AUTOMATIC DECOMPOSITION OF LARGE-SCALE INDUSTRIAL PROCESSES FOR DISTRIBUTED MPC ON THE SHELL-YOKOGAWA PLATFORM FOR ADVANCED CONTROL AND ESTIMATION (PACE)

Wentao Tang [a,b,*], Pierre Carrette [a], Yongsong Cai [a], John M. Williamson [a], and Prodromos Daoutidis [c]

[a] *Surface Operations, Projects and Technology, Shell Global Solutions (U.S.) Inc., Houston, TX 77082*
[b] Next position: *Department of Chemical and Biomolecular Engineering, NC State University, Raleigh, NC 27606*
[c] *Department of Chemical Engineering and Materials Science, University of Minnesota, Minneapolis, MN 55455*

Abstract – The kernel of advanced process control (APC) technology lies in the formulation and solution of model predictive control (MPC) problems. A significant challenge in the contemporary practice of APC is its efficient online implementation on large-scale industrial systems. As the state-of-the-art APC technology, the Platform for Advanced Control and Estimation (PACE) of Shell and Yokogawa has adopted a systematic framework of handling dynamic optimization of large-scale systems, where an automatic decomposition procedure has been developed to generate subsystems for distributed MPC. The decomposition is implemented on network representations of the MPC models that capture interactions among process variables, and is based on the concept of community detection which aims to maximize the statistical significance of the subsystems as subnetworks with preferred internal interconnections. This paper introduces the fundamentals of such a decomposition approach and the incorporation of this functionality into PACE, followed by a case study on a crude distillation process to showcase the advantages of its application on industrial problems.

Keywords – Network decomposition, community detection, model predictive control, plantwide control

Introduction

The deployment of advanced process control (APC) in the chemical process industry since the 1970s has brought significant benefits (Qin and Badgwell, 2003). An APC platform typically adopts a model predictive control (MPC) formulation to handle multi-input-multi-output constrained systems. The control decisions are thus made according to the solution of an optimal control problem, where a cost function is minimized subject to model dynamics and constraints (Rawlings et al., 2017). Such an APC platform allows stable process operations, increases economic profits, reduces emissions, and decreases the frequency of alarms and operator intervention. Shell's pioneering APC technology, from Dynamic Matrix Control (DMC) (Cutler and Ramaker, 1979), Quadratic Dynamic Matrix Control (QDMC) (Garcia and Morshedi, 1986), Shell Multivariable Optimizing Control (SMOC) (Marquis and Broustail, 1988), and SMOCPro (Cott, 2007), to the Platform for Advanced Control and Estimation (PACE) developed in alliance with Yokogawa (Amrit et al., 2015), has been continually refined with the advance of computing and software capabilities.

Large scale and tight material and energy integration are typical features of modern plants (Baldea and Daoutidis, 2012). A key challenge to the control of such large-scale process networks is that a centralized paradigm of optimizing monolithically over the entire system is undesirable, either due to the computational time needed by the optimization solver or due to its inflexibility. On the other hand, simply partitioning the system into multiple parts and controlling each of them as if they do not interact with each other (decentralized control) usually will not retain the benefits of a system-wide MPC. Therefore, *decomposition and coordination* are essential to achieve large-scale optimal control decisions on the basis of *interacting subsystems*. This paradigm, which can be traced at least back to Morari et al. (1980), is frequently referred to as "*distributed MPC*" (Scattolini, 2009; Christofides et al., 2013).

Two key questions in the implementation of distributed MPC are *how to decompose the process model* and *how to coordinate the subsystem controllers*. In this paper, we focus on the first question. Specifically, to best maintain the control performance under decomposition and reduce the computational time, the subsystems should be configured in such a way that the overall interactions across the subsystems are

* Corresponding author. Email: wtang23@ncsu.edu.

much weaker than those inside the subsystems. A network-theoretic framework is used to this end, where the structure of the dynamic system to be controlled is first represented as a network (graph) and a *community detection* procedure is applied to this network representation to generate a desirable decomposition. This approach was proposed by authors of this paper and their coworkers in their academic research (Daoutidis et al., 2018, 2019). This paper reports its implementation on Shell's industrial process control platform with a refined community detection algorithm that is more suitable for practical application.

In the rest of the paper, we provide a conceptual introduction of the design of the PACE technology and briefly review the relevant literature to elucidate the fundamental ideas underlying the decomposition strategy. We outline how community detection has been successfully implemented in PACE and use a challenging problem from Shell's petrochemical processes to demonstrate the benefits of decomposition and coordination in handling large-scale systems. From a broader perspective, this work contributes to a better appreciation of the contemporary theory and practice of APC, of which Shell-Yokogawa's PACE technology is representative, and documents a successful industry-academia synergy for bridging the gap between theory and practice.

