Solution Strategies for Integrated Distribution, Production, and Relocation Problems Arising in Modular Manufacturing

R. Cory Allen[1,2], Styliani Avraamidou[2,3], Sergiy Butenko[4], and Efstratios N. Pistikopoulos[1,2,*]

[1]Artie McFerrin Department of Chemical Engineering, Texas A&M University, College Station, TX 77843, United States

[2]Texas A&M Energy Institute, Texas A&M University, College Station, TX 77843, United States

[3]Department of Chemical and Biological Engineering, University of Wisconsin - Madison, Madison, WI 53706, United States

[4]W. Michael Barnes Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77843, United States

[*]Corresponding author: stratos@tamu.edu

Abstract

Recently, there has been a paradigm shift by certain energy and chemical companies towards modular manufacturing, whereby transportable modular production units can be relocated between production facilities to meet the spatial and temporal changes in the availabilities, demands, and prices of the underlying commodities. We refer to the optimal distribution, production, and storage of commodities, and the relocation and operation of the modular production units as the *dynamic multiple commodity supply chain problem with modular production units*. To this end, we present a "flow-based" and a "path-based" mixed-integer linear programming formulation to model the problem. In an effort to solve large-scale instances of the problem, we propose an iterative three-stage matheuristic for the "path-based" formulation. In the first stage of the matheuristic, a feasible solution to the problem is generated by an integrated column generation and Lagrangian relaxation based heuristic. In the second stage of the matheuristic, a path-relinking procedure is utilized as a local search heuristic to further improve the solution. And in the final stage of the matheuristic, the Lagrangian multipliers are updated via a subgradient method. The effectiveness of the matheuristic is illustrated through numerical experiments with a set of randomly generated test instances. For the large-scale test instances, the results show that the matheuristic produces quality solutions orders of magnitude faster than a "state-of-the-art" mixed-integer linear programming solver.

Keywords Production; Scheduling; Heuristics; Large scale optimization

1 Introduction

Traditionally, energy and chemical companies have relied upon "the bigger it is, the better it is" approach for the design of their supply chains [Arora et al., 2020]. This design approach was driven by "economies of scale", which dictates that the capital and operational cost per unit of a commodity produced by a production facility is inversely related to its production capacity. Consequently, this meant that large centralized production facilities were favored over smaller decentralized production facilities [Allen et al., 2018; Baldea et al., 2017]. However, recently there has been a push by energy and chemical companies towards decentralized supply chains whose production facilities are designed and operated utilizing a building block approach [Allman et al., 2021; Bielenberg and Palou-Rivera, 2019; Roy, 2017].

In this building block approach, production facilities are comprised of transportable modular production units, that when chained together can act as a complex production unit, which can be constructed, deconstructed, and rearranged. The underlying technologies of these modular production units need not be the same; thereby, allowing the materials consumed and produced as well as the conversion factors to vary across the different types of modular production units. Moreover, these modular production units can be relocated between different locations in the decentralized supply chain to absorb the spatial and temporal changes in availabilities, demands, and prices of the underlying commodities [Allman and Zhang, 2020; Becker et al., 2019; Bhosekar and Ierapetritou, 2021; Shao and Zavala, 2020]. We refer to the optimal distribution, production, and storage of commodities, and the relocation and operation of modular production units as the *dynamic multiple commodity supply chain problem with modular production units* (DMC-MPU), to be formally defined in Section 2. One of the most similar problems in the literature was presented by Becker et al. [2019]; however, in their problem they do not address material storage decisions and do not allow intermediate materials to be passed between production facilities.

To determine the optimal design and operation of these types of decentralized supply chains, practitioners typically map the problem to its corresponding mathematical programming formulation. Depending on the types of decisions made and the types objective function/s utilized by the practitioner, it can be cast as a linear programming (LP) problem, non-linear programming (NLP) problem, pure integer programming (IP) problem, a mixed integer linear programming (MILP) problem, or a mixed integer non-linear programming (MINLP) problem [Alarcon-Gerbier and Buscher, 2022]. The decisions in these problems can typically be divided into two different sets: (i) the first set, typically revolves around production and relocation decisions for the modular production units and are typically modeled utilizing some type of "flow-based" formulation; (ii) while the second set, typically involves decisions regarding the amount of material consumed and produced by the modular production units and the manner in which material is distributed between and within production facilities [Allen et al., 2020; Allman and Zhang, 2020; Becker et al., 2021; Punyim et al., 2022; Shehadeh, 2020]. For a detailed review the applications of modular production units please see [Alarcon-Gerbier and Buscher, 2022; Baldea et al., 2017; Becker et al., 2021].

