Systems & Control Transactions

Original Research Article
Peer Reviewed Conference Proceeding
ESCAPE 36 - European Symposium on Computer Aided Process Engineering
Sheffield, UK. 21-24 June 2026

PSE PRESS

Decomposition of MINLP Formulations in Process Family Design using Progressive Hedging

Ali Asger[a], Bernard Knueven[b], and Carl Laird[a]*

[a] Carnegie Mellon University, Pittsburgh, PA, USA
[b] National Laboratory of the Rockies, Golden, CO, USA
* Corresponding Author: claird@andrew.cmu.edu.

ABSTRACT

Distributed deployment of process systems can benefit from modularity and shared components across multiple variants, reducing both manufacturing costs and engineering effort. Process family design formalizes this idea by simultaneously optimizing a family of process variants while determining a shared platform of common components. This results in a large-scale mixed-integer nonlinear program (MINLP) that couples nonlinear process models with discrete platform-allocation decisions. In this work, we solve the process family design MINLP using a progressive hedging (PH)-based decomposition strategy that exploits its block-angular structure. To improve convergence for this nonconvex problem, we introduce dynamic gradient-based penalty updates, a decoupled primal-dual strategy via separate PH runs, and parallel optimization-based bounds tightening of first-stage variables. Computational results on a water desalination case study demonstrate that the proposed approach improves solution quality and reduces computational time compared to baseline PH, while achieving a lower total cost of the process family than the previous discretization-based MILP formulation [1]. These results highlight the effectiveness of tailored decomposition strategies for large-scale process family design problems.

Keywords: Process Family Design, Mixed Integer Nonlinear Programming, Progressive Hedging

INTRODUCTION

Increasing demand for energy and escalating resource depletion is going to require rapid and large-scale deployment of emerging process technologies. Existing process design methodologies are largely driven by economies of scale, with each process installation optimized independently to minimize capital and operating costs. While this paradigm is well suited for larger capacity single-site deployment, it becomes less effective for distributed deployment applications, where manufacturing standardization can play a significant role in reducing overall cost and engineering effort. In such contexts, modular design strategies that exploit standardization offer a promising alternative.

Process Family Design [1] has recently been proposed as a systematic optimization-based framework to support such modular design strategies while maintaining flexibility. Rather than optimizing a single instance of a process system, we optimize a family of process variants of a process system while promoting the use of shared components drawn from a common platform. A key feature of the optimization formulation is that the platform itself, i.e. the selection and sizing of shared components, is co-optimized with the individual process variants. This coupling enables trade-offs between individual process variant performance and the overall platform commonality to be explicitly captured within the optimization problem.

The resulting optimization problem is inherently a challenging large-scale mixed-integer nonlinear program (MINLP). Continuous nonlinear constraints arise from the process system model, e.g. process physics and cost correlations, while discrete decisions govern the platform assignment across the family of variants. The complexity arises not only from the nonlinearities within the individual variants but also from the coupling constraints that enforce platform commonality across the family. As the number of variants and potential platform components increases, the platform size grows rapidly, often rendering monolithic global optimization approaches intractable.

In prior work, Stinchfield et al [1] addressed the computational challenge of solving the complex MINLP by discretizing the design space of common modules on the platform. By pre-computing the feasibility of candidate designs and converting the problem into a mixed-integer linear program (MILP), the global optimum of the discretized space could be found efficiently. However, the discretization approach presents several limitations. First, it requires expensive pre-computation to evaluate the Cartesian product of all candidate platform designs across all variants, i.e. simulate each combinatoric instance of discretized candidate designs for each common module on the platform across each variant of the family. Second, the solution quality is limited by the resolution of the discretization of the design space; improving accuracy requires a finer grid, which reintroduces combinatorial explosion.

More recently, a decomposition strategy has been applied to the discretization-based process family design formulation. Stinchfield et al [2] leverage the block-angular structure arising from shared platform variables to decompose the MILP formulation and improve computational tractability. While this approach demonstrates the value of decomposition for large process family design problems, it remains dependent on discretization and the associated pre-computation. In this work, we address these limitations by formulating and solving the process family design problem directly as an MINLP. Building on prior decomposition ideas, we propose a decomposition strategy based on Progressive Hedging (PH) [3], a method used for stochastic programming but well-suited here due to the block-angular structure of the formulation. In contrast to applying PH to the discretization-based MILP, we apply PH directly to the original MINLP by relaxing the complicating platform consensus constraints and penalizing deviations. This enables full decomposition of the family of process variants into smaller subproblems that can be solved independently.