Literature Review on Automatic Decomposition

The query for a decomposition in control problems dates back at least to the 1970s in the studies pertinent to stability analysis of decentralized control (Michel et al., 1978; Vidyasagar, 1980). 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 proposed (Šiljak, 1991). These methods require restrictive system structures and are not suitable for chemical processes that are generally well connected as a whole. In a different vein, works on interaction analysis (McAvoy, 1983) used the concept of relative gain array (RGA) (Bristol, 1966) to pair inputs and outputs. Especially with the development of robust control theory after the 1980s, the interaction analysis was combined with stability analysis (Grosdidier and Morari, 1986; Yu and Fan, 1990), thus providing systematic guidelines to design base-layer control loops on plantwide scales (Ng and Stephanopoulos, 1996; Skogestad, 2004). However, for a significant period after distributed MPC was proposed, the decomposition of large-scale systems in the sense of partitioning into several MPC subsystems (i.e., optimization subproblems) had not been well addressed (Christofides et al., 2013).

The emergence of network science has brought forth the understanding of the organization of large-scale networks by investigating their macroscopic topological features and the dynamics associated with them (Barabási, 2016). Community structure is a typical block structure existing in many biological networks (Girvan and Newman, 2002), which refers to blocks with significantly denser interconnections inside these blocks than those among them. As pointed out in a number of studies (Tang and Daoutidis, 2018b; Constantino et al., 2019; Constantino and Daoutidis, 2019; Tang et al., 2019), the fundamental role of community structures in the control of networks lies in that they lend themselves to the adoption of *modular* controllers, which promotes the feedback sparsity (i.e., reduces the controller complexity) while preserving the control performance. This paves the way for a systematic framework of large-scale process decomposition.

A versatile range of network representations of dynamical systems has been proposed, which flexibly capture the interactions among process variables under different characterizations. These include directed graphs (digraphs) for the interactions among manipulated inputs, states, and controlled outputs (Jogwar and Daoutidis, 2017) and bipartite graphs for input–output relations, which can be weighted by appropriately defined interaction measures (Tang and Daoutidis, 2018a; Tang et al., 2018b). In Tang et al. (2018a), network representations were proposed to directly capture the variable-constraint interactions in the optimization formulation of the MPC problem, which allows the decomposition of optimization problems in general (Allman et al., 2019; Mitrai and Daoutidis, 2020).

In the above-mentioned works, two algorithms of community detection were highlighted, namely Newman's *spectral algorithm* (Newman, 2006) and the Louvain (fast unfolding) algorithm (Blondel et al., 2008). Both algorithms aim to maximize a quality index, called *modularity*, which characterizes the difference between the density of interconnections among nodes inside the same communities and such density in a randomized network (Newman and Girvan, 2004), and can be interpreted as the statistical significance of the existence of community structures in the network (Newman, 2016). 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 in finding decompositions with a higher modularity value. However,

- 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 the 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, we consider Newman's spectral algorithm as more suitable for the purpose of decomposing control problems.

The advantage of community detection-based network decomposition has been well demonstrated in the literature through simulations (Pourkargar et al., 2017, 2019). In these studies, the subsystem controllers in a distributed MPC scheme iterate their decisions according to a block coordinate descent algorithm in sequential or parallel orders. It was noted in these studies that community-based decompositions result in significant decrease of computational time without large degradation in the control performance. With these observations, we adopt community detection as the method of choice for decomposing large-scale systems in PACE.

Overview of Platform for Advanced Control and Estimation (PACE)

In this section we give a high-level overview of our current technology platform, PACE, highlighting the distinctive features that differentiate it from its counterparts. Fig. 1 shows a historical roadmap of APC technology at Shell since the 1970s. The development of PACE since the 2010s was driven by the needs of improving the performance and operational acceptance, facilitating migrations and new deployments, and reducing the cost and time for software maintenance. So far, PACE has been applied to most of the refining, chemical, liquified natural gas, and gas-to-liquid plants in Shell as well as to processes outside Shell through partnership with Yokogawa.

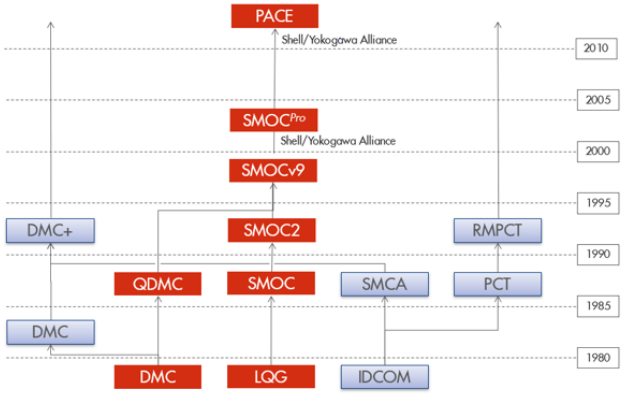


Figure 1: History and differentiation of APC technology

In a nutshell, PACE is an APC platform for all-in-one solution of data analysis, system identification and modeling, model quality validation, disturbance estimation and Kalman filtering, and static, dynamic, and economic optimization, and automatic step testing. The PACE software is divided into a "Design-Time" part, responsible for offline procedures including identification, controller design, simulations, and a "Run-Time" part for online controller configuration and tuning, step tests, and performance reporting (Carrette, 2020).

