\in\mathcal{K}} \lambda_k = 1 \tag{16c}$$
$$0 \leq \lambda_k \leq 1 \quad \forall\, k \in \mathcal{K} \tag{16d}$$

For most problems of practical interest, the cardinality of $\mathcal{K}$ is much too large for the master problem (16) to be helpful in solving the problem. However, in almost all practical cases, all but a very small number of λ variables will be zero in the optimal solution. As such, a restricted master problem is formulated by considering only a subset of points on the y-simplex, $\hat{\mathcal{Y}} \subset \mathcal{Y}$, sometimes referred to as the "basis set."

Since the RMP is a restriction of the original problem, the objective value of its solution is an upper bound on the original problem's objective value. To refine this bound, new columns must be generated that have potential for improving the solution. To do this, dual variables π, corresponding to constraints (16b), and μ, corresponding to constraint (16c), from the RMP solution are passed to the following "subproblem":

$$\min_{y,z} \quad (c_y^T - \pi^T A_y) y + c_z^T z - \mu \tag{17a}$$
$$\text{s.t.} \quad D_y y + D_z z \leq e \tag{17b}$$

When solving this problem, the objective value is often referred to as the "reduced cost", as it represents the best possible improvement in the current upper bound by including the new column y^*, the y solution to the subproblem. If the reduced cost is negative, y^* is added to the basis set and the column generation algorithm repeats with a new solution of the RMP. If the reduced cost is nonnegative, then no new columns can improve the incumbent solution, and the column generation algorithm is converged to the optimal solution.

A critical characteristic for using this approach that we have overlooked so far is that the subproblem (17) should be significantly easier to solve than the original, monolithic problem (15). This requirement is nontrivial, and is most commonly satisfied with a subproblem that displays "decomposable structure"; that is, it can be broken into

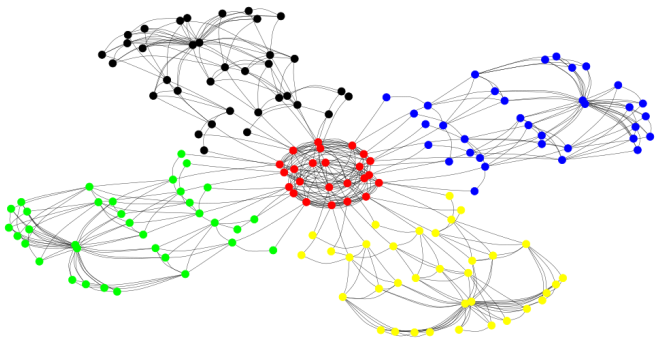


Fig. 4: Example of a constraint graph from a dynamic facility location problem well suited for solving via column generation [105]. Note the core-periphery structure, such that when the core nodes (red) are removed, the peripheral communities are completely disconnected.

multiple small optimization problems that, when solved independently, give the exact solution to the original larger optimization problem. When column generation is applied to a subproblem with decomposable structure, the approach is often referred to as the "Dantzig-Wolfe decomposition." Connecting this to the graph structure of the original optimization problem, we note that problems with any core-periphery structure in the constraint graph are candidates for solution via column generation. The best candidates are those that, when the core nodes are removed, have disconnected communities (Fig. 4), as this indicates that the subproblem (corresponding to the periphery) will have decomposable structure.

While originally developed for linear programs (LPs), the concept of column generation has been extended to work for more challenging classifications of problems. In particular, column generation can be quite powerful for solving integer programs (IPs) and mixed integer linear programs (MILPs), as first demonstrated by the work of Desrosiers et al. [106]. In this approach, the continuous relaxation of the original problem is solved via column generation. If the solution is non-integral in any integer variables, a branch-and-bound scheme is applied which generates additional constraints to the RMP to promote integrality of integer variables. Then, each branching node in the branch-and-bound tree is initialized with feasible columns in the basis set of its parent node and solved using column generation. The process repeats, using standard branch-and-bound principles for navigating the branching tree and pruning nodes. A IP/MILP solution approach combining branch-and-bound with column generation is referred to as "branch-and-price." Branch-and-price has been widely used to much success for many problems in the operations research community, including vehicle routing [107], [108], facility location [105], [109], and capacity planning problems [110], [111]. For a more complete review of the method, we refer the reader to [112].

Perhaps more relevant to chemical process systems, it has also recently been shown that branch-and-price can be quite useful for solving certain mixed integer nonlinear programs (MINLPs) [113], known to be among the most difficult class of problems to solve and occurring naturally for many optimal process design and process operations problems. For these problems, it has been shown that when the complicating constraints (15b) are all linear and the complicating variables y are all integer variables, branch-and-price can solve the MINLP to global optimality assuming a global MINLP optimizer is used for solving the subproblem(s). This stands out compared to most decomposition solution approaches, which when applied to MINLPs give no guarantees that the optimality gap will be closed. This approach has been demonstrated to be quite powerful when applied to design, planning, and scheduling problems considering many time periods and/or many uncertain scenarios, as the subproblem typically can be decomposed in time or by scenario.

C. Benders Decomposition

The second major type of decomposition also dates to the early 1960's and comes from the pioneering work of J.F. Benders [114]. The eponymous Benders decomposition utilizes an approach that will seem analogous to the column generation algorithm presented in the prior subsection, but instead of only considering a subset of variables in a master problem, we consider only a subset of constraints in the master problem, iteratively generating new "rows", or "Benders cuts" as the decomposition solution algorithm progresses. This work was originally applied to "mixed variables" programming problems, as in the following example:

$$\min_{x,y} \quad c^T x + f(y) \tag{18a}$$
$$\text{s.t.} \quad Ax + g(y) \leq b \tag{18b}$$
$$x \geq 0 \tag{18c}$$
$$y \in \mathbb{Z} \tag{18d}$$

In this problem, y variables are considered to be the complicating variables such that, when these variables are fixed, the problem becomes significantly easier to solve. In this formulation, it is clear that when y is fixed at $\bar{y}$, the problem becomes an LP in x. For LPs, which display strong duality, an equivalent formulation is:

$$\max_{\lambda} \quad (b - g(\bar{y}))^T \lambda + f(\bar{y}) \tag{19a}$$
$$\text{s.t.} \quad A^T \lambda \leq c \tag{19b}$$
$$\lambda \geq 0, \tag{19c}$$

where λ are the dual variables of constraints (18b). The above problem is often referred to as the subproblem. The feasible space of the subproblem is defined by a finite set of extreme points $\bar{u} \in \mathcal{P}$, i.e. vertices of the feasible region, and extreme rays $\tilde{u} \in \mathcal{R}$, i.e. linearly independent directions in which the feasible region is unbounded. Duality theory states that these extreme points and rays can be used to define the following master problem equivalent to the original problem (18):

$$\min_{y,z} \quad z \tag{20a}$$
$$\text{s.t.} \quad 0 \geq (b - g(y))^T \tilde{u} \quad \forall\, \tilde{u} \in \mathcal{R} \tag{20b}$$