While PH is heuristic when applied to nonconvex problems [3], we demonstrate that a tailored algorithmic setup significantly improves convergence and solution quality. Specifically, we implement a dynamic penalty parameter ρ update scheme that utilizes gradient information at each iteration to drive the variants towards a consensus platform design. This framework is implemented using the open-source package *mpi-sppy* [4], enabling parallel solution of the subproblems. We validate this approach on a desalination process family, comparing the computational performance and solution optimality against both off-the-shelf solvers and the previous MILP discretization method.

BACKGROUND

Various product family design approaches have been extensively investigated across a wide range of industrial applications [4, 5]. In general, optimization approaches for nonlinear product family design have been focused on stochastic methods, e.g. genetic algorithms [7]. Although effective for yielding solutions to challenging problems in practice, there are no guarantees on the quality of solutions obtained.

Relatively few works have utilized mathematical programming techniques to the product family design problem. Khajavirad and Michalek [8] proposed an MINLP-based optimization approach for solving the nonlinear product family design problem, however they considered a fixed platform design. We are not aware of any contribution that directly addresses the MINLP formulation for simultaneous platform and family design.

To address the scalability limits of the process family design MINLP formulation, decomposition methods are required. The process family design problem exhibits a *block-angular* structure analogous to two-stage stochastic programming, where the variant-specific variables are independent ("second-stage"), and the platform-specific variables are coupled ("first-stage") and must be identical across the family ("non-anticipativity").

Progressive hedging (PH) [3] is a horizontal decomposition algorithm that is well-suited for this structure. By relaxing the coupling constraints and penalizing deviations from the group average, PH decomposes the problem into independent subproblems that can be solved in parallel. For a two-stage optimization, the PH algorithm can be stated as described in Algorithm 1.

Algorithm 1: Progressive hedging for two-stage stochastic programming

1 **Initialization**: Let $\nu \leftarrow 0$ and $w^{\nu} \leftarrow 0, \forall s \in \mathrm{S}$. Compute for each $s \in \mathrm{S}$

$$x^{(\nu+1)}(s) \in \operatorname{argmin}_x f_1(x) + f_2(y; \vec{x}(s))$$

where, $f_1(x)$ is the first-stage cost and $f_2(y; \vec{x}(s))$ is the second stage cost.

2 **Iteration Update**: $\nu \leftarrow \nu + 1$

3 **Aggregation**: $\bar{x}^{(\nu)} \leftarrow {\sum_{s\in S} \pi_s x^{(\nu)}(s)}/{\sum_{s\in S} \pi_s}$

4 **Price Update**: Compute for each $s \in \mathrm{S}$

$$w^{(\nu)}(s) \leftarrow w^{(\nu-1)} + \rho[x^{\nu}(s) - \bar{x}^{(\nu)}]$$

5 **Decomposition**: Compute for each $s \in \mathrm{S}$

$$x^{(\nu+1)}(s) \in \operatorname{argmin}_x f_1(x) + f_2\big(y; \vec{x}(s)\big) + w^{(\upsilon)}(s)^{\mathrm{T}} x + \frac{\rho}{2}\left\|x - \bar{x}^{(\nu)}\right\|^2$$

6 **Termination**: If a criterion is met, Stop. Otherwise go to step 2.

The PH algorithm is initialized by independently

solving all the subproblems. Each iteration of the algorithm begins with an aggregation step, in which the probability-based weighted primal mean $\bar{x}^{(\nu)}$ is computed, which provides an estimate of the optimal decision vector that satisfies the non-anticipativity constraints. The associated dual variables $w^{(\nu)}(s)$ act as Lagrange multipliers and are updated using the primal mean $\bar{x}^{(\nu)}$, the current values of the non-anticipative variables $x^{\nu}(s)$, and quadratic penalty parameter ρ. The algorithm subsequently decomposes by subproblem: each subproblem is resolved with its objective function augmented by the updated dual terms and a quadratic penalty term. Termination of the PH algorithm may be based on internal PH criteria, such as relative optimality gap or a specified tolerance on violations of non-anticipativity constraints.