While "state-of-the-art" MILP solvers, such as CPLEX and Gurobi, can often solve small scale instances of these types of problems, they begin to struggle as the size of the problem

increases [Cplex, 2009; Gurobi Optimization, LLC, 2022]. This is attributed to the fact that the associated optimization problem integrates a production scheduling problem that includes distribution and storage decisions for commodities with a dynamic facility location problem, both of which are known NP-hard problems [Allman and Zhang, 2020; Jena et al., 2017]. Moreover, since the modular production units can have different nameplate capacities and underlying technologies, the difficulty of the problem is further exacerbated, due to the shear number of layout combinations of the modular production units at the production facilities. To address the computational complexity of the larger scale instances that typically occur in industry, practitioners have begun to utilize various types of heuristics to generate good feasible solutions, such as evolutionary heuristics, Lagrangian based heuristics, linear relaxation heuristics, and local search heuristics [Jena et al., 2017; Punyim et al., 2022; Silva et al., 2020].

To the authors knowledge, all of the mathematical programming formulations for modeling relocation decisions for modular production units in the literature rely on "flow-based" formulations similar to those put forth in the vehicle and production routing literature [Toth and Vigo, 2002; Adulyasak et al., 2015]. Unfortunately, as the number of production facilities increase in the supply chain, the number of binary variables required to model these relocation decisions increases exponentially. This in turn, can cause computational difficulties for even solving the LP relaxation at the root node of the branch and bound tree [Allen et al., 2023; Jena et al., 2017]. To address this issue for routing problems, practitioners typically recast the mathematical programming model "flow-based" formulation to spatially and temporally track the locations of the vehicles to a "path-based" formulation. In theory this dramatically increases the number of binary variables in the mathematical programming problem; however, in practice when column generation generation is utilized this is not often the case [Allman and Zhang, 2020; Barnhart et al., 1998; Mourgaya and Vanderbeck, 2007]. Moreover, it also for the use of column generation based matheurstics to expeditiously explore the solution space for the routing decisions for the vehicles [Archetti and Speranza, 2014; Beheshti and Hejazi, 2015].

To address these aforementioned issues, the contributions in this work are as follows:

i. we create two MILP formulations to solve the DMC-MPU, which utilize "flow-based" and "path-based" approaches respectively;
ii. we present a column generation procedure to expeditiously explore the solution space of the "path-based" MILP formulation, which includes a bespoke dynamic programming problem to solve the column generation subproblems;
iii. we present a Lagrangian relaxation procedure to further decompose the "path-based" formulation;
iv. we present an iterative three-stage matheuristic to generate feasible solutions to the DMC-MPU, which includes a path-relinking procedure to improve locally optimal solutions;
v. we illustrate the effectiveness of the matheuristic through the use of numerical experiments on a set of randomly generated test instances – the test instances were generated in such a way as to minimize the bias in the creation to ensure a fair comparison between the matheuristic and a "state-of-the-art" MILP solver solving the traditional "flow-based" formulation.

The outline of the paper is as follows. In Section 2, the problem definition of the DMC-

MPU is given. In Section 3, the "flow-based" MILP formulation and the "path-based" MILP formulation of the DMC-MPU is given. In Section 4, we present an integrated column generation and Lagrangian relaxation procedure to solve the LP relaxation of the "path-based" MILP formulation. In Section 5, the iterative three-stage matheuristic is presented for solving the DMC-MPU. In Section 6, the results to the numerical experiments are presented. In Section 7, we give our concluding remarks.