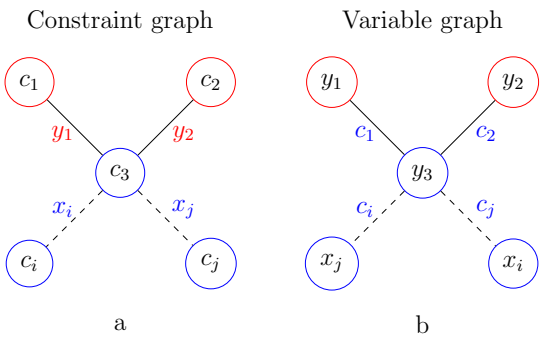


Fig. 5: Block structure in the variable and constraint graph that can be used for Benders decomposition. (a) The nodes in red belong in one block and the nodes in blue in the second block and the variables in red color are the complicating variables. (b) The variables in red belong in the one block and are assigned in the master problem whereas the nodes in blue and the associated edges are assigned in the subproblem.

$$z - f(y) \geq (b - g(y))^T \bar{u} \quad \forall\, \bar{u} \in \mathcal{P} \tag{20c}$$

Analogous to column generation, the total number of constraints to consider scales poorly with the dimensionality of the original problem and is typically too large for the master problem to be of any utility, since the computation of all the cuts is equivalent to solving the original problem. As such, it is reasonable to start by solving a relaxed master problem with no or only a small number of constraints (20b) and (20c). Next, the values of the y variables in the solution of the master problem are set as $\bar{y}$ in the subproblem, which is then solved.

For given $\bar{y}$ the subproblem is either unbounded, i.e., the objective diverges to an infinite value, or feasible. In the first case, an extreme ray which proves that the solution is unbounded $\tilde{u}$ is taken from the subproblem and used to generate a new constraint of the form (20b) in the relaxed master problem. Constraints of this nature are often referred to as "feasibility cuts," since the unbounded nature of the subproblem implies that $\bar{y}$ is not a feasible solution to the original problem. In the second case, the subproblem returns a feasible finite solution $\bar{u}$, which is used to generate a new constraint of the form (20c) in the relaxed master problem. Constraints of this nature are often referred to as "optimality cuts"; together with the feasibility cuts they inform the master problem about the effect of the complicating variables on the subproblem and guide the algorithm towards the optimal solution.

Because the relaxed master problem is inherently a relaxation of the original problem, its solution provides a lower bound of the true optimum. Similarly, since the subproblem is an equivalent formulation of the original problem with y variables fixed, it is a restriction of the original problem whose objective value corresponds to an upper bound of the optimum. As such, the Benders decomposition iterates between solving the relaxed master problem and generating new cuts from the subproblem until both converge on the same objective value.

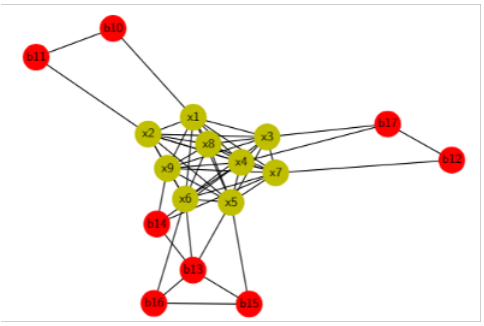


Fig. 6: Inferred core-periphery structure of the variable graph of the synthes3 benchmark problem which can be used as the basis for Generalized Benders Decomposition. The red nodes in the periphery are the binary variables which are assigned in the master problem whereas the yellow nodes in the core are the continuous variables assigned in the subproblem.

Given the iterative and sequential solution of the master problem and the subproblem, Benders decomposition exploits the underlying hierarchical structure of an optimization problem. This hierarchy is usually manifested as a core-periphery, core-community or multi-core community structure in the graph representation of an optimization problem. An important class of problems where these structures arise is enterprise-wide optimization problems which consider simultaneously decisions across scales, i.e., planning, scheduling, and control. For such problems, the inference of the parameters of a nested stochastic block model can reveal the structure of the problem across multiple temporal and spatial scales. The estimated structure can be further used as the basis for the application of Benders and nested Benders decomposition architectures [81], [115], [116]. In such cases, one can exploit the hierarchical structure of the variable or constraint graph. Specifically, in the variable graph, the block of nodes that contain the integer variables can be assigned in the master problem and the other nodes are assigned in the subproblems as presented in Fig. 5 (b) and 6. In the constraint graph, the set of nodes that have edges that capture integer variables are assigned in the master problem and the other nodes and edges are assigned in the subproblem (see Fig. 5 (a)).

While originally developed for problems of the form (18), Benders decomposition was later extended to work for problems with nonlinear subproblems. This approach, first developed in the work of Geoffrion in the 1970's [117], has been given the name "generalized Benders decomposition". For brevity, we will not relist the full formulations for this case; however, the algorithm follows the same approach. Namely, a subproblem with complicating variables is solved, and its dual solution is used to generate new feasibility and optimality cuts which are added in the master problem. For the cases where the subproblem is convex, it can be shown

that the generalized Benders decomposition will converge on the globally optimal solution. However, in the nonconvex case, there are no guarantees that the solution obtained will be even locally optimal [118], [119].

Numerous theoretical and technical advances in (generalized) Benders decomposition have been focused on acceleration and further generalization of the method, considering approaches for alternative master problem formulations [120], [121], advanced cut generation and management [122]–[124], integer variables in the subproblem [125], [126], multicut implementation [127], [128], nonconvex subproblems [129], or uncertain parameters in the subproblem [130]. For problems which need to be solved repeatedly with different values of problem parameters, such as those encountered in process operations and control, it is possible and often beneficial to apply machine learning tools to learn a set of cut constraints for initializing the master problem [131] or approximating the solution of the subproblem [132]. For a more complete review of the method, we refer the reader to [133]. Benders decomposition has been applied widely across the process systems literature, with particular application to problems of a multi-scale nature, such as those combining scheduling with process control [9], [115], [116], [134], supply chain planning [135], and process design [136].

D. Lagrangian Decomposition

A third decomposition solution approach is referred to as the Lagrangian decomposition. This approach was developed a bit later than the first two, but has been much more widely applied in decision making problems for chemical process systems, including in the process control community. This approach has its roots in the classical Lagrangian relaxation, an approach which converts a constrained optimization problem to an unconstrained problem via the introduction of "Lagrange multipliers," which are also referred to as "dual variables." Pioneering work by Guignard and Kim [137] demonstrated that this idea can be particularly useful for problems with complicating constraints. As an example of such a problem, consider the following nonlinear program (NLP):

$$\min_{x,y} \quad f_1(x) + f_2(y) \tag{21a}$$
$$\text{s.t.} \quad g_1(x) \leq 0 \tag{21b}$$
$$g_2(y) \leq 0 \tag{21c}$$
$$h_1(x) + h_2(y) \leq 0 \tag{21d}$$

In this problem, constraints (21d) are referred to as complicating constraints, as if they were removed from the problem, it would have purely decomposable structure with 2 independent subproblems. In the Lagrangian decomposition, the complicating constraints are not removed, but instead "dualized", or placed in the objective, by applying the Lagrangian relaxation:

$$\min_{x,y} \quad f_1(x) + \lambda^T h_1(x) + f_2(y) + \lambda^T h_2(y) \tag{22a}$$
$$\text{s.t.} \quad g_1(x) \leq 0 \tag{22b}$$
$$g_2(y) \leq 0 \tag{22c}$$

Clearly, this problem has a decomposable structure: two independent optimization problems can be generated which determine the optimal values of x and y, respectively. Moreover, when λ is chosen to be nonnegative, problem (22) is a relaxation of problem (21), which means that its solution will give an objective value that is a lower bound on the original problem objective. For an optimization problem that satisfies all of the conditions for strong duality, one can always iterate to a value of λ that gives a solution that is feasible to the original problem, thus closing the optimality gap and guaranteeing a globally optimal solution. However, in the general case, the Lagrangian decomposition is only a heuristic approach that only gives a lower bound on the solution. Depending on the problem, using this approach to find a good feasible solution may be tricky.