However, while PH is guaranteed to converge for convex problems, it acts as a heuristic for non-convex MINLPs [3]. The efficiency of the PH algorithm convergence is strongly impacted by the choice of the penalty parameter ρ [9]. In particular, the value of the penalty parameter ρ influences both the primal decision variables and the updates of dual weights [10]. In the update of dual price $w^{(\nu)}(s)$ (Step 4), the step size based on deviation from the mean is scaled by ρ. As a result, ρ directly controls the rate at which dual variables progress. In the following decomposition step, for each $s \in \mathrm{S}$, the algorithm minimizes an augmented objective, where the two terms of the augmented part ($w^{(\upsilon)}(s)^{\mathrm{T}}x + \frac{\rho}{2}\left\|x - \bar{x}^{(\nu)}\right\|^2$) exert competing influences during optimization: the dual term ($w^{(\upsilon)}(s)^{\mathrm{T}}x$) drives decision variables towards values favored by dual multipliers, and the quadratic term ($\frac{\rho}{2}\left\|x - \bar{x}^{(\nu)}\right\|^2$) promotes proximity to the mean decision vector. Consequently, varying the penalty parameter ρ alters the trajectory of the primal variables and overall convergence of the algorithm. Larger values of ρ amplify the contribution of the quadratic penalty, leading to aggressive updates to primal convergence, whereas smaller values of ρ yield a smoother and gradual evolution of dual variables, which may be preferable in contexts such as lower bound computation.

Setting a standard static penalty parameter across first-stage variables often leads to slow convergence or cycling in non-convex applications. Watson and Woodruff [9] proposed a variable-specific heuristic for calculation of the penalty parameters for each variable based on their unit-cost, i.e. coefficient in the objective term. They further noted that per-iteration ρ values may be more appropriate for some problems. Building on this work, Naepels and Woodruff [10] combined the heuristic with a gradient-based approach to calculate ρ values and introduced a method to iteratively compute variable-dependent ρ values based on a first-order condition, allowing updates as the algorithm proceeds.

In this work, we build upon these penalty parameter ρ selection strategies within the PH framework. While prior approaches compute gradients only at the initial iteration under the assumption that the relevant objective terms are linear, and assume the availability of a feasible solution, either from an incumbent or starting point that must be specified by the user a priori in existing *mpi-sppy* implementations, we instead address nonlinear objectives by computing gradients at every iteration using the current primal iterates and then switching to incumbents as they are found. Moreover, motivated by challenges posed by the nonlinear nonconvex nature of the process family design MINLP, we extend these ideas by developing decoupled primal-dual penalty parameter ρ strategies that address the interactions between primal and dual convergence, leading to improved overall PH convergence.

PROBLEM SETUP

We consider the process family design problem as formalized by [1]. In this section, we explain how the process family design problem can be formulated as a nonlinear generalized disjunctive program (GDP) and subsequently transformed into a mixed integer nonlinear program (MINLP).

Problem Description

The goal of process family design is to simultaneously design a large number of variants of a given process system that can be deployed across a wide range of installations. For example, deploying a carbon capture system at multiple industrial sites with varying flue gas sources. While the general flowsheet configuration of the carbon capture process system will remain the same at each site, the design requirements for each site (e.g. capture rate, flue gas concentration) can be different. Thus, every variant $v \in V$ will be designed to meet specific requirements; parameterized in $\boldsymbol{r}_v$.

A process system is defined by all the *unit module types* necessary to build an instance of the process; defined by $m \in M$. The unit module types could include absorber, heat exchanger, regenerator, etc. Each variant $v \in V$ requires exactly one of each unit module type $m \in M$. The *unit module type design* for each unit module type m and each variant is defined by variable vector $\boldsymbol{d}_{v,m}$.

To exploit opportunities for shared components (and engineering design effort) we consider a platform of common unit modules that can be shared across the family of process variants. The unit module types M are separated into *commonly design unit modules* (set C included on the platform) and *uniquely designed unit modules* (set U), where sets C and U are disjoint ($C \cap U = \emptyset$) and their union gives the set M ($C \cup U = M$). The common designs making up the platform $\mathcal{P}$ are indexed by label

$l \in L_c$ to capture different candidate designs for each common module type $c \in C$. The design specifications for these common unit modules on the platform are captured by $\hat{\boldsymbol{d}}_{c,l}$. All unique unit module types $u \in U$ are designed variant specific.

Problem Formulation

The process system design of every single variant $v \in V$ with different set of design requirements $\boldsymbol{r}_v$ is governed by a set of nonlinear (in most applications) equations that describe the cost, physics, and performance of the process system. Our objective is to minimize the overall annualized cost of the process family while simultaneously determining (a) a platform with optimal designs of common unit module types for variants to choose from, (b) the distribution of these common unit module designs among the family of process variants, and (c) the variant-specific design variables for modules not in the platform and the optimal operating conditions for each of the variants.

