Progressive Hedging Decomposition for Solutions of Large-Scale Process Family Design Problems

Georgia Stinchfielda, Jean-Paul Watsonb, Carl D. Lairda\*

a Carnegie Mellon University, Pittsburgh, Pennsylvannia 15201, USA

b Lawrence Livermore National Laboratory, Livermore, California 94550, USA

\*claird@andrew.cmu.edu

Abstract

Rapid, wide-scale deployment of green process systems, such as carbon capture and water desalination systems, is essential for combatting climate change. Methods relying on traditional design or modularity fail to capture the benefits of both economies of numbers and economies of scale. We propose a *process family design* approach, which designs a collection of related processes while simultaneously exploiting opportunities for common elements. In previous work, we explored different optimization formulations to solve this problem. In this work, we develop a decomposition approach to tackle larger problems efficiently. We consider a water desalination case study, which is too large to solve within a reasonable timeframe with the full discretization formulation. We instead exploit the block-angular structure of the full discretization formulation to decompose and solve the problem implicitly using Progressive Hedging (PH). We use the open-source Python package mpi-sppy to execute PH which allows us to leverage parallelization and a HPC cluster to further improve solution time.

**Keywords**: Decomposition, Process Design, Parallelization, Water Desalination.

* 1. Introduction

Some challenges faced in climate change-related process systems engineering rely on rapid, wide-scale, and tailored deployment of a particular chemical process. For example, a promising solution for addressing the rapid depletion of bodies of freshwater is to deploy water desalination systems to purify alternative water sources. Currently, common design approaches either focus on economies of scale (traditional design) or economies of numbers (modular design), but both have their drawbacks. In previous work, we introduced a process family design strategy that combines features of both approaches. We have demonstrated process family design via several case studies (Stinchfield et al., 2023). However, for very large-scale problems, computational time is still prohibitive. In this paper, we develop a decomposition approach based on the Progressive Hedging (PH) algorithm (Rockafellar and Wets, 1991) and demonstrate efficient parallel scalability on high-performance computing architectures using the open-source package mpi-sppy (Knueven et al., 2023).

In past work, we introduced a mathematical model for solving a discretized version of the process family design problem (Stinchfield et al., 2023 and Chen et al. 2022). This model considers two sets of discrete decision variables. The first set of variables selects which unit module designs are included in the process platform from a set of candidate options; the second set of variables determines which of these unit module designs are assigned to each variant. In this work, we exploit a parallelized PH algorithm to solve large-scale process family design problems. PH is a well-known algorithm traditionally used to solve stochastic programming problems. PH works as a heuristic in the context of models with discrete decision variables, but as of recently provides gap-closing capabilities. Further, PH can address discrete variables at any stage in the problem structure. While our problem is not a two-stage stochastic programming problem, it shares a mathematically equivalent block-angular structure, so PH can be directly leveraged. We decompose our problem by process variant, treating the discrete platform unit module design variables as “first-stage” variables (in stochastic programming terminology) and the discrete assignment of unit module designs to variants as second-stage. A large-scale case study on water desalination is then solved via PH using mpi-sppy (Knueven et al., 2023) on a multi-node HPC using Gurobi© as the sub-problem solver (Gurobi Optimization, 2023).

* 1. Background

Our process family design strategy was first inspired by *product* family and platform design, which has documented success in a variety of applications found in the automotive, aircraft, and technology industries (Simpson et al., 2014). Given a set of product variants, product family design creates a platform of common components that are shared across variants, optimizing the remaining elements uniquely. We map this approach to process systems engineering by viewing *product* variants instead as *process* variants; this differs from common design approaches by introducing customization and standardization simultaneously. Traditional engineering aims to exploit economies of scale, with unique designs for each variant. Manufacturing one-off designs is significantly more expensive and time consuming compared to mass manufacturing of standardized components. In contrast, modular design derives cost savings and shortened timelines from economies of numbers through standardization of manufacturing. However, the designs offered are significantly less flexible and typically result in sub-optimal operating conditions. In process family design, we simultaneously design the platform of shared unit modules along with each of the process variants in the family. This approach enables trade-offs between both customization and standardization. In this way, we aim to optimally exploit economies of scale and economies of numbers simultaneously.

* 1. Problem Statement

The goal is to produce a process system design for a set of process variants (a complete set of designs forms the process family, ) using a set of standardized unit module designs (from the process platform, ). Each *process variant* is defined by all requirements the variant must meet along with the defining characteristics of the deployment site. The *process system architecture* defines the necessary *unit module types* required to construct the process system, denoted by the set . For example, a refrigeration system might have a set of unit module types comprised of an evaporator, compressor, condenser, and valve. A unit module type is general; a *unit module type design* specifies all necessary information to build a particular instance of a unit module type. For a variant to have a complete *process variant design*, there must be a fully specified unit module design, , for each unit module type . In addition, the combination of unit module designs must be feasible, meaning it satisfies all requirements and characteristics of the process variant.