The Lagrangian decomposition is particularly powerful as it can be used not only for problems with complicating constraints, but also for those with complicating variables, through a very straightforward reformulation of the problem. Consider, for example, the following NLP with complicating variables y that, if removed or fixed, would result in a problem with decomposable structure:

$$\min_{x,y,z} \quad f_1(x,y) + f_2(y,z) \tag{23a}$$
$$\text{s.t.} \quad g(x,y) \leq 0 \tag{23b}$$
$$h(y,z) \leq 0 \tag{23c}$$

This problem can be reformulated to one with complicating constraints by introducing an additional set of "copy variables", $\bar{y}$:

$$\min_{x,y,\bar{y},z} \quad f_1(x,y) + f_2(\bar{y},z) \tag{24a}$$
$$\text{s.t.} \quad g(x,y) \leq 0 \tag{24b}$$
$$h(\bar{y},z) \leq 0 \tag{24c}$$
$$y = \bar{y} \tag{24d}$$

When formulated in this way, constraints (24d) act as complicating constraints and are treated as such in the Lagrangian decomposition. As such, good candidate problems for Lagrangian decomposition are those with community structure in either the variable or the constraint graph (Fig 7), as removal of the edges connecting communities by relaxation leads to independent subproblems. Since each community in this graph will correspond to a Lagrangian subproblem, finding a highly modular community structure ensures a statistically minimal number of complicating constraints (if using the variable graph) or complicating variables (if using the constraint graph), which should result in faster convergence of the decomposition algorithm.

Critical to the convergence of Lagrangian decomposition algorithm is the update of the dual variables λ. The optimal value of λ is the one that satisfies the dual problem defined by (22), $\max_{\lambda \geq 0} (\inf_{x,y \in \mathcal{D}} (22a))$, where $\mathcal{D}$ is the set of solutions for x and y that satisfy (22b) and (22c). Typically, a steepest descent algorithm is applied for dual updates, such

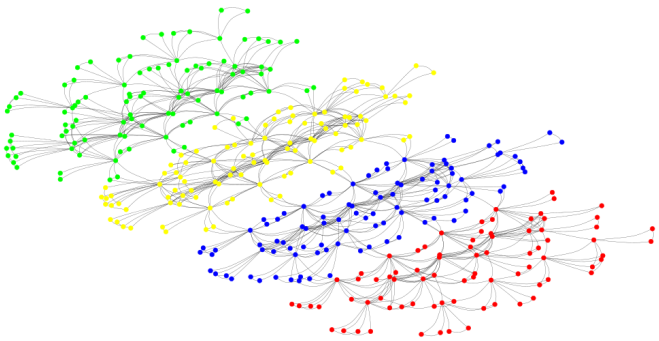


Fig. 7: Example of a constraint graph of a coordinated network scheduling problem well suited for solving via (augmented) Lagrangian decomposition due to its community structure [138]. Similar community structure in the variable graph also provides a structure which can be exploited by Lagrangian decomposition.

that the dual variable at iteration $k+1$ is defined by the value of the dual variables and primal solution at iteration k:

$$\lambda^{(k+1)} = \max(0, \lambda^{(k)} + \delta_k(h_1(x^{(k)}) + h_2(y^{(k)})) \quad (25)$$

where δ_k is a positive, user defined step size that can change (usually, decreasing) with iteration k. This approach makes intuitive sense: when $h_1 + h_2 > 0$, the constraint is being violated and a higher Lagrangian "penalty" should be applied, when $h_1 + h_2 < 0$ but $\lambda > 0$, complicating constraints are satisfied but it may be possible to achieve a better feasible solution, and when $h_1+h_2 = 0$ or $h_1+h_2 < 0$ and $\lambda = 0$, the Karush-Kuhn-Tucker (KKT) conditions have been satisfied, indicating a feasible stationary point has been obtained and the algorithm has converged.

Unfortunately, Lagrangian decomposition sometimes displays relatively slow convergence. An alternative approach that can often accelerate solutions is to augment the Lagrangian relaxation with a quadratic penalty term, which ideally will more quickly drive the algorithm towards a feasible solution [139]. This approach, known as the "augmented Lagrangian decomposition", is particularly well suited when subproblems are coupled with equality constraints, such as in (26). In this case, the relaxed problem would be:

$$\min_{x,y,\bar{y},z} \quad f_1(x,y) + f_2(\bar{y},z) + \lambda^T(y-\bar{y}) + \rho\|y-\bar{y}\|_2^2 \quad (26a)$$

$$\text{s.t.} \quad g(x,y) \leq 0 \quad (26b)$$

$$h(\bar{y},z) \leq 0 \quad (26c)$$

Where ρ is a user-defined quadratic penalty coefficient for the dualized equality constraint. A popular approach to solving this problem is the ADMM, adapted from the method of multipliers developed in the late 60's [140] and popularized more recently by the work of Stephen Boyd [16]. This approach iteratively solves the independent subproblems and follows them with a dual update based on the quadratic penalty coefficient ρ as follows: first, one of the subproblems is solved based on the current value of the dual variables λ, and primal variables from the previous iteration:

$$\min_{x,y} \quad f_1(x,y) + (\lambda^{(k)})^T y + \frac{\rho}{2}\|y-\bar{y}^{(k-1)}\|_2^2 \quad (27a)$$

$$\text{s.t.} \quad g(x,y) \leq 0 \quad (27b)$$

From this first subproblem solve, current iteration values of y, $y^{(k)}$, are obtained, which are used to solve a subsequent subproblem:

$$\min_{\bar{y},z} \quad f_2(\bar{y},z) - (\lambda^{(k)})^T\bar{y} + \frac{\rho}{2}\|\bar{y}-y^{(k)}\|_2^2 \quad (28a)$$

$$\text{s.t.} \quad h(\bar{y},z) \leq 0 \quad (28b)$$

Once all subproblems have been solved in the iteration, dual variables are updated in a manner similar to the steepest decent approach given in (29), but using the parameter ρ as the step size:

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho(y^{(k)} - \bar{y}^{(k)}) \quad (29)$$

Various extensions to ADMM have been proposed which aim to accelerate convergence or guarantee optimality for a broader class of problems. One such approach is the ELLADA method [71], which makes use of approximate, rather than exact, subproblem updates [141], the Anderson acceleration [142], [143], and a multi-layer algorithm with auxiliary slack variables [70].