Thus, the process family design problem can be formulated as a nonlinear generalized disjunctive program (GDP) that can be transformed into a mixed integer nonlinear program (MINLP):

$$\min_{p,i,y,d,\hat{d},o} \sum_{v \in V} w_v p_v \tag{1}$$

$$\text{s.t.} \quad p_v = f_v^p(\boldsymbol{r}_v, \boldsymbol{d}_v, \boldsymbol{o}_v) \;\; \forall\, v \in V \tag{2}$$

$$\boldsymbol{i}_v = f_v^i(\boldsymbol{r}_v, \boldsymbol{d}_v, \boldsymbol{o}_v) \;\; \forall\, v \in V \tag{3}$$

$$0 = h_v(\boldsymbol{r}_v, \boldsymbol{d}_v, \boldsymbol{o}_v) \;\; \forall\, v \in V \tag{4}$$

$$\bigvee_{l \in L_c} \begin{bmatrix} Y_{v,c,l} \\ \boldsymbol{d}_{v,c} = \hat{\boldsymbol{d}}_{c,l} \end{bmatrix} \;\; \forall\, v \in V, c \in C \tag{5}$$

$$\hat{\boldsymbol{d}}_{c,l-1} \le \hat{\boldsymbol{d}}_{c,l} \quad \forall\, c \in C, l \in L_c \backslash \{1\} \tag{6}$$

$$\boldsymbol{d}_v^L \le \boldsymbol{d}_v \le \boldsymbol{d}_v^U \quad \forall\, v \in V \tag{7}$$

$$\hat{\boldsymbol{d}}_{c,l}^L \le \hat{\boldsymbol{d}}_{c,l} \le \hat{\boldsymbol{d}}_{c,l}^U \quad \forall\, c \in C, l \in L_c \tag{8}$$

$$\boldsymbol{o}_v^L \le \boldsymbol{o}_v \le \boldsymbol{o}_v^U \quad \forall\, v \in V \tag{9}$$

$$\boldsymbol{i}_v^L \le \boldsymbol{i}_v \le \boldsymbol{i}_v^U \quad \forall\, v \in V \tag{10}$$

$$Y_{v,c,l} \in \{True, False\} \;\; \forall\, v \in V, c \in C, l \in L_c \tag{11}$$

The objective in (1) minimizes the weighted annualized cost aggregated over all variants $v \in V$. Constraints (2-4) embed, for each variant, the process model equations, including the cost (p_v), performance indicators ($\boldsymbol{i}_v$), and process system equations as functions of the design requirements $\boldsymbol{r}_v$, design variables $\boldsymbol{d}_v$ and operating variables $\boldsymbol{o}_v$. The formulation introduces commonality across variants through explicit logical constraints governing the selection of shared unit module designs. Constraint (5) employs disjunctions and associated Boolean variables $Y_{v,c,l}$ to assign common unit modules to variants by enforcing equality among the relevant design variables. Constraint (6) enforces a size-based ordering of the platform-level module designs as a symmetry-breaking constraint. Constraints (7 - 10) set bounds on the design variables $\boldsymbol{d}_v$, common designs $\hat{\boldsymbol{d}}_{c,l}$, operating variables $\boldsymbol{o}_v$., and performance indicators $\boldsymbol{i}_v$. Constraint (11) specifies the binary nature of the assignment variables.

PROGRESSIVE HEDGING SETUP

Problem Decomposition

Decomposition algorithms involve splitting a large problem into smaller problems and solving them iteratively until a convergence criteria is met. Stochastic programs inherently possess a block-angular structure that allows for the decomposition of a problem into smaller sub-problems. In this structure, first-stage variables link the subproblems through non-anticipativity constraints, while second-stage variables are specific to each subproblem.

Our problem can be represented with a similar block-angular structure, as shown in Figure 1. We decompose our problem into subproblems, each representing a smaller process family design problem, with the constraint that all subproblems share the same platform design. Therefore, our first-stage variables are continuous common unit module design variables in the platform ($\hat{\boldsymbol{d}}_{c,l}$) and the second-stage variables are all the local variables associated with each process family design subproblem ($p_v, \mathbf{i}_v, \mathbf{r}_v, \mathbf{d}_{v,m}, \mathbf{o}_v, \mathrm{Y}_{v,c,l}$).

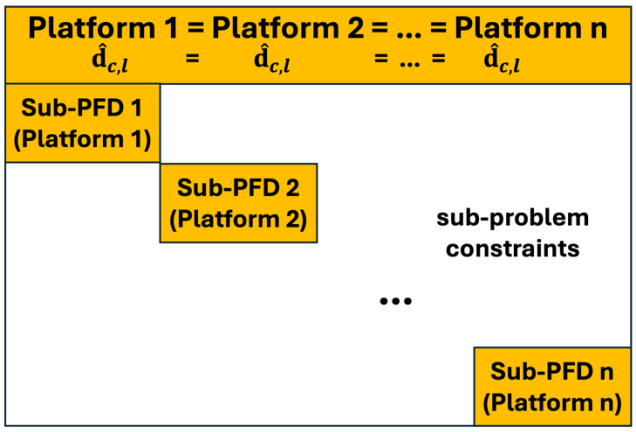

Figure 1. Block-angular structure of the process family design problem

Dynamic Gradient-based ρ Updates

The convergence of PH for non-convex MINLPs is highly sensitive to the penalty parameter ρ. A static global ρ is often insufficient: values that are too small result in slower convergence and failure to enforce consensus, while values that are too large cause the algorithm to cycle or lock into suboptimal local minima. To address this, we implement a dynamic, variable-specific update