2 Problem Definition

Consider a central planner who is trying to optimally operate a multi-commodity supply chain with an underlying distribution system over a discretized time horizon, $\mathcal{T}$. The locations in the supply chain can function as commodity sinks, commodity sources, production facilities, warehouses, or some combination thereof. The production facilities in the supply chain are comprised of transportable modular production units, $\mathcal{K}$. These modular production units can be transferred between production facilities at discrete points in the time horizon and can utilize different underlying technologies to produce or consume different commodities. It should be noted, that the modular production units can have differing nameplate capacities and have underlying technologies, which in turn affect their input raw materials, their output products, and the respective conversion factors. This in turn, allows the type and maximum amount of commodity, $c \in \mathcal{C}$, consumed and/or produced at the production facilities, to evolve throughout the discretized time horizon.

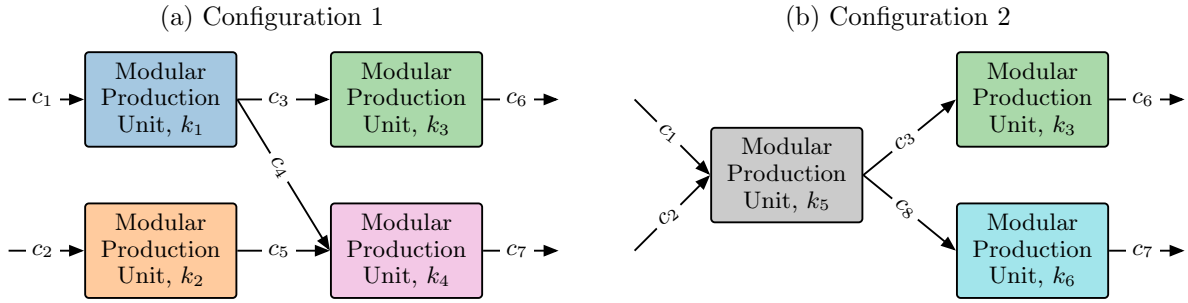

Figure 1: Two different configurations of modular production units at a production facility.

Illustrative Example 1 *To elucidate the concept of modular manufacturing, assume the central planner has access to six modular production units,* $\{k_1, k_2, \ldots, k_6\}$*, with known conversion factors, nameplate capacities, and operating costs. The central planner would like to utilize a subset of these modular production units at a production facility to produce the products,* $\{c_6, c_7\}$*, via the raw materials,* $\{c_1, c_2\}$*. Figure 1 is an illustration of two possible layouts that the central planner could employ at the production facility. In the first layout, Fig. 1a, four modular production units,* $\{k_1, k_2, k_3, k_4\}$*, are utilized, while in the second layout, Fig. 1b, three modular production units,* $\{k_3, k_5, k_6\}$*, are utilized. Depending on the conversion factors, nameplate capacities, operating of the modular production units, commodity availabilities, commodity demands, etc. the first configuration may be preferable to the second or vice-versa.*

The superstructure of the supply chain can be mapped to a fully connected directed graph,

$G_1(V_1, E_1)$. The vertices, V_1, represent the locations in the supply chain and the edges, E_1, indicate the pathways for the shipment of commodities between locations. The vertices, $V_1(c) \subseteq V_1$, indicate the locations in the supply chain that allow a specific commodity, $c \in \mathcal{C}$, and the edges, $E_1(c) \subseteq E_1$, represent the locations that a specific commodity, $c \in \mathcal{C}$, can be transported "from" and "to".

The manner in which the modular production units can spatially and temporally traverse through the supply chain can be captured through the use of the directed acyclic graph, $G_2(V_2, E_2)$. The vertices in the graph are tuples that indicate the facilities and the corresponding time periods that a modular production unit can be located at. The edges in the graph represent the modular production units ability to spatially and temporally traverse the supply chain. For each modular production unit, $k \in \mathcal{K}$, there is a subgraph, $G_2(V_2(k), E_2(k))$, that captures the modular production unit's ability to traverse the graph. The vertices, $V_2(k) \subseteq V_2$, and edges, $E_2(k) \subseteq E_2$, in the subgraph are based upon the initial location of the modular production unit and the time it takes the modular production unit to traverse between two production facilities.