Modeling and Step Testing

PACE enables the modeling of rich cause-and-effect structures. As illustrated in Fig. 2, the skeleton of a PACE model is dynamics (transfer functions) from manipulated variables (MVs) and disturbance variables (DVs) to intermediate variables (IVs) to process output variables (POVs). The involvement of IVs allows the modeling of complex systems convoluted from simpler transfer functions, easier maintenance, and the feedforward of non-output measurements as calibration contributions. It also facilitates the incorporation of base-layer control (BLC) models, ranging from simple BLC loops, saturated, coupled, to cascaded ones, in a highly flexible manner. Nonlinear blocks can also be specified for POVs based on user-defined equations.

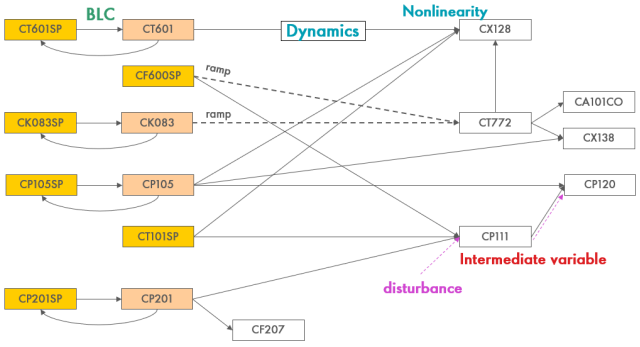


Figure 2: Illustration of PACE model structure

The step test is a crucial yet expensive prerequisite procedure of collecting the necessary data for model identification. In PACE, the step test becomes automated with an internal package integrated with control, thus simplifying the transition between control and testing. The exciting signals for the auto-step are designed such that the information content contained in data is optimized. During the step test, constraints are well managed, which is combined with the planning of subsequent experiments.

Controller Tuning and Calibration

The way that static and dynamic optimization are handled in PACE improves both the operational flexibility and accuracy. First, in static optimization, multiple economic functions (EFs) specified by linear or nonlinear equations, together with controlled variable (CV) specifications, can be user-defined with a hierarchy of priorities. Second, the dynamic optimization formulation is generated from the controller tuning parameters. Instead of tuning the weighting matrices (Q,R) as in usual MPC, the PACE users specify the speeds of MV responses, CV responses and dynamic tracking of the EFs. Such a tuning strategy allows more consistent and accurate responses, management of overshooting, and also adaptation to activation/inactivation of CVs. Moreover, a state-of-the-art commercial nonlinear optimization solver replaces traditional linear programming (LP) solvers for handling nonlinear processes.

To achieve offset-free control (Muske and Badgwell, 2002; Pannocchia and Rawlings, 2003), the optimization formulations involve a calibration mechanism, which accounts for and aims at eliminating the discrepancies between model prediction and actual POV values. PACE calibration is a combination of (i) event detection for detecting fast significant disturbances and (ii) Kalman filter for handling usual white noise sources (with the filtering speed tuned by the user). The calibration is furthermore de-tuned for robustness, by assum-

ing uncertainty factors on the transfer function models. In its unmeasured disturbance modeling, PACE incorporates both input and output disturbances, whose shape parameters are tunable. Such flexible designs along with the use of IVs in the model structure allow the calibration algorithm to better capture the disturbances than the classical bias estimation and update scheme (e.g., Cutler and Ramaker (1979)).

Subsystems in PACE

Equipped with decomposition and coordination strategies, PACE is currently developing the capability of handling large-scale challenging problems. The configuration of subsystems can be either directly user-specified (i.e., the user assigns all MVs and POVs or part of them into several subsystems), or preferably determined automatically by community detection. With the configuration, the process variables and model blocks are contained in the subsystems, and hence the optimization variables, constraints, and objectives in the dynamic optimization problems of the subsystems are automatically formulated. The distributed MPC essentially applies an iterative algorithm where in each iteration, the subsystem problems are solved through parallel computing. For online implementation, the computational time allowed for distributed optimization computation is highly restricted. In practice, we are typically limited to a single or a few iterations at every sampling period. The solutions obtained from such early termination should be tested before commissioning, and ad hoc measures can be taken to ensure that the closed-loop dynamic behavior is satisfactory.

The iterative algorithm that is currently used involves the following treatment:

- For the dynamic optimization problem of each subsystem, the MV and CV weights are tuned based on the subsystem model alone. In addition, quadratic terms associated with the *shared variables* (i.e., the variables that affect other subsystems) are added to the subsystem's objective function, and such quadratic terms are tuned as if they are CVs.
- After each iteration, the solutions associated with the predicted trajectories of the shared variables are fed forward to their downstream subsystems (i.e., those that they affect), as if such shared variables were DVs. In this way, the optimization problems are restricted to the variables inside subsystems.
- The convex combination scheme of Stewart et al. (2011) is used, i.e., if $\hat{u}_i^*(t)^{[k+1]}$ is the optimal solution for MV u_i at predicted time t in iteration $k+1$, only a fraction $\beta_i \in [0,1]$ of the update will be actually taken:

$$\hat{u}_i(t)^{[k+1]} = \hat{u}_i(t)^{[k]} + \beta_i \left(\hat{u}_i^*(t)^{[k+1]} - \hat{u}_i(t)^{[k]} \right). \qquad (1)$$

  Here the coefficient β_i depends only on the MV index i and does not vary with time index t throughout the receding horizon. The choice of β_i is rationally performed based on a Hessian approximation procedure.