method based on the first-order optimality conditions, as proposed by Naepels and Woodruff [10]:

$$\rho^{\nu}(s) = {[c_D - w^{(\nu)}(s)]}/{[x^{(\nu)}(s) - \bar{x}^{(\nu)}]} \quad (12)$$

with $c_D = -\nabla f(\hat{x})$

However, the effectiveness of this update rule depends critically on the availability of accurate gradient information. Naepels and Woodruff [10] compute gradients only at the first iteration under the assumption of a linear objective, and thus the gradients remain constant throughout the PH iterations. In contrast, the process family design MINLP involves nonlinear terms in the objective, for which gradient information varies with iterations. Consequently, restricting gradient evaluation to the initial iteration can lead to inaccurate updates and degraded convergence behaviour. To address this limitation, we compute gradients dynamically at every PH iteration using the current primal iterates. Specifically, the gradient term c_D in (12) is evaluated as $c_D = -\nabla f(x^{(\nu)})$, until a feasible incumbent solution becomes available. Once an incumbent is found, gradient evaluations are switched to the incumbent solution to stabilize updates and reduce oscillations in later iterations. This dynamic gradient evaluation enables penalty parameter ρ to adapt to local curvature in the nonlinear objective, improving convergence and robustness of PH for non-convex MINLPs.

While dynamic gradient-based updates provide a well-motivated mechanism for adapting the penalty parameter ρ, it also introduces increased sensitivity to primal convergence. In practice, the non-anticipative variables $x^{\nu}(s)$ may rapidly approach their consensus values during early PH iterations, after which convergence slows. In such cases, the difference $|x^{\nu}(s) - \bar{x}^{(\nu)}|$ becomes very small, causing the update in (12) to produce disproportionately large values of ρ. Overly large values of ρ are ineffective and can stall convergence in both primal and dual variables. To prevent this behaviour, Naepels and Woodruff [10] also provide explicit criteria to determine when ρ should be updated based on measures of primal and dual convergence.

Decoupled Primal-Dual Strategy

The dynamic gradient-based updates described above yields subproblem-dependent penalty parameters $\rho^{\nu}(s)$, which must be aggregated into a single subproblem-independent value to apply progressive hedging. Consequently, a rule is needed to reconcile the subproblem-specific values of $\rho^{\nu}(s)$ computed from (12).

Naepels and Woodruff [10] examine several aggregation strategies for subproblem-dependent penalty parameters, including minimum, average, and maximum values across subproblems. Consistent with preceding discussion, different aggregation rules place different emphasis on enforcing primal consensus versus stabilizing dual updates. Building on this observation, we adopt a decoupled primal-dual strategy by solving two progressive hedging procedures with distinct aggregation rules for the penalty parameter $\rho^{\nu}(s)$.

In the first PH procedure, the aggregation of the subproblem-dependent penalty parameters is chosen to promote primal consensus. Specifically, an average aggregation rule is applied,

$$\rho^{\nu}_{primal} = {\sum_{s\in S} \rho^{\nu}(s)}/{|S|} \quad (13)$$

which smooths subproblem-to-subproblem variability while maintaining sufficient penalty magnitude to enforce agreement among the nonanticipative variables. This configuration prioritizes convergence of primal decision variables.

In the second PH procedure, the aggregation rule is selected to favor stable dual behaviour. In this case the penalty parameter is defined using the minimum positive values across subproblems,

$$\rho^{\nu}_{dual} = \min\{\rho^{\nu}(s): \rho^{\nu}(s) > 0, s \in S\} \quad (14)$$

This choice limits the magnitude of the multiplier updates and mitigates oscillations in the dual variables, particularly in later iterations when subproblem solutions are near consensus.

The two PH procedures are executed independently, each using a single, subproblem-independent penalty parameter per iteration, as required by the progressive hedging framework. Together, these complementary strategies provide a practical mechanism for improving convergence behaviour in the presence of nonconvexities and discrete decisions. Although this approach does not alter the fundamental structure of the PH algorithm, it allows different aggregation strategies to be exploited. In the results presented later, this decoupled strategy is shown to improve convergence relative to a single PH configuration.