Augmented Lagrangian decomposition strategies are particularly well suited for solving distributed, multi-agent optimization problems that result from, for example, systems with multiple decision making entities that seek to cooperate but are self-interested, physically separated, or limited in their ability to share information [138], [144]. In these types of applications, it is often not possible to choose how the original monolithic problem is decomposed based on its structure; instead, subproblem structure is enforced by the inherent physical separation of the distributed decision making entities. These methods have found extensive use in the distributed MPC literature, with applications in chemical process systems [72], [145], power systems [146], [147], energy management in buildings [148], and autonomous vehicles [149], [150].

V. Perspectives for Future Work

The following subsections highlight some open questions and directions for future research in structured approaches to control and optimization.

A. Performance Considerations

While there is evidence that the existence and detection of block structures, e.g., communities, is strongly correlated with control or optimization performance, the detection of structures, e.g., by maximizing a modularity index, is not directly driven by a performance measure. On the other hand, strictly optimizing the decomposition with respect to the resulting performance is computationally intractable, although some expensive meta-heuristics such as genetic algorithms may be applied [151]. Depending on the specific application, it may be a plausible approach to first correlate the subsystems' features with the corresponding performance, perhaps

making use of recent advances in machine learning, and then decompose according to empirical correlations.

For example, we may consider the decomposition of a continuous-variable optimization problem (e.g., for Lagrangian decomposition algorithm) that needs to be optimal with respect to computational time. Assuming that (i) the single-iteration computational time for solving the i-th subproblem with n_i variables and m_i constraints is proportional to $n_i^\alpha m_i^\beta$ (for some $\alpha, \beta > 0$), (ii) the computation in all the subsystems are perfectly parallelized, so that the single-iteration total computational time is proportional to $\max_i n_i^\alpha m_i^\beta$, and (iii) the number of iterations needed for the coordination is proportional to $(\sum_{i,j} n_{ij})^\gamma$, where n_{ij} is the number of overlapping variables between subsystems i and j and $\gamma > 0$, then we may decompose the system according to

$$\max \left(\max_i n_i^\alpha m_i^\beta \right) \left(\sum_{i,j} n_{ij} \right)^\gamma \tag{30}$$

instead of a modularity index. Yet, the above formulation may be too simplistic to capture all the factors that may affect computational time. It is therefore an important issue to extract the relevant features and develop an appropriate performance correlation.

B. Constrained Structure Detection

It is often the case that the decomposition needs to satisfy some constraints on its configuration. A typical case is the decomposition of mixed-integer nonlinear programming problems for Benders' algorithm, where (i) all the integer variables (as the "complicating" variables) and nonconvex constraints should be preferably contained in the master problem, (ii) the sub-problem should contain only continuous variables and convex constraints, thus forming a convex programming problem, whose size should be must larger than the master problem. Similarly, for nonlinear branch-and-price, global convergence is only guaranteed when all complicating variables are integer variables and the master problem contains only linear constraints. As another example, for large problems with a small number of nonlinear constraints, it may be desirable to generate a nonlinear subproblem that is as small as possible to speed up solution times. Finally, domain knowledge, human instruction, or a physical separation between system components may inform the decomposition. For example, stochastic and dynamic programming problems should have repetitive patterns that reflect the same system under different uncertain parameter values or at discretized time instants in a horizon, while in distributed control, it may be acceptable to require that the same type of units be treated in a homogeneous way in the decomposition. In all of these cases, it can be beneficial to embed constraints to the community detection or stochastic block-modeling algorithms to restrict the detected structure to one that meets some physical or computational requirements.

Essentially, constrained structure detection becomes a combinatorial optimization problem under constraints, e.g., the maximization of modularity under linear inequalities on the allocation vector:

$$\begin{aligned} &\max \ Q(g) \\ &\text{s.t.} \ Ag \leq b \end{aligned} \tag{31}$$

As the unconstrained version can be amenable to efficient routines such as hierarchical agglomeration or division, we envision that the constrained problem can be converted to an unconstrained problem with a dualization approach, e.g., by defining the augmented Lagrangian

$$\bar{Q}(g) = Q(g) + \lambda^\top (Ag - b) + \frac{\rho}{2} \|Ag - b\|^2, \tag{32}$$

and adjusting the values of $\lambda \geq 0$ and $\rho > 0$ respectively. Alternatively, one could consider the constraints as complicating, as when they are removed, the problem becomes significantly easier to solve. When viewed through this lens, a column generation approach could be applied on the structure detection algorithm itself, whereby the column generating subproblem is simply an unconstrained structure detection problem (with added dual terms in the objective to calculate reduced cost), while the master problem picks among the generated columns (different partitions of the network, in this case), calculating a dual cost for satisfying any constraints.

C. Hybrid and Nested Decomposition Structures

The decomposition methods described above exploit either the hierarchical or the distributed structure of an optimization problem. However, another approach is to exploit both structures simultaneously. Cross decomposition [152] is an algorithm which implements simultaneously Benders decomposition and Lagrangian relaxation for the solution of a mixed integer optimization problem. Conceptually, this algorithm can be applied in cases where an optimization problem has structure both in the constraint and variable graph. Alternatively, such a approach could be beneficial for a core-periphery structure similar to that in Fig. 4, but where the periphery forms weakly connected but not fully disconnected communities.

In addition to cross decompositions, one could envision various nested decomposition strategies that exploit multiple levels of hierarchical structure in the underlying network. For example, consider an optimization problem that, at the problem-wide level, has a community structure, but once the communities are separated into their own subproblems, they may each have additional community or core-periphery structure that could be further exploited. Here, one could envision an outer-level Lagrangian decomposition, with Lagrangian subproblems solved by, for example, column generation or a further Lagrangian decomposition. For such nested hierarchical structures, open questions remain such as how "deep" into the hierarchy is it still effective to detect and exploit structure, as well as how information can be most effectively shared within the hierarchy of decomposition algorithms.

D. When is a Decomposition Needed?

Although decomposition-based solution methods have been widely applied for large-scale problems, for some cases, such as convex MINLPs, they can be efficient even for small-scale optimization problems [153]. Similarly, while Dantzig-Wolfe decomposition has been widely applied for the solution of MILP problems, its effectiveness over other solution methods such as branch and cut [154] is not known *a priori*. Indeed, recent work has demonstrated that for one class of optimization problems with the same structure but varying parameters or numbers of variables and constraints, a single solution approach between decomposing the original problem or solving the original problem monolithically is not uniformly superior with respect to time required to find a globally optimal solution [105]. The problem of whether to use a decomposition over a monolithic solution approach can be posed as an algorithm selection problem [154]–[156] where the goal is to find the algorithm that solves an optimization problem in the minimum solution time.

VI. Conclusions

In this tutorial paper, we focus on decomposition as an important principle of resolving large-scale control and optimization problems. In the context of control problems (especially that of model predictive control) and optimization problems, we reviewed how decision making can be achieved on the basis of subsystems under a decomposition of the monolithic system (problem), and how such a desirable decomposition can be determined. In particular, we highlight the detection of latent block structures in networks as a general framework of systematically finding statistically significant mesoscale interaction patterns, thus generating subsystems for distributed control and optimization. We have also discussed promising future directions, especially on answering the open questions regarding the characterization of the performance of decomposition configurations, incorporation of constraints on decomposed subsystems, hybridization of prototypical block structures, and comprehension of the necessity of decomposition. Given its wide applicability and theoretical profundity, we believe that this broad area of decomposition and decomposition-based problem solving is worth deeper exploration.