The unit module types are separated into the set of common unit module types and unique unit module types (where and ). A *process platform* contains a number of unit module designs for each common unit module type . Building this platform introduces standardization by using common unit module types for each of the process variants. A *process family* is a complete set of process variant designs where each of the common unit module types have designs sourced from the process platform, and the unique unit module types have their respective designs independently determined. The goal is to simultaneously optimize all the process variants, the platform of common unit module designs, and assign common unit module designs for each from the platform for all variants , and determine designs for all unique unit module types for all variants .

* 1. Solution Approach
     1. Optimization Formulation

In previous work, a discretization approach was proposed for this process family design problem (Stinchfield et al., 2023). We develop a set of *candidate unit module designs* for all common unit module types . We identify each candidate unit module design using the label ; the set of all labels is denoted by . For each possible *combination* *of candidate unit module designs* (which is exactly one candidate unit module designfor each) and process variant conditions, we perform an optimization of the system model. This optimization determines the unique unit module designs , operating conditions , feasibility **,** and total annualized cost of the system . The set captures all feasible alternatives for variant . The binary decision variable determines which candidate common unit module types will be included in the platform. A second decision variable indicates whether alternative is selected for variant . The optimization formulation for this approach is shown in (1).

(1a)

(1b)

(1c)

(1d)

(1e)

(1f)

Here, the weight of variant represents the expected sales. The objective (1a) is to minimize the total weighted cost of designing all variants . In (1b), we constrain the number of designs allowable in the platform via (where ) to enforce specified levels of standardization. In (1c), we ensure only one alternative is selected for each variant . Due to favorable properties of the presented formulation, we can relax to be between in (1f), and (under mild assumptions) this variable will converge to binary at optimality due to similarities to the classic -median formulation.

* + 1. Decomposition and Solution Strategy

Given the significant size of the problem, we use PH decomposition, typically for stochastic programming problems, for efficient solution (Rockafellar and Wets, 1991). Stochastic programs have a block angular mathematical structure, which is favorable for decomposition because most of the variables can be separated into disjoint sets of variables, that only appear in the constraints of the associated subproblem . The remaining variables prevent the optimization problem from being perfectly separable as they appear in two or more of the subproblems. To decompose via PH, we create a copy of every complicating variable, denoted as , in each sub-problem . The PH algorithm then iteratively solves subproblems to for all ; these equality constraints are referred to as *non-* *anticipativity* constraints in the explicit (i.e., full) form of the problem. PH exploits this structure by solving the decomposed sub-problems separately (a step that is trivially parallelizable) by first relaxing the non-anticipativity constraints. To ensure all complicating variables are ultimately non-anticipative, PH penalizes deviations from the first-stage variable average at each iteration by incorporating a linear and proximal penalty term into the sub-problem objectives. PH updates a weight vector after solving all sub-problems at iteration that informs the penalization terms of the magnitude of deviation from the disaggregated complicating variables and their average ; represents the probability, or “weight”, of scenario . In addition, PH utilizes the parameter to control the magnitude of penalization for deviations.

While our process family design problem is not a stochastic program, it shares the characteristic block-angular structure. In our case, we form sub-problems corresponding to subsets of the variants . This is feasible because the variable is only dependent on a variant and the associated alternative set . The complicating (i.e., first-stage) variables are , which specify what common unit module designs should be offered in the platform; the values of these variables clearly must be equivalent across subproblems. PH thus allows us to solve smaller subproblems rather than attempting to solve the overall formulation (1) with every variant and candidate unit module design considered concurrently. Additionally, using the open-source Python package mpi-sppy (Knuevenet al., 2023) allows us to perform the PH algorithm in parallel with incumbent finders and lower bounding capabilities.

* 1. Case Study and Results

To demonstrate the proposed decomposition approach, we consider the problem of designing a large family of desalination systems for high salinity produced water. We use an equation-oriented model written in Pyomo (Bynum et al., 2021) that was developed as a part of the PARETO framework (Drouven et al., 2022). This system is made up of a single-effect evaporation unit coupled with a single-stage adiabatic compressor. Produced water enters the shell side of the evaporator to be split into a vapor stream and concentrated brine stream. The adiabatic compressor converts the vapor stream into superheated steam. Exiting the compressor there is a minimum recycle line for consideration of design flow operation. The superheated steam provides the heat of evaporation. Finally, the condensation is removed as a freshwater stream.