Optimization-Based Bounds Tightening for First-Stage Variables

Bounds tightening is a critical refinement technique to strengthen relaxations and improve convergence used for solution of MINLPs [11]. Among the available approaches, optimization-based bounds tightening (OBBT) is particularly effective, albeit computationally expensive. In this work, we exploit the two-stage structure of the problem and the sensitivity of progressive hedging convergence to the bounds of the first-stage (nonanticipative) variables by applying OBBT exclusively to these variables as a pre-solve routine.

The OBBT procedure is applied to the first-stage variables independently for each subproblem. For a given first-stage variable, the lower and upper bounds are tightened by solving the auxiliary optimization problems

that minimize and maximize the variable subject to the subproblem constraints. These bound-tightening problems are solved in parallel across all subproblems, exploiting the same decomposition structure as the PH framework.

Following the solution of the OBBT subproblems, the resulting bounds are aggregated by selecting the tightest valid bounds across the subproblems. Specifically, for each first-stage variable, the updated lower bound is taken as the maximum of the subproblem-specific lower bounds, and the updated upper bound is taken as the minimum of the subproblem-specific upper bounds. This aggregation ensures that the tightened bounds remain feasible for all subproblems while reducing the feasible region of the shared variables.

The bound-tightening process is applied iteratively. After each round of OBBT, the aggregated bounds are updated, and the procedure is repeated until successive bound improvements fall below a specified tolerance. Once the updates become insignificant, the tightened bounds are fixed and used to start the PH algorithm.

By performing OBBT in parallel and restricting it to the first-stage variables, the additional computational overhead remains minimal, while accelerating convergence of PH.

CASE STUDY

We now discuss the water desalination case study and the specific process family design problem we used to demonstrate our optimization approach. The water desalination process system model was developed under the PARETO framework [12].

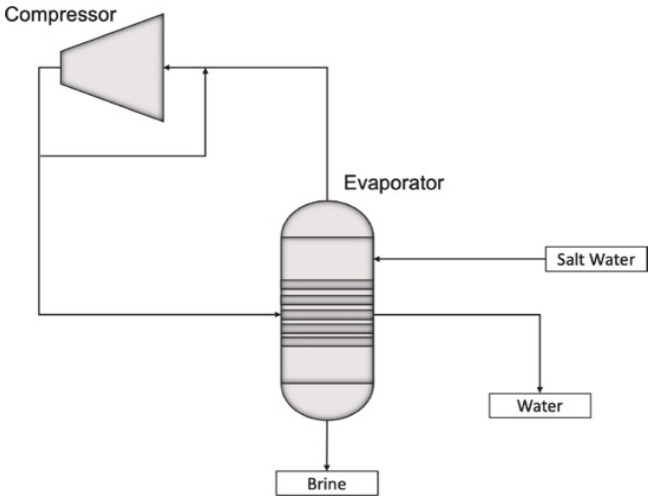

Figure 2. Water Desalination System

Around 60 million barrels of produced water per day is expected from oil and gas fracking across the United States by 2030, which will require treatment of produced water (i.e. desalination) at scale [12]. The salinity and flowrate of the produced water have high variability across wells in different geographical regions. One potential desalination technology is to first evaporate, and then mechanically recompress the vapor [13]. In our case study, we consider a desalination system with a single-effect evaporator and a single-stage adiabatic compressor as shown in Figure 2.

Building on the case study from previous work [1], we consider the process family design problem that accommodates a range of varying produced water feed conditions across the Midland, Texas area. Therefore, the set of process variants $v \in V$ is defined by different combinations of feed flowrates and feed salt concentrations. In this case study, we consider desired specifications of feed salt concentration between 20g/kg and 150g/kg and feed flowrate between 0.1 kg/s and 1 kg/s. Overall, a set of 76 process variants is considered for this process family design problem.

The compressor and the evaporator are the common unit module types to be designed in the platform, i.e. M = C = [compressor, evaporator], with design variables of compressor design flow (between 0.05 kg/s and 1 kg/s) and evaporator area (between 20 m^2 and 400 m^2).

RESULTS & DISCUSSIONS

This section presents the results of the case study introduced in the preceding section by employing the decomposition approach and the algorithmic methodology described in the Progressive Hedging Setup section. The key findings derived from these results are subsequently discussed.

The process family was designed using the optimization formulation described in (1-11). The full process family design problem of 76 variants was decomposed into 19 smaller process family subproblems with 4 variants each. We consider a platform specification that allows 3 designs, i.e. $|L_c|$ = 3, for both the compressor and evaporator. The variants were ordered by size ($\boldsymbol{r}_v$) and bundled into subproblems.