References

[1] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, no. 6804, pp. 651–654, 2000.

[2] G. A. Pagani and M. Aiello, "The power grid as a complex network: A survey," *Phys. A: Stat. Mech. Appl.*, vol. 392, no. 11, pp. 2688–2700, 2013.

[3] A. A. Ganin, M. Kitsak, D. Marchese, J. M. Keisler, T. Seager, and I. Linkov, "Resilience and efficiency in transportation networks," *Sci. Adv.*, vol. 3, no. 12, p. e1701079, 2017.

[4] T. A. B. Snijders, "The statistical evaluation of social network dynamics," *Sociol. Methodol.*, vol. 31, no. 1, pp. 361–395, 2001.

[5] M. Baldea and P. Daoutidis, *Dynamics and nonlinear control of integrated process systems*. Cambridge University Press, 2012.

[6] D. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific, 2012.

[7] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on stochastic programming: Modeling and theory*, 3rd ed. SIAM, 2021.

[8] M. Morari, Y. Arkun, and G. Stephanopoulos, "Studies in the synthesis of control structures for chemical processes: Part i: Formulation of the problem. process decomposition and the classification of the control tasks. analysis of the optimizing control structures," *AIChE J.*, vol. 26, no. 2, pp. 220–232, 1980.

[9] Y. Chu and F. You, "Integrated scheduling and dynamic optimization of complex batch processes with general network structure using a generalized Benders decomposition approach," *Ind. Eng. Chem. Res.*, vol. 52, no. 23, pp. 7867–7885, 2013.

[10] D. Mora-Mariano, M. A. Gutiérrez-Limón, and A. Flores-Tlacuahuac, "A Lagrangean decomposition optimization approach for long-term planning, scheduling and control," *Comput. Chem. Eng.*, vol. 135, p. 106713, 2020.

[11] I. Mitrai and P. Daoutidis, "Decomposition of integrated scheduling and dynamic optimization problems using community detection," *J. Process Control*, vol. 90, pp. 63–74, 2020.

[12] I. E. Grossmann, *Advanced optimization for process systems engineering*. Cambridge University Press, 2021.

[13] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[14] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, 2009.

[15] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson, "A survey of distributed optimization," *Annu. Rev. Control*, vol. 47, pp. 278–305, 2019.

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

[17] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "DQM: Decentralized quadratically approximated alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 64, no. 19, pp. 5158–5173, 2016.

[18] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, "Asynchronous distributed ADMM for large-scale optimization – part I: Algorithm and convergence analysis," *IEEE Trans. Signal Process.*, vol. 64, no. 12, pp. 3118–3130, 2016.

[19] A. Makhdoumi and A. Ozdaglar, "Convergence rate of distributed ADMM over networks," *IEEE Trans. Autom. Control*, vol. 62, no. 10, pp. 5082–5095, 2017.

[20] P. Daoutidis, W. Tang, and S. S. Jogwar, "Decomposing complex plants for distributed control: perspectives from network theory," *Comput. Chem. Eng.*, vol. 114, pp. 43–51, 2018.

[21] W. Tang and P. Daoutidis, "The role of community structures in sparse feedback control," in *Am. Control Conf. (ACC)*. IEEE, 2018, pp. 1790–1795.

[22] P. H. Constantino, W. Tang, and P. Daoutidis, "Topology effects on sparse control of complex networks with Laplacian dynamics," *Sci. Rep.*, vol. 9, p. 9034, 2019.

[23] P. Daoutidis, W. Tang, and A. Allman, "Decomposition of control and optimization problems by network structure: concepts, methods and inspirations from biology," *AIChE J.*, vol. 65, no. 10, p. e16708, 2019.

[24] A. Allman, W. Tang, and P. Daoutidis, "DeCODe: a community-based algorithm for generating high-quality decompositions of optimization problems," *Optim. Eng.*, vol. 20, no. 4, pp. 1067–1084, 2019.

[25] W. Tang, P. Carrette, Y. Cai, J. M. Williamson, and P. Daoutidis, "Automatic decomposition of large-scale industrial processes for distributed MPC on the Shell-Yokogawa platform for advanced control and estimation (PACE)," in *Foundations of Computer-Aided Process Operations / Chemical Process Control (FOCAPO/CPC 2023)*, 2023.

[26] A.-L. Barabási, *Network science*. Cambridge University Press, 2016.

[27] M. A. Porter, J.-P. Onnela, and P. J. Mucha, "Communities in networks," *Not. AMS*, vol. 56, no. 9, pp. 1082–1097, 2009.

[28] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Soc. Netw.*, vol. 5, no. 2, pp. 109–137, 1983.

[29] C. J. Anderson, S. Wasserman, and K. Faust, "Building stochastic blockmodels," *Soc. Netw.*, vol. 14, no. 1-2, pp. 137–161, 1992.

[30] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Phys. Rev. E*, vol. 83, no. 1, p. 016107, 2011.

[31] T. P. Peixoto, "Hierarchical block structures and high-resolution model selection in large networks," *Phys. Rev. X*, vol. 4, no. 1, p. 011047, 2014.

[32] X. Zhang, T. Martin, and M. E. J. Newman, "Identification of core-periphery structure in networks," *Phys. Rev. E*, vol. 91, no. 3, p. 032803, 2015.

[33] P. Latouche, E. Birmele, and C. Ambroise, "Variational bayesian inference and complexity control for stochastic block models," *Stat. Model.*, vol. 12, no. 1, pp. 93–115, 2012.

[34] T. P. Peixoto, "Parsimonious module inference in large networks," *Phys. Rev. Lett.*, vol. 110, no. 14, p. 148701, 2013.

[35] ——, "Nonparametric bayesian inference of the microcanonical stochastic block model," *Phys. Rev. E*, vol. 95, no. 1, p. 012317, 2017.

[36] S. P. Borgatti and M. G. Everett, "Models of core/periphery structures," *Soc. Netw.*, vol. 21, no. 4, pp. 375–395, 2000.

[37] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 99, no. 12, pp. 7821–7826, 2002.

[38] M. E. J. Newman, "Equivalence between modularity optimization and maximum likelihood methods for community detection," *Phys. Rev. E*, vol. 94, no. 5, p. 052315, 2016.

[39] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, no. 2, p. 026113, 2004.

[40] M. J. Barber, "Modularity and community detection in bipartite networks," *Phys. Rev. E*, vol. 76, no. 6, p. 066102, 2007.

[41] E. A. Leicht and M. E. J. Newman, "Community structure in directed networks," *Phys. Rev. Lett.*, vol. 100, no. 11, p. 118703, 2008.

[42] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Phys. Rev. E*, vol. 74, no. 1, p. 016110, 2006.

[43] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 2, pp. 172–188, 2007.

[44] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Phys. Rev.*, vol. 659, pp. 1–44, 2016.

[45] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 103, no. 23, pp. 8577–8582, 2006.

[46] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech. Theor. Exp.*, vol. 2008, no. 10, p. P10008, 2008.

[47] C.-E. Bichot and P. Siarry, *Graph partitioning*. John Wiley & Sons, 2013.

[48] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell Syst. Tech. J.*, vol. 49, no. 2, pp. 291–307, 1970.