For this case study, we define each variant by the flow rate of salt water and the concentration of salt. Salt concentrations of produced water sites across multiple Texas water basins are documented in the PARETO project, and only salt concentrations between 20 g/kg and 150 g/kg were included. Flowrates considered were between 0.1 kg/s and 1 kg/s (55 to 500 bbls/day). Additionally, all the unit module types in the system (evaporator and compressor) were selected to be commonly designed. We selected 20 candidate unit module designs for each common unit module type. For the evaporator, we design for heat exchange area and for the compressor, we design for the flow rate.

We solved this process family design for a desalination system requiring 10,897 variants. We constrained the number of unit module design options available in the platform to three evaporator designs and three compressor designs (i.e., for all ). The optimal platform selected evaporator area designs of , and and compressor flow designs of kg/s, kg/s and kg/s. The combination of common unit module designs assigned to each variant is captured in Figure 1.

A colorful squares with different colored lines

Description automatically generated with medium confidence

Figure 1. Process Family of the Water Desalination Case Study

The selected alternative (i.e., the combination of common unit module designs selected from the platform) is identified by the element in the legend.

Using mpi-sppy (Knueven et al., 2023), this problem was decomposed into 99 subproblems (with approximately 100 variants per subproblem) and solved using 99 parallel processes. We used a default rho value () of 50 and limited Gurobi© to a maximum of 16 threads per process. We calculated lower bounds using the Frank-Wolfe progressive hedging spoke (fwph) and searched for incumbents using the xhatshuffle spoke. Results were obtained on the Quartz HPC cluster at Lawrence Livermore National Laboratory using 30 nodes. Each node has two 18-core Xeon E5-2695 processors (2.1 GHz) and 128 GiB of RAM. We ran PH for a maximum of 25 iterations, which took approximately 5 hours. The algorithm terminated with a 1.399% optimality gap.

* 1. Conclusions

We demonstrated the proposed decomposition approach for solving a large-scale process family design problem on a case study of water desalination for produced water. We were able to design 10,897 process variants simultaneously. For comparison, attempting to solve the same problem using formulation (1) without decomposition achieved a 47.8% gap after six hours of Gurobi© solve time. For future work, PH convergence can be improved by tuning the penalization parameter, . Watson and Woodruff (2011) suggest heuristics for constant values, but they depend on objective costs associated with the complicating variables; our problem does not follow this assumption. Incorporating a gradient-based update to values after each iteration of the algorithm is a promising step that could improve the time needed to reach convergence.

* 1. Disclaimer and Acknowledgements

This project was funded by the U.S. Department of Energy, National Energy Technology Laboratory an agency of the United States Government, through a support contract. Neither the United States Government nor any agency thereof, nor any of their employees, nor the support contractor, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. The NETL support contractor’s contributions to this work were funded by the National Energy Technology Laboratory under the Mission Execution and Strategic Analysis contract (DEFE0025912) for support services. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the Department of Energy’s Office of Electricity’s Advanced Grid Modeling (AGM) program. This effort was partially funded by the U.S. Department of Energy’s Institute for the Design of Advanced Energy Systems (IDAES) supported by the Office of Fossil Energy and Carbon Management’s Simulation-Based Engineering/Crosscutting Research Program.

References

B. Knueven, et al., 2023, “A Parallel Hub-and-Spoke System for Large-Scale Scenario-Based Optimization Under Uncertainty” Mathematical Programming Computation, 15, pp. 591-619.

Gurobi Optimization, LLC, 2023, “Gurobi Optimizer Reference Manual”.

G. Stinchfield, et al., 2023, “Optimization-Based Approaches for Design of Chemical Process Families using ReLU surrogates”, in Proceedings of 2023 Foundations of Computer-Aided Process Operations (FOCAPO) and Chemical Process Control (CPC) Conference

J.P. Watson and D.L. Woodruff, 2011, "Progressive Hedging Innovations for a Class of Stochastic Mixed-Integer Resource Allocation Problems", Computational Management Science (8), pp. 355-370.

M.G. Drouven, A.J. Calderon, M. A. Zamarripa, and K. Beattie, 2022, “Pareto: An Open-Source Produced Water Optimization Framework”, Optimization and Engineering, pp. 1–21.

M.L. Bynum, G.A. Hackebeil, W.E. Hart, C.D. Laird, B.L. Nicholson, J.D. Siirola, J.-P. Watson, and D.L. Woodruff, 2021, “Pyomo – Optimization Modeling in Python”, Springer.

R.T. Rockafellar and R.J. Wets, 1991, “Scenarios and Policy Aggregation in Optimization under Uncertainty”, Mathematics of Operations Research, Vol. 16, pp. 119–147.

T.W. Simpson, J. Jiao, Z. Siddique, and K. Holtta-Otto, 2014, “Advances in Product Family and Product Platform Design”, Springer, Vol. 1.

Z. Chen, et al., 2022, "Optimization-Based Design of Product Families with Common Components", Computer Aided Chemical Engineering, pp. 91-96.