In the literature, relevant advances have been made for real-time iterations in centralized MPC (Diehl et al., 2005; Yang and Biegler, 2013) and acceleration techniques in distributed optimization (Tang and Daoutidis, 2022, 2021). We believe that the development of more efficient and real-time implementable distributed optimization algorithms with guaranteed performance is an important direction of future research.

Community Detection in PACE

In the previous sections, we reviewed the literature about decomposing large-scale systems through community detection in networks and introduced the main technical features of PACE. In this section we present how automatic decomposition is realized in PACE.

Network Representation

As described previously, the main body of the model structure in PACE comprises model blocks (transfer functions) from MVs, IVs, and POVs. This naturally allows the construction of a *directed network* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set of nodes $\mathcal{V} = \{1, 2, \ldots, n\}$ represent the variables, and each directed edge (i, j) in the edge set $\mathcal{E}$ corresponds to the model block from variable i to variable j, if such a model block exists. The topology of the directed network can be represented by a sparse *adjacency matrix* A, where its (i, j)-th entry $a_{ij} = 1$ if $(i, j) \in \mathcal{E}$ and $a_{ij} = 0$ if $(i, j) \notin \mathcal{E}$. In our current implementation, we do not assign weights to the edges for the following reasons.

1. It can be argued that since the model is established through system identification, negligible, insignificant, or physically meaningless model blocks should have either not been identified or already removed by the control engineer.
2. The decomposition should be such that the subsystems have minimal *number* of interactions among them rather than total *weights*, since the number may affect the computational performance more strongly.
3. There does not exist a rigorous and clearly defined way of defining edge weights that guarantees any property of the resulting distributed MPC.

We also perform some pre-processing steps on the network representation. These procedures are needed to ensure that the complexity of the PACE model is resolved before the decomposition and that the result can be better understood and accepted by the user.

- The variables that do not participate in the dynamic optimization, including disturbance variables and inactivated MVs, are removed from the network. If such removal results in isolated nodes (which do not connect to any other node), then the isolated nodes are not considered for community detection, but only assigned to a separate subsystem afterwards.
- For each BLC loop, the variables involved should be assigned to the same subsystem. Hence, these BLC variables are considered as indivisible groups and first agglomerated as a single node. The same is carried out for the variables involved in each static nonlinear transformation.

Modularity and Resolution Parameter Tuning

The community detection in the directed network aims at maximizing the *modularity* (Leicht and Newman, 2008), which is a function of all possible partitions $g = (g_1, \ldots, g_n)$:

$$Q(g) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{m} \left(a_{ij} - \gamma \frac{k_i^+ k_j^-}{m} \right) \delta_{g_i g_j}, \tag{2}$$

where k_i^+ is the out-degree of node i, k_j^- is the in-degree of node j, and m is the total number of edges:

$$k_i^+ = \sum_{j=1}^{n} a_{ij}, \;\; k_j^- = \sum_{i=1}^{n} a_{ij}, \;\; m = \sum_{i=1}^{n} k_i^+ = \sum_{j=1}^{n} k_j^-. \tag{3}$$

g_i is the index of the community to which node i belongs, and δ is the Kronecker's delta, i.e., $\delta_{g_i g_j} = 1$ if nodes i and j are in the same community and 0 otherwise. Thus, $k_i^+ k_j^- / m$ is regarded as the expected number of edges between nodes i and j in a randomized network and therefore a "standard threshold" whose difference with a_{ij} is the extent to which these two nodes prefer to be in the same community. The parameter $\gamma > 0$ here, called the *resolution parameter*, offers a tuning of this threshold (Reichardt and Bornholdt, 2006). When γ is increased, the community detection tends to find a larger number of smaller communities, while smaller γ promotes a coarser decomposition.

For the user's convenience, we allow the user to simply specify the desired number of communities K and an algorithm is used to adaptively find the resolution γ such that the resulting decomposition is into K subsystems.

- If it is known that at γ_1 and γ_2, the maximization of Q leads to $K_1 < K$ and $K_2 > K$ communities, respectively, then assuming that $\ln K$ and $\ln \gamma$ have a linear relation approximately[1], we update γ by

$$\ln \gamma = \frac{\ln K_2 - \ln K}{\ln K_2 - \ln K_1} \ln \gamma_1 + \frac{\ln K - \ln K_1}{\ln K_2 - \ln K_1} \ln \gamma_2. \tag{4}$$

- At the first iteration, $\gamma = 1$ is used. When either the lower bound γ_1 or the upper bound γ_2 of the resolution parameter γ is not known (without loss of generality, say that only γ_1 is known), then we assume that $K \propto \gamma$, and update by

$$\gamma = \frac{K}{K_1} \gamma_1. \tag{5}$$

- After the update, if under γ the number of communities is exactly K, then terminate the iterations. Otherwise accordingly update γ_1 and K_1 or γ_2 and K_2.