[49] A. Michel, R. Miller, and W. Tang, "Lyapunov stability of interconnected systems: Decomposition into strongly connected subsystems," *IEEE Trans. Circuit. Syst.*, vol. 25, no. 9, pp. 799–809, 1978.

[50] M. Vidyasagar, "Decomposition techniques for large-scale systems with nonadditive interactions: Stability and stabilizability," *IEEE Trans. Autom. Control*, vol. 25, no. 4, pp. 773–779, 1980.

[51] D. D. Šiljak, *Decentralized control of complex systems*. Academic Press, 1991.

[52] T. J. McAvoy, *Interaction analysis: Principles and applications*. ISA, 1983.

[53] E. Bristol, "On a new measure of interaction for multivariable process control," *IEEE Trans. Autom. Control*, vol. 11, no. 1, pp. 133–134, 1966.

[54] P. Grosdidier and M. Morari, "Interaction measures for systems under decentralized control," *Automatica*, vol. 22, no. 3, pp. 309–319, 1986.

[55] C.-C. Yu and M. K. H. Fan, "Decentralized integral controllability and D-stability," *Chem. Eng. Sci.*, vol. 45, no. 11, pp. 3299–3309, 1990.

[56] C. S. Ng and G. Stephanopoulos, "Synthesis of control systems for chemical plants," *Comput. Chem. Eng.*, vol. 20, pp. S999–S1004, 1996.

[57] M. L. Luyben, B. D. Tyreus, and W. L. Luyben, "Plantwide control design procedure," *AIChE J.*, vol. 43, no. 12, pp. 3161–3174, 1997.

[58] S. Skogestad, "Control structure design for complete chemical plants," *Comput. Chem. Eng.*, vol. 28, no. 1-2, pp. 219–234, 2004.

[59] W. D. Seider, D. R. Lewin, J. D. Seader, S. Widagdo, R. Gani, and K. M. Ng, *Product and process design principles: Synthesis, analysis and evaluation*, 4th ed. John Wiley & Sons, 2016.

[60] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Syst. Mag.*, vol. 22, no. 1, pp. 44–52, 2002.

[61] A. N. Venkat, J. B. Rawlings, and S. J. Wright, "Stability and optimality of distributed model predictive control," in *Proc. 44th Conf. Decis. Control (CDC)*. IEEE, 2005, pp. 6680–6685.

[62] J. Liu, D. Muñoz de la Peña, and P. D. Christofides, "Distributed model predictive control of nonlinear process systems," *AIChE J.*, vol. 55, no. 5, pp. 1171–1184, 2009.

[63] J. Liu, X. Chen, D. Muñoz de la Peña, and P. D. Christofides, "Sequential and iterative architectures for distributed model predictive control of nonlinear process systems," *AIChE J.*, vol. 56, no. 8, pp. 2137–2149, 2010.

[64] B. T. Stewart, A. N. Venkat, J. B. Rawlings, S. J. Wright, and G. Pannocchia, "Cooperative distributed model predictive control," *Syst. Control Lett.*, vol. 59, no. 8, pp. 460–469, 2010.

[65] B. T. Stewart, S. J. Wright, and J. B. Rawlings, "Cooperative distributed model predictive control for nonlinear systems," *J. Process Control*, vol. 21, no. 5, pp. 698–704, 2011.

[66] R. R. Negenborn and J. M. Maestre, "Distributed model predictive control: An overview and roadmap of future research opportunities," *IEEE Control Syst. Mag.*, vol. 34, no. 4, pp. 87–97, 2014.

[67] E. K. Ryu and S. Boyd, "Primer on monotone operator methods," *Appl. Comput. Math.*, vol. 15, no. 1, pp. 3–43, 2016.

[68] W. Tang and P. Daoutidis, "Distributed nonlinear model predictive control through accelerated parallel ADMM," in *Am. Control Conf.* IEEE, 2019, pp. 1406–1411.

[69] F. Farokhi, I. Shames, and K. H. Johansson, "Distributed MPC via dual decomposition and alternative direction method of multipliers," in *Distributed model predictive control made easy*. Springer, 2014, pp. 115–131.

[70] K. Sun and X. A. Sun, "A two-level distributed algorithm for general constrained non-convex optimization with global convergence," *arXiv preprint arXiv:1902.07654*, 2019.

[71] W. Tang and P. Daoutidis, "Fast and stable nonconvex constrained distributed optimization: the ELLADA algorithm," *Optim. Eng.*, vol. 23, pp. 259–301, 2022.

[72] ——, "Coordinating distributed MPC efficiently on a plantwide scale: The Lyapunov envelope algorithm," *Comput. Chem. Eng.*, vol. 155, p. 107532, 2021.

[73] P. D. Christofides, R. Scattolini, D. Muñoz de la Peña, and J. Liu, "Distributed model predictive control: A tutorial review and future research directions," *Comput. Chem. Eng.*, vol. 51, pp. 21–41, 2013.

[74] S. S. Jogwar and P. Daoutidis, "Community-based synthesis of distributed control architectures for integrated process networks," *Chem. Eng. Sci.*, vol. 172, pp. 434–443, 2017.

[75] S. S. Jogwar, "Distributed control architecture synthesis for integrated process networks through maximization of strength of input–output impact," *J. Process Control*, vol. 83, pp. 77–87, 2019.

[76] M. Xie, L. Zhang, and W. Xie, "Subsystem decomposition of complex nonlinear systems," *CIESC J.*, vol. 72, no. 3, pp. 1557–1566, 2021.

[77] D. B. Pourkargar and S. S. Jogwar, "Distributed model predictive control of integrated process networks: Optimal decomposition for varying operating point," in *American Control Conference (ACC)*. IEEE, 2021, pp. 801–807.

[78] W. Tang and P. Daoutidis, "Network decomposition for distributed control through community detection in input–output bipartite graphs," *J. Process Control*, vol. 64, pp. 7–14, 2018.

[79] W. Tang, D. Babaei Pourkargar, and P. Daoutidis, "Relative time-averaged gain array (RTAGA) for distributed control-oriented network decomposition," *AIChE J.*, vol. 64, no. 5, pp. 1682–1690, 2018.

[80] W. Tang, A. Allman, D. B. Pourkargar, and P. Daoutidis, "Optimal decomposition for distributed optimization in nonlinear model predictive control through community detection," *Comput. Chem. Eng.*, vol. 111, pp. 43–54, 2018.

[81] I. Mitrai, W. Tang, and P. Daoutidis, "Stochastic blockmodeling for learning the structure of optimization problems," *AIChE J.*, vol. 68, no. 6, p. e17415, 2022.

[82] D. B. Pourkargar, A. Almansoori, and P. Daoutidis, "Impact of decomposition on distributed model predictive control: A process network case study," *Ind. Eng. Chem. Res.*, vol. 56, no. 34, pp. 9606–9616, 2017.

[83] ——, "Comprehensive study of decomposition effects on distributed

output tracking of an integrated process over a wide operating range," *Chem. Eng. Res. Des.*, vol. 134, pp. 553–563, 2018.

[84] D. B. Pourkargar, M. Moharir, A. Almansoori, and P. Daoutidis, "Distributed estimation and nonlinear model predictive control using community detection," *Ind. Eng. Chem. Res.*, vol. 58, no. 30, pp. 13 495–13 507, 2019.