Figure 3 describes which combinations of compressor and evaporator designs offered in the platform $\mathcal{P}$ are assigned to each variant $v \in V$. The x-axis corresponds to the feed flow rate that describes a particular variant v; the y-axis corresponds to the concentration of salt in the feed at variant v.

All the computational studies were performed on the TRACE HPC cluster at Carnegie Mellon University using a single node. The node has two 64-core AMD EPYC 7713 processors (2 GHz) and 2 TB RAM. To ensure consistency in our results, we limit Gurobi© to a maximum of 8 threads.

Table 1 compares the performance of three solution strategies for the water desalination case study: (1) progressive hedging with $\hat{x}$-based gradient penalty update

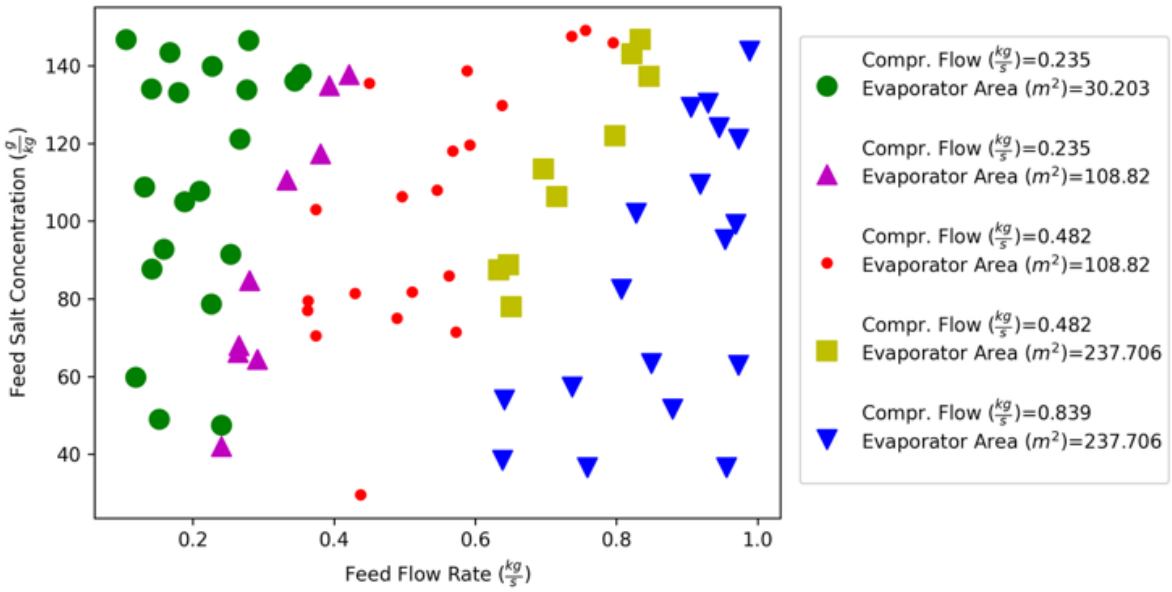

Figure 3. Water Desalination Process Family

following Naepels and Woodruff [10], (2) the proposed progressive hedging approach with dynamic $x^{(\nu)}$-based gradient penalty updates, optimization-based bounds tightening, and a decoupled primal-dual strategy, and (3) a direct MINLP solution using Gurobi© 13. For the progressive hedging-based approaches, the algorithm was run for 50 PH iterations.

Table 1: Comparison of different solution strategies on the water desalination case study.

Solution Strategy	Incumbent Objective ($M)	Dual Bound ($M)	Solution Time (secs)
PH (Grad $\rho, \hat{x}$-based) [10]	10, 131.93 8850.26	6754.36 7117.96	5372 4435
PH (Grad ρ, OBBT, decoupled $x^{(\nu)}$-based) [This work]			
Gurobi© 13 MINLP	9062.82	7744.95	7200

The baseline PH approach yields the highest incumbent objective value and the weakest dual bound, reflecting the limitations of $\hat{x}$-based gradient penalty updates in nonconvex settings. In contrast, the proposed PH configuration achieves a substantially lower incumbent objective value with reduced solution time, while also producing a stronger dual bound. The improvement in the dual bound is consistent with the intended effect of the decoupled primal-dual strategy, which stabilizes dual multiplier updates while maintaining effective progress in the primal iterates.

The direct MINLP approach using Gurobi© 13 produces a competitive incumbent solution with the imposed 7200 s time limit. However, the proposed PH method attains a better incumbent solution in a shorter computation time. The dual bound reported by Gurobi© 13 is included for context. Overall, the results indicate that the proposed PH-based approach provides an effective balance between primal solution quality, dual bound strength, and computational effort for large-scale process family design problems.