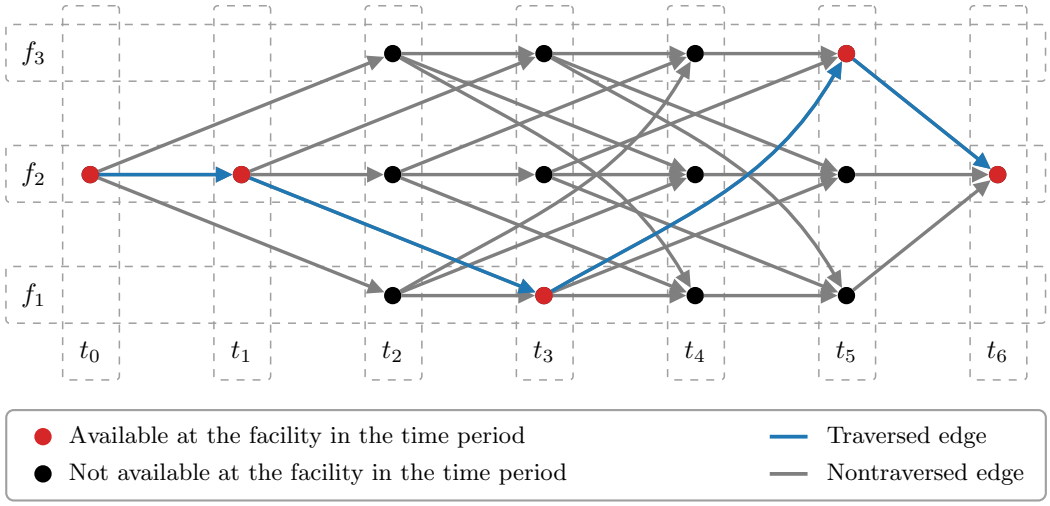

Figure 2: Example of the movement of a modular production unit in the supply chain.

Illustrative Example 2 *Figure 2 is a fictitious illustration of the movement of a modular production unit in the supply chain. In this example, there is a set of production facilities,* $\{f_1, f_2, f_3\}$*, that the modular production unit can visit over the course of the time horizon,* $\{t_1, \ldots, t_5\}$*. The modular production unit is originally located at* f_1 *and it takes the modular production unit one time period to be transferred from one facility to the other. Consequently, if the modular production unit is moved from one facility to another it cannot be operational during the time period it is in transport. From inspection of the figure, it is clear that there are two additional time periods,* t_0 *and* t_6*, which are utilized for an artificial "source-node" and "sink-node", respectively. Initially, the modular production unit is transported from the artificial "source-node" to a different facility,* f_2*, after the first time period and it arrives there at the beginning of the third time period,* t_3*; therefore, it can be operated in the second time period. At the end of the third time period, the modular production unit is routed to the third facility,* f_3*, and stays there until the end of the time horizon.*

Illustrative Example 3 *Figure 3 is a fictitious Gantt chart of the movement and operational status of a set of modular production units,* $\{k_1, k_2, \ldots, k_{10}\}$*, in the supply chain. In this example, there is a set of production facilities,* $\{f_1, f_2, \ldots, f_{20}\}$*, that the modular production units can visit over the course of the time horizon,* $\{1, 2, \ldots, 20\}$*. It should be noted, that a modular production unit's operational status can either be "active" or "inactive" and the hatched regions indicates when it is in transport between production facilities.*