[85] W. He and S. Li, "Enhancing topological information of the Lyapunov-based distributed model predictive control design for large-scale nonlinear systems," *Asian J. Control*, 2022.

[86] P. Segovia, V. Puig, E. Duviella, and L. Etienne, "Distributed model predictive control using optimality condition decomposition and community detection," *J. Process Control*, vol. 99, pp. 54–68, 2021.

[87] P. H. Constantino and P. Daoutidis, "A control perspective on the evolution of biological modularity," *IFAC-PapersOnLine*, vol. 52, no. 11, pp. 172–177, 2019.

[88] W. Tang, P. H. Constantino, and P. Daoutidis, "Optimal sparse network topology under sparse control in Laplacian networks," *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 273–278, 2019.

[89] R. R. Rocha, L. C. Oliveira-Lopes, and P. D. Christofides, "Partitioning for distributed model predictive control of nonlinear processes," *Chem. Eng. Res. Des.*, vol. 139, pp. 116–135, 2018.

[90] X. Yin and J. Liu, "Subsystem decomposition of process networks for simultaneous distributed state estimation and control," *AIChE J.*, vol. 65, no. 3, pp. 904–914, 2019.

[91] L. S. Masooleh, J. E. Arbogast, W. D. Seider, U. Oktem, and M. Soroush, "An efficient algorithm for community detection in complex weighted networks," *AIChE J.*, vol. 67, no. 7, p. e17205, 2021.

[92] ——, "Distributed state estimation in large-scale processes decomposed into observable subsystems using community detection," *Comput. Chem. Eng.*, vol. 156, p. 107544, 2022.

[93] J. Wang, C. Song, J. Zhao, Z. Mo, and Z. Xu, "Distributed model predictive control-oriented network decomposition based on full dynamic response," *AIChE J.*, vol. 69, no. 1, p. e17951, 2023.

[94] M. R. Kilinc and N. V. Sahinidis, "Exploiting integrality in the global optimization of mixed-integer nonlinear programming problems with baron," *Optim. Meth. Softw.*, vol. 33, pp. 540–562, 2018.

[95] D. Bongartz, J. Najman, S. Sass, and A. Mitsos, "MAiNGO – McCormick-based Algorithm for mixed-integer Nonlinear Global Optimization," *Process Systems Engineering (AVT.SVT), RWTH Aachen University*, 2018.

[96] R. Misener and C. A. Floudas, "ANTIGONE: Algorithms for coNTinuous / Integer Global Optimization of Nonlinear Equations," *J. Global Optim.*, vol. 59, pp. 503–526, 2014.

[97] D. E. Bernal, S. Vigerske, F. Trespalacios, and I. E. Grossmann, "Improving the performance of DICOPT in convex MINLP problems using a feasibility pump," *Optim. Meth. Softw.*, vol. 35, pp. 171–190, 2020.

[98] M. R. Bussieck and S. Vigerske, "MINLP solver software," *GAMS Development Corporation*, 2014.

[99] A. J. Conejo, E. Castillo, R. Mínguez, and R. García-Bertrand, *Decomposition techniques in mathematical programming: Engineering and science applications*. Springer, 2006.

[100] O. E. Flippo and A. H. Rinnooy Kan, "Decomposition in general mathematical programming," *Math. Program.*, vol. 60, no. 1-3, pp. 361–382, 1993.

[101] A. M. Geoffrion, "Elements of large-scale mathematical programming part i: Concepts," *Manag. Sci.*, vol. 16, no. 11, pp. 652–675, 1970.

[102] ——, "Elements of large scale mathematical programming part ii: Synthesis of algorithms and bibliography," *Manag. Sci.*, vol. 16, no. 11, pp. 676–691, 1970.

[103] J. Jalving, S. Shin, and V. M. Zavala, "A graph-based modeling abstraction for optimization: concepts and implementation in Plasmo.jl," *Math. Program. Comput.*, vol. 14, pp. 699–747, 2022.

[104] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Oper. Res.*, vol. 8, pp. 1–157, 1960.

[105] A. Allman and Q. Zhang, "Dynamic location of modular manufacturing facilities with relocation of individual modules," *Eur. J. Oper. Res.*, vol. 286, pp. 494–507, 2020.

[106] J. Desrosiers, F. Soumis, and M. Desrochers, "Routing with time windows by column generation," *Networks*, vol. 14, pp. 545–565, 1984.

[107] G. Desaulniers, J. Desrosiers, and M. M. Solomon, "Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems," *Ess. Surv. Math.*, pp. 309–324, 2002.

[108] A. Wang, A. Subramanyam, and C. E. Gounaris, "Robust vehicle routing under uncertainty via branch-price-and-cut," *Optim. Eng.*, vol. 23, pp. 1895–1948, 2022.

[109] A. Klos and S. Gortz, "A branch-and-price algorithm for the capacitated facility location problem," *Eur. J. Oper. Res.*, vol. 179, pp. 1109–1125, 2007.

[110] K. J. Singh, A. B. Philpott, and R. K. Wood, "Dantzig-Wolfe decomposition for solving multistage stochastic capacity-planning problems," *Oper. Res.*, vol. 57, pp. 1271–1286, 2009.

[111] A. Flores-Quinoz, J. M. Pinto, and Q. Zhang, "A column generation approach to multiscale capacity planning for power-intensive process networks," *Optim. Eng.*, vol. 20, pp. 1001–1027, 2019.

[112] M. E. Lubbecke and J. Desrosiers, "Selected topics in column generation," *Oper. Res.*, vol. 53, pp. 1007–1023, 2005.

[113] A. Allman and Q. Zhang, "Branch-and-price for a class of nonconvex mixed-integer nonlinear programs," *J. Global Optim.*, vol. 81, pp. 861–880, 2021.

[114] J. Benders, "Partitioning procedures for solving mixed-variables programming problems ‘," *Numer. Math.*, vol. 4, no. 1, pp. 238–252, 1962.

[115] I. Mitrai and P. Daoutidis, "A multicut generalized Benders decomposition approach for the integration of process operations and dynamic optimization for continuous systems," *Comput. Chem. Eng.*, vol. 164, p. 107859, 2022.

[116] ——, "Efficient solution of enterprise-wide optimization problems using nested stochastic blockmodeling," *Ind. Eng. Chem. Res.*, vol. 60, no. 40, pp. 14 476–14 494, 2021.

[117] A. M. Geoffrion, "Generalized Benders decomposition," *J. Optim. Theor. Appl.*, vol. 10, pp. 237–260, 1972.

[118] N. Sahinidis and I. E. Grossmann, "Convergence properties of generalized Benders decomposition," *Comput. Chem. Eng.*, vol. 15, no. 7, pp. 481–491, 1991.

[119] M. J. Bagajewicz and V. Manousiouthakis, "On the generalized benders decomposition," *Comput. Chem. Eng.*, vol. 15, no. 10, pp. 691–700, 1991.

[120] A. Ruszczyński and A. Świetanowski, "Accelerating the regularized decomposition method for two stage stochastic linear problems," *Eur. J. Oper. Res.*, vol. 101, no. 2, pp. 328–342, 1997.

[121] J. Linderoth and S. Wright, "Decomposition algorithms for stochastic programming on a computational grid," *Comput. Optim. Appl.*, vol. 24, no. 2, pp. 207–250, 2003.