Empirically, we found that the above rules allow us to find the correct resolution parameter for a given K within 10 iterations.

Spectral Algorithm for Recursive Bisectioning

Under a given γ, the maximization of modularity Q follows a recursive bisectioning procedure. Defining $c_{ij} = a_{ij} - \gamma k_i^+ k_j^- / m$, as pointed out in Newman (2006), when a community $\mathcal{S}$ is partitioned into two sub-communities $\mathcal{S}_+$ and $\mathcal{S}_-$, the resulting modularity increase is

$$\Delta Q(s) = \frac{1}{m} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} (s_i s_j - 1) c_{ij} = \frac{1}{m} \left(s^\top C_{\mathcal{S}} s - e^\top C_{\mathcal{S}} e \right), \tag{6}$$

where $s = (s_i)_{i \in \mathcal{S}}$, $s_i = +1$ if $i \in \mathcal{S}_+$ and -1 if $i \in \mathcal{S}_-$, $C_{\mathcal{S}} = [c_{ij}]_{i,j \in \mathcal{S}}$, e is a vector whose all elements equal to 1.

To maximize $\Delta Q(s)$ with respect to $s \in \{-1, 1\}^{|\mathcal{S}|}$, the following techniques are used.

- An approximate solution $s = \text{sign}(v_1(C'_{\mathcal{S}}))$ is taken first, in which $C'_{\mathcal{S}} = (C_{\mathcal{S}} + C_{\mathcal{S}}^\top)/2$, $v_1(\cdot)$ refers to the unit vector associated with the largest eigenvector of the matrix, and $\text{sign}(\cdot)$ is an element-wise sign function.
- The vector s is further *fine-tuned* by tentatively flipping the sign of each component, and the flipping with the maximum increase in ΔQ is accepted each time.

In the end, if the maximized $\Delta Q(s)$ is above a threshold value $\alpha = 10^{-3}$, then the bisectioning is accepted. Such a threshold value α prevents the production of extremely small communities[2]. Also, to guarantee the numerical stability of fine-tuning, we require that for the sign flipping to be accepted, its resulting modularity increase must be at least $\alpha/10$ and the total number of such flippings in each bisectioning should not exceed the number of nodes in $\mathcal{S}$.

Connectedness Restoration and Load Balancing

It is possible that after the spectral method, the communities found are not connected inside themselves, in which case the subsystems can not be considered as physically coherent portions of the process. Also, the sizes of communities may differ significantly, which is undesirable from the perspective of parallel computing in distributed MPC. Therefore, we carry out two major post-processing steps after modularity maximization.

First, in every community detected, a depth-first search (DFS) is performed to characterize all its *connected components* (i.e., subgraphs in which every pair of nodes are connected by at least an undirected path). Except for the largest connected component, which will preserve the identity of the

[1] According to the statistical interpretation of Newman (2016), the resolution γ should be chosen as $\gamma = (\omega_0 - \omega_1)/(\ln \omega_0 - \ln \omega_1)$, where the "propensity parameters" ω_0 and ω_1 are such that if $g_i = g_j$, the expectation of a_{ij} is $\omega_0 k_i^+ k_j^- / m$, and if $g_i \neq g_j$, the expectation of a_{ij} is $\omega_1 k_i^+ k_j^- / m$. Suppose that we have K communities, and the fraction of edges across communities is $\varepsilon \ll 1$, then we should have $\omega_0 + (K-1)\omega_1 = K$ and $(1-\varepsilon)/\omega_0 + \varepsilon/\omega_1 = 1$, which results in $\gamma \approx K / \ln(1/\varepsilon)$.

[2] Consider a large network with n nodes in which there are $\alpha \cdot n$ nodes ($\alpha \ll 1$) not connected to the rest of the network. Assigning this group of nodes, denoted as $\mathcal{S}$, into a separate community leads to $\Delta Q = \sum_{i \in \mathcal{S}} \sum_{j \notin \mathcal{S}} k_i^+ k_j^- / m^2 \approx m_{\mathcal{S}} / m$. Here $m_{\mathcal{S}}$ is the number of edges inside $\mathcal{S}$. If the edge distributions in and out of $\mathcal{S}$ are uniform, then $\Delta Q \approx m_{\mathcal{S}} / m \approx \alpha$.

community, every remaining connected component is moved into another community. This destination community is chosen as the one with which the connected components have most connections, and in the case that the connected component has no connection with any other community, i.e., is isolated, a new community is created for the isolated component. Such adjustment steps are repeated until no community has more than 1 connected components.

Load balancing is then performed by recursively merging the smallest community into a larger one.