CONCLUSION

This paper investigated the direct solution of process family design problems formulated as large-scale mixed-integer nonlinear programs using a progressive-hedging-based decomposition strategy. To improve the practical performance of progressive hedging for nonconvex MINLPs, dynamic gradient-based ρ updates, a decoupled primal-dual strategy implemented through separate PH configurations, and optimization-based bounds tightening for first-stage variables were introduced. Computational results on a water desalination case study demonstrate that the proposed approach achieves improved primal solution quality, stronger dual bounds, and reduced computational time relative to a baseline progressive hedging implementation, while

achieving an approximately 2.5% reduction in total cost of process family compared to the discretized MILP formulation of Stinchfield et al [1].These results highlight the effectiveness of tailored decomposition strategies for large-scale structured nonconvex MINLPs that are challenging for off-the-shelf solvers. Future work will explore hybrid decomposition strategies that incorporate branching in the space of first-stage variables.

ACKNOWLEDGEMENTS

This work was authored in part by the National Laboratory of the Rockies for the U.S. Department of Energy (DOE), operated under Contract No. DE-AC36-08GO28308. This work was supported by the Laboratory Directed Research and Development (LDRD) Program at the National Laboratory of the Rockies. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

AUTHOR IDENTIFIERS

Author ORCIDs:
Ali Asger: 0000-0003-4434-1141
Bernard Knueven: 0000-0002-3694-6274
Carl Laird: 0000-0001-8430-1561

REFERENCES

1. Stinchfield G, Morgan JC, Naik S, Biegler LT, Eslick JC, Jacobson C, Miller DC, Siirola JD, Zamarripa M, Zhang C, Zhang Q, Laird CD. A mixed integer linear programming approach for the design of chemical process families. Computers & Chemical Engineering 183:108620 (2024). https://doi.org/10.1016/j.compchemeng.2024.108620
2. Stinchfield G, Watson, JP, & Laird CD. Progressive hedging decomposition for solutions of large-scale process family dedign problems. Comput Aided Chem Eng 53:1285-120 (2024) https://doi.org/10.1016/B978-0-443-28824-1.50215-5
3. Rockafellar RT, Wets RJB. Scenarios and policy aggregation in optimization under uncertainty. Mathematics of OR 16:119-147 (1991). https://doi.org/10.1287/moor.16.1.119
4. Knueven B, Mildebrath D, Muir C, Siirola JD, Watson JP, Woodruff DL. A parallel hub-and-spoke system for large-scale scenario-based optimization under uncertainty. Math. Prog. Comp. 15:591-619 (2023). https://doi.org/10.1007/s12532-023-00247-3
5. Simpson TW, Siddique Z, Jiao JR. Product platform and product family design. Springer US (2006). https://doi.org/10.1007/0-387-29197-0
6. Gonzalez-Zugasti JP. Models for platform-based product family design. PhD thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology (2000)
7. D'Souza B, Simpson? TW. A genetic algorithm based method for product family design optimization. Engineering Optimization 35:1-18 (2003). https://doi.org/10.1080/0305215031000069663
8. Khajavirad A, Michalek JJ. A decomposed gradient-based approach for generalized platform selection and variant design in product family optimization. Journal of Mechanical Design 130: (2008). https://doi.org/10.1115/1.2918906
9. Watson JP, Woodruff DL. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. Comput Manag Sci 8:355-370 (2010). https://doi.org/10.1007/s10287-010-0125-4
10. Naepels U, Woodruff DL. Gradient-based rho parameter for progressive hedging. Optimization Online, Aug. 2023.
11. http://optimization-online.org/2023/08/gradient-based-rho-parameter-for-progressive-hedging/
12. Laird C, Bynum M, Castillo A, Knueven B, Siirola J. Decomposing optimization-based bounds tightening problems via graph partitioning.. Proposed for presentation at the 2020 INFORMS Annual Meeting. : (2020). https://doi.org/10.2172/1826429
13. http://optimization-online.org/?p=16954
14. Drouven MG, Caldéron AJ, Zamarripa MA, Beattie K. PARETO: an open-source produced water optimization framework. Optim Eng 24:2229-2249 (2022). https://doi.org/10.1007/s11081-022-09773-w
15. Onishi VC, Carrero-Parreño A, Reyes-Labarta JA, Fraga ES, Caballero JA. Desalination of shale gas produced water: a rigorous design approach for zero-liquid discharge evaporation systems. Journal of Cleaner Production 140:1399-1414 (2017). https://doi.org/10.1016/j.jclepro.2016.10.012