[122] T. L. Magnanti and R. T. Wong, "Accelerating benders decomposition: Algorithmic enhancement and model selection criteria," *Operations research*, vol. 29, no. 3, pp. 464–484, 1981.

[123] G. K. Saharidis and M. G. Ierapetritou, "Improving benders decomposition using maximum feasible subsystem (MFS) cut generation strategy," *Comput. Chem. Eng.*, vol. 34, no. 8, pp. 1237–1245, 2010.

[124] R. Pacqueau, F. Soumis, and L.-N. Hoang, *A fast and accurate algorithm for stochastic integer programming, applied to stochastic shift scheduling*. Groupe d'études et de recherche en analyse des décisions, 2012.

[125] J.-F. Cordeau, G. Stojkovic, F. Soumis, and J. Desrosiers, "Benders decomposition for simultaneous aircraft routing and crew scheduling," *Transport. Sci.*, vol. 35, pp. 375–388, 2001.

[126] J. Verstichel, J. Kinable, P. De Causmaecker, and G. Vanden Berghe, "A combinatorial Benders decomposition for the lock scheduling problem," *Comput. Oper. Res.*, vol. 54, pp. 117–128, 2015.

[127] J. R. Birge and F. V. Louveaux, "A multicut algorithm for two-stage stochastic linear programs," *Eur. J. Oper. Res.*, vol. 34, no. 3, pp. 384–392, 1988.

[128] F. You and I. E. Grossmann, "Multicut benders decomposition algorithm for process supply chain planning under uncertainty," *Ann. Oper. Res.*, vol. 210, no. 1, pp. 191–211, 2013.

[129] X. Li, A. Tomasgard, and P. I. Barton, "Nonconvex generalized benders decomposition for stochastic separable mixed-integer nonlinear programs," *J. Optim. Theor. Appl.*, vol. 151, no. 3, pp. 425–454, 2011.

[130] S. Rebennack, "Combining sampling-based and scenario-based nested benders decomposition methods: application to stochastic dual dynamic programming," *Math. Program.*, vol. 156, pp. 343–389, 2016.

[131] I. Mitrai and P. Daoutidis, "Learning to initialize generalized Benders decomposition via active learning," in *FOCAPO/CPC, San Antonio, Texas*, 2023.

[132] E. Larsen, E. Frejinger, B. Gendron, and A. Lodi, "Fast continuous and integer L-shaped heuristics through supervised learning," *arXiv preprint arXiv:2205.00897*, 2022.
[133] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei, "The benders decomposition algorithm: A literature review," *Eur. J. Oper. Res.*, vol. 259, no. 3, pp. 801–817, 2017.
[134] Y. Nie, L. T. Biegler, and J. M. Wassick, "Integrated scheduling and dynamic optimization of batch processes using state equipment networks," *AIChE J.*, vol. 58, no. 11, pp. 3416–3432, 2012.
[135] J. N. Hooker, "Planning and scheduling by logic-based Benders decomposition," *Oper. Res.*, vol. 55, pp. 588–602, 2007.
[136] X. Li, Y. Chen, and P. I. Barton, "Nonconvex generalized Benders decomposition with piecewise convex relaxations for global optimization of integrated process design and operation problems," *Ind. Eng. Chem. Res.*, vol. 51, pp. 7287–7299, 2012.
[137] M. Guignard and S. Kim, "Lagrangean decomposition: A model yielding stronger Lagrangean bounds," *Math. Program.*, vol. 39, pp. 215–228, 1987.
[138] A. Allman and Q. Zhang, "Distributed fairness-guided optimization for coordinated demand response in multi-stakeholder process networks," *Comput. Chem. Eng.*, vol. 161, p. 107777, 2022.
[139] E. Gunn, M. Thorburn, and A. Rai, "A decomposition method based on the augmented Lagrangian," *INFOR: Inform. Syst. Oper. Res.*, vol. 26, pp. 91–113, 1988.
[140] M. R. Hestenes, "Multiplier and gradient methods," *J. Optim. Theor. Appl.*, vol. 4, pp. 303–320, 1969.
[141] J. Eckstein and W. Yao, "Approximate ADMM algorithms derived from Lagrangian splitting," *Comput. Optim. Appl.*, vol. 68, no. 2, pp. 363–405, 2017.
[142] D. G. Anderson, "Iterative procedures for nonlinear integral equations," *J. ACM*, vol. 12, no. 4, pp. 547–560, 1965.
[143] P. Pulay, "Convergence acceleration of iterative sequences. the case of SCF iteration," *Chem. Phys. Lett.*, vol. 73, no. 2, pp. 393–398, 1980.
[144] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Trans. Signal Process.*, vol. 63, pp. 482–497, 2014.
[145] G. Xiao, Y. Yan, J. Bao, and F. Liu, "Robust distributed economic model predictive control based on differential dissipativity," *AIChE J.*, vol. 67, p. e17198, 2021.
[146] R. Halvgaard, L. Vandenberghe, N. K. Poulsen, H. Madsen, and J. B. Jorgensen, "Distributed model predictive control for smart energy systems," *IEEE Trans. Smart Grid*, vol. 7, pp. 1675–1682, 2016.
[147] P. Braun, T. Faulwasser, L. Grune, C. M. Kellett, S. R. Weller, and K. Worthmann, "Hierarchical distributed ADMM for predictive control with applications in power networks," *IFAC J. Syst. Control*, vol. 25, pp. 10–22, 2018.
[148] M. Mork, A. Xhonneax, and D. Muller, "Nonlinear distributed model predictive control for multi-zone building energy systems," *Energy Build.*, vol. 264, p. 112066, 2022.
[149] R. Van Parys and G. Pipeleers, "Distributed MPC for multi-vehicle systems moving in formation," *Robot. Autonom. Syst.*, vol. 97, pp. 144–152, 2017.
[150] H. Zheng, R. R. Negenborn, and G. Lodewijks, "Fast ADMM for distributed model predictive control of cooperative waterborne AGVs," *IEEE Trans. Control Syst. Technol.*, vol. 25, pp. 1406–1413, 2016.
[151] L. Xie, X. Cai, J. Chen, and H. Su, "GA based decomposition of large scale distributed model predictive control systems," *Control Eng. Pract.*, vol. 57, pp. 111–125, 2016.
[152] T. J. Van Roy, "Cross decomposition for mixed integer programming," *Math. Program.*, vol. 25, pp. 46–63, 1983.
[153] J. Kronqvist, D. E. Bernal, A. Lundell, and I. E. Grossmann, "A review and comparison of solvers for convex MINLP," *Optim. Eng.*, vol. 20, pp. 397–455, 2019.
[154] M. Kruber, M. E. Lübbecke, and A. Parmentier, "Learning when to use a decomposition," in *Integration of AI and OR Techniques in Constraint Programming: 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017, Proceedings 14*. Springer, 2017, pp. 202–210.
[155] J. R. Rice, "The algorithm selection problem," in *Advances in computers*. Elsevier, 1976, vol. 15, pp. 65–118.
[156] I. Mitrai and P. Daoutidis, "A graph classification algorithm to determine when to decompose optimization problems," in *ESCAPE-33, Athens, Greece*, 2023.