- Suppose that before balancing, the largest community has n_1 nodes. Then we consider the "effective" number of communities as $K_e = \lfloor n/n_1 \rfloor$ and merging should be done for the communities smaller than the K_e-th one.
- For each small community to be merged, we look for a destination community whose size, when added to the size of this small community, is closest to n/K_e.
- We also require the two communities to merge to be connected. If a community does not connect to any other one, then this criterion is not applied.

In the tuning of resolution parameter γ, the number of communities K under the γ should be comprehended as the K_e here after load balancing.

Case Study

For the purpose of illustration, we consider a crude distillation process for a refinery, whose model contains 363 inputs, 381 outputs (among which there are 246 intermediate variables that are both inputs and outputs), and 801 model blocks, as visualized in Fig. 3. A full formulation of its dynamic optimization problem contains 875972 rows (constraints) and 828016 columns (variables), with 2430226 nonzero relations between the variables and constraints[3]. To our best knowledge, there has not been any work in the literature that addresses distributed MPC on a comparable plantwide scale, although problems with such complexity is common in practice.

The state-of-the-art optimization solver used in PACE takes 35.4 seconds to solve a single centralized dynamic optimization problem on average, which is too high compared to the sampling time of 60 seconds. Such a controller implemented for online operations has been well known to suffer from severe delays and also frequent shut-off due to computational timeout[4]. Therefore, automatic decomposition is useful in keeping the controllers under normal operations for such large-scale systems.

The directed network of variables that represents this process is shown in Fig. 4(a), with 476 nodes and 787 edges. An intuitive glance at the network topology suggests that there exists community structure in the process, and hence decomposing such a system is desirable. By examining the computational time and control performance under different numbers of communities K, we empirically set $K = 5$, which results in a good trade-off. The resulting decomposition into 5 communities is shown in Fig. 4(b). The number of edges lying across communities is 31, which is 3.9% of the total number of edges, and the number of inputs impacting outputs in other subsystems is 29, which is 8.0% of the total number of inputs. These indices demonstrate that the subsystems are indeed weakly coupled.

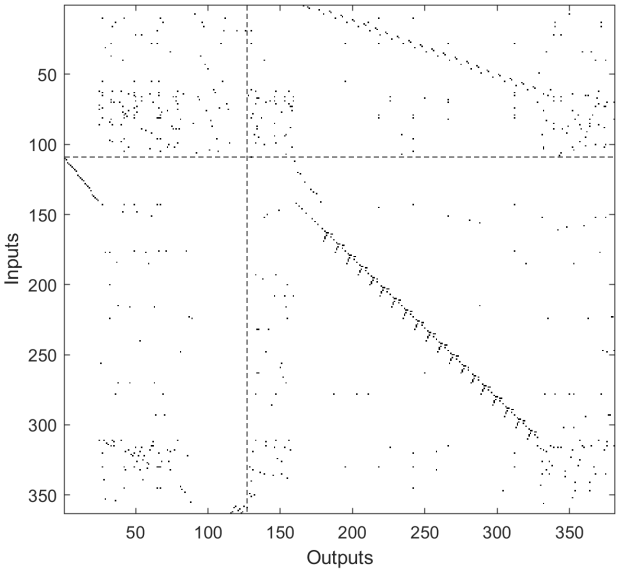


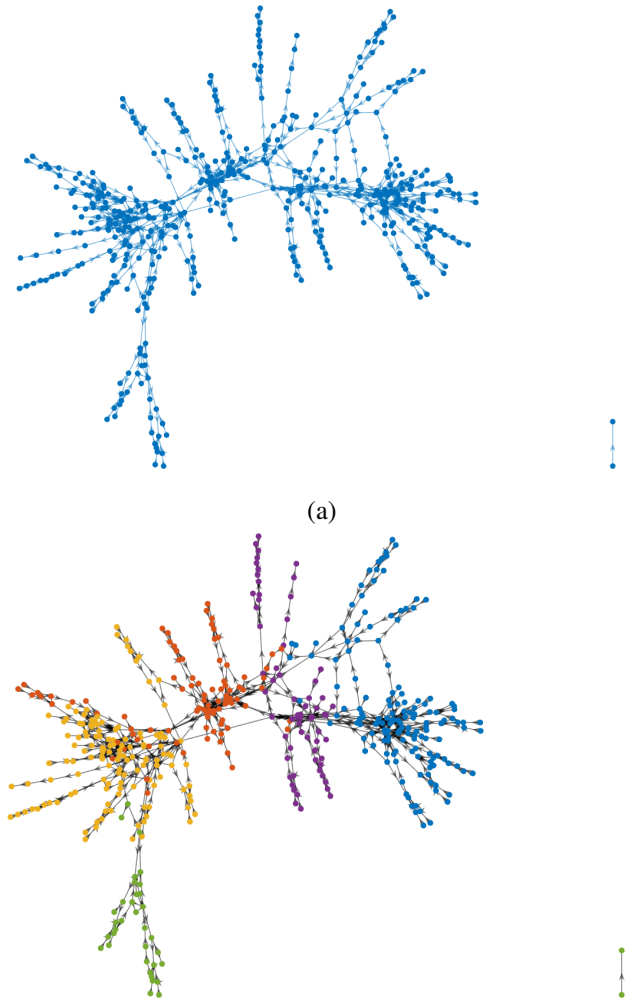
Figure 3: Model structure of the crude distillation process. (A black pixel stands for the presence of a model block. The dashed lines separate non-IVs (upper among inputs, left among outputs) and IVs (lower among inputs, right among outputs).

After the decomposition by community detection, a simulation is run for the closed-loop system. The simulated scenario considers disturbances happening on several MVs and POVs. The average computational time for solving the dynamic optimization problem at each sampling time is now reduced to 6.1 seconds, with an acceleration factor of 5.8. The trajectories under centralized MPC and under distributed MPC exhibit highly similar behaviors (they are omitted for brevity here). Such significant improvement in the computational performance of dynamic optimization has also been observed in several other benchmark processes of Shell. For a gas-to-liquid process with 68 MVs and 412 CVs, after decomposition into 4 subsystems, the average computational time is reduced from 10.0 to 3.3 seconds (i.e., the speed is accelerated with a factor of 3.0). For a hydrocracking process whose model comprises of 25 MVs and 117 CVs, a 4-subsystem decomposition accelerates the computation from

[3] Necessary simplification of the dynamic optimization problem, such as using a minimal horizon length for closed-loop stability and agglomerating time indices to "sparsify" the receding horizon, have already been used.

[4] The average computational time mentioned here was collected under the simulation of a typical normal operating condition. Occasionally, due to the parameter changes in the dynamic optimization problem, the computational time can reach over 70 seconds.

16.0 to 3.9 (with a factor of 4.1).



(a)

(b)

Figure 4: Network representation (a) and a 5-subsystem decomposition (b) of the crude distillation process. (Different colors correspond to communities.)

Conclusions

In this paper, we focused on the idea of *automatic decomposition*, which is necessary for structured, systematic solution of dynamic optimization problems arising in the MPC of large-scale systems. Following a literature review on the evolution of relevant academic research throughout multiple decades, we presented the successful implementation of an automatic decomposition method in the Shell-Yokogawa's new-generation APC platform – PACE, and showed its advantages when applied to real-world large-scale industrial processes. We thus demonstrated how the fundamental idea of decomposition originating in the early ages of APC, cross-pollinated with recent academic advances, has reshaped and empowered a leading modern process control technology.

References

Allman, A., Tang, W., and Daoutidis, P. (2019). DeCODe: a community-based algorithm for generating high-quality decompositions of optimization problems. *Optim. Eng.*, 20(4):1067–1084.

Amrit, R., Canney, W., Carrette, P., Linn, R., Martinez, A., Singh, A., Skrovanek, T., Valiquette, J., Williamson, J., and Zhou, J. (2015). Platform for Advanced Control and Estimation (PACE): Shell's and Yokogawa's next generation advanced process control technology. *IFAC-PapersOnLine*, 48(8):1–5.

Baldea, M. and Daoutidis, P. (2012). *Dynamics and nonlinear control of integrated process systems*. Cambridge University Press.

Barabási, A.-L. (2016). *Network science*. Cambridge University Press.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *J. Stat. Mech. Theor. Exp.*, 2008(10):P10008.

Bristol, E. (1966). On a new measure of interaction for multivariable process control. *IEEE Trans. Autom. Control*, 11(1):133–134.

Carrette, P. (2020). APC technology in Shell: Levaraging technical advancements for increased benefits. In *Texas-Wisconsin-California Control Consortium*.

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

Constantino, P. H. and Daoutidis, P. (2019). A control perspective on the evolution of biological modularity. *IFAC-PapersOnLine*, 52(11):172–177.

Constantino, P. H., Tang, W., and Daoutidis, P. (2019). Topology effects on sparse control of complex networks with Laplacian dynamics. *Sci. Rep.*, 9:9034.

Cott, B. J. (2007). Unit-wide model predictive control with SMOCPro. In *CSChE Annual Conference*. D.G. Fisher Award Keynote Lecture.

Cutler, C. R. and Ramaker, B. L. (1979). DMC – a computer control algorithm. In *AIChE Annual Meeting*.

Daoutidis, P., Tang, W., and Allman, A. (2019). Decomposition of control and optimization problems by network structure: concepts, methods and inspirations from biology. *AIChE J.*, 65(10):e16708.

Daoutidis, P., Tang, W., and Jogwar, S. S. (2018). Decomposing complex plants for distributed control: perspectives from network theory. *Comput. Chem. Eng.*, 114:43–51.

Diehl, M., Bock, H. G., and Schlöder, J. P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM J. Control Optim.*, 43(5):1714–1736.

Garcia, C. E. and Morshedi, A. (1986). Quadratic programming solution of dynamic matrix control (QDMC). *Chem. Eng. Comm.*, 46(1-3):73–87.

Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. *Proc. Natl. Acad. Sci. U.S.A.*, 99(12):7821–7826.

Grosdidier, P. and Morari, M. (1986). Interaction measures for systems under decentralized control. *Automatica*, 22(3):309–319.

Jogwar, S. S. and Daoutidis, P. (2017). Community-based synthesis of distributed control architectures for integrated process networks. *Chem. Eng. Sci.*, 172:434–443.

Leicht, E. A. and Newman, M. E. J. (2008). Community structure in directed networks. *Phys. Rev. Lett.*, 100(11):118703.

Marquis, P. and Broustail, J. P. (1988). SMOC, a bridge between state space and model predictive controllers: application to the automation of a hydrotreating unit. *IFAC Proc. Vol.*, 21(4):37–45.

McAvoy, T. J. (1983). *Interaction analysis: Principles and applications*. ISA.

Michel, A., Miller, R., and Tang, W. (1978). Lyapunov stability of interconnected systems: Decomposition into strongly connected subsystems. *IEEE Trans. Circuit. Syst.*, 25(9):799–809.

Mitrai, I. and Daoutidis, P. (2020). Decomposition of integrated scheduling and dynamic optimization problems using community detection. *J. Process Control*, 90:63–74.

Morari, M., Arkun, Y., and Stephanopoulos, G. (1980). 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.*, 26(2):220–232.

Muske, K. R. and Badgwell, T. A. (2002). Disturbance modeling for offset-free linear model predictive control. *J. Process Control*, 12(5):617–632.

Newman, M. E. J. (2006). Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U.S.A.*, 103(23):8577–8582.

Newman, M. E. J. (2016). Equivalence between modularity optimization and maximum likelihood methods for community detection. *Phys. Rev. E*, 94(5):052315.

Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113.

Ng, C. S. and Stephanopoulos, G. (1996). Synthesis of control systems for chemical plants. *Comput. Chem. Eng.*, 20:S999–S1004.

Pannocchia, G. and Rawlings, J. B. (2003). Disturbance models for offset-free model-predictive control. *AIChE J.*, 49(2):426–437.

Pourkargar, D. B., Almansoori, A., and Daoutidis, P. (2017). Impact of decomposition on distributed model predictive control: A process network case study. *Ind. Eng. Chem. Res.*, 56(34):9606–9616.

Pourkargar, D. B., Moharir, M., Almansoori, A., and Daoutidis, P. (2019). Distributed estimation and nonlinear model predictive control using community detection. *Ind. Eng. Chem. Res.*, 58(30):13495–13507.

Qin, S. J. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Eng. Pract.*, 11(7):733–764.

Rawlings, J. B., Mayne, D. Q., and Diehl, M. M. (2017). *Model predictive control: theory, computation, and design*. Nob Hill Publishing, 2nd edition.

Reichardt, J. and Bornholdt, S. (2006). Statistical mechanics of community detection. *Phys. Rev. E*, 74(1):016110.

Scattolini, R. (2009). Architectures for distributed and hierarchical model predictive control – a review. *J. Process Control*, 19(5):723–731.

Skogestad, S. (2004). Control structure design for complete chemical plants. *Comput. Chem. Eng.*, 28(1-2):219–234.

Stewart, B. T., Wright, S. J., and Rawlings, J. B. (2011). Cooperative distributed model predictive control for nonlinear systems. *J. Process Control*, 21(5):698–704.

Tang, W., Allman, A., Pourkargar, D. B., and Daoutidis, P. (2018a). Optimal decomposition for distributed optimization in nonlinear model predictive control through community detection. *Comput. Chem. Eng.*, 111:43–54.

Tang, W., Babaei Pourkargar, D., and Daoutidis, P. (2018b). Relative time-averaged gain array (RTAGA) for distributed control-oriented network decomposition. *AIChE J.*, 64(5):1682–1690.

Tang, W., Constantino, P. H., and Daoutidis, P. (2019). Optimal sparse network topology under sparse control in Laplacian networks. *IFAC-PapersOnLine*, 52(20):273–278.

Tang, W. and Daoutidis, P. (2018a). Network decomposition for distributed control through community detection in input–output bipartite graphs. *J. Process Control*, 64:7–14.

Tang, W. and Daoutidis, P. (2018b). The role of community structures in sparse feedback control. In *Am. Control Conf. (ACC)*, pages 1790–1795. IEEE.

Tang, W. and Daoutidis, P. (2021). Coordinating distributed MPC efficiently on a plantwide scale: The Lyapunov envelope algorithm. *Comput. Chem. Eng.*, 155:107532.

Tang, W. and Daoutidis, P. (2022). Fast and stable nonconvex constrained distributed optimization: the ELLADA algorithm. *Optim. Eng.*, 23:259–301.

Vidyasagar, M. (1980). Decomposition techniques for large-scale systems with nonadditive interactions: Stability and stabilizability. *IEEE Trans. Autom. Control*, 25(4):773–779.

Šiljak, D. D. (1991). *Decentralized control of complex systems*. Academic Press.

Yang, X. and Biegler, L. T. (2013). Advanced-multi-step nonlinear model predictive control. *J. Process Control*, 23(8):1116–1128.

Yu, C.-C. and Fan, M. K. H. (1990). Decentralized integral controllability and D-stability. *Chem. Eng. Sci.*, 45(11):3299–3309.