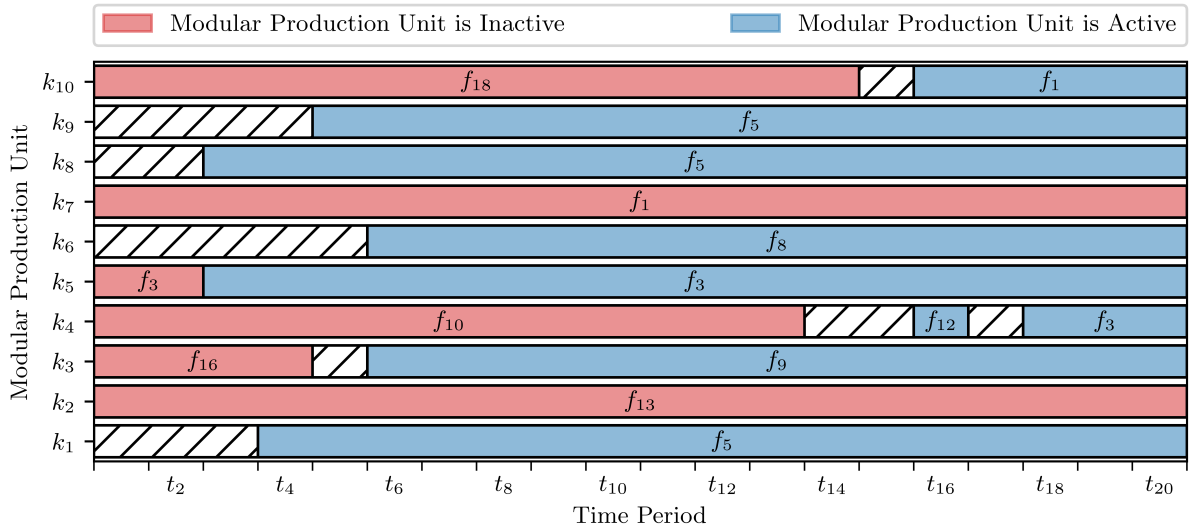

Figure 3: Gantt chart tracking the movement and operational status of the modular production units.

The decisions of the central planner are as follows and are made every time period: (i) how are the modular production units relocated between the production facilities; (ii) what is the production capabilities of the modular production units; (ii) how commodities flow between the modular production units at each production facility; (iv) how much of a commodity is stored in each warehouse; (v) how much of a commodity is disposed at the auxiliary commodity sink; (vi) how much of a commodity is purchased from the auxiliary commodity sources; and (vii) how the commodities are relocated between locations. The decisions are dependent on the underlying commodity distribution network, locations of the production facilities, availabilities of raw materials at the locations in the supply chain, and demands of final products at the locations in the supply chain, and the cost functions for the corresponding decisions. The objective of the central planner is to minimize the cost of the aforementioned operational decisions while simultaneously ensuring that the demands of commodities are met. It is assumed that perfect information regarding the availabilities of commodities, demands of commodities, and the cost of making each type of aforementioned decision is known.

3 Problem Formulation

In this section, two MILP formulations of the DMC-MPU are given: (i) the first formulation utilizes a "flow-based" approach to spatially and temporally track the modular production units, (ii) while the second formulation utilizes a "path-based" approach.

3.1 Nomenclature

Graphs

$G_1(V_1, E_1)$ a directed graph whose vertices capture the commodity source locations, production facilities, and commodity sink locations, and the edges indicate the pathways for the direct shipment of commodities between said locations and production facilities

$G_2(V_2, E_2)$ directed acyclic graph that captures the spatial and temporal locations of the modular production units – the vertices represent the facilities and corresponding time periods that a modular production unit can be located at, and the edges represent the modular production units' ability to spatially and temporally traverse the supply chain

Sets

$\mathcal{C}$ commodities, $\{1, 2, \ldots, |\mathcal{C}|\}$

$\mathcal{C}_1$ commodities that serve as raw materials, $\{c \in \mathcal{C}_1 \colon \mathcal{C}_1 \subset \mathcal{C}\}$

$\mathcal{C}_2$ commodities that serve as intermediate products, $\{c \in \mathcal{C}_2 \colon \mathcal{C}_2 \subset \mathcal{C}\}$

$\mathcal{C}_3$ commodities that serve as final products, $\{c \in \mathcal{C}_3 \colon \mathcal{C}_3 \subset \mathcal{C}\}$

$\mathcal{D}$ locations with commodity sinks, $\{d \in \mathcal{D} \colon \mathcal{D} \subseteq V_1\}$

$\mathcal{F}$ locations with production facilities, $\{f \in \mathcal{F} \colon \mathcal{F} \subseteq V_1\}$

$\mathcal{K}$ modular production units, $\{1, 2, \ldots, |\mathcal{K}|\}$

$\mathcal{P}$ all paths that the modular production units, $\mathcal{K}$, can traverse in the graph, $G_2(V_2, E_2)$

$\mathcal{T}$ discretized time horizon, $\{1, 2, \ldots, |\mathcal{T}|\}$, where operational decisions can be made

$\widehat{\mathcal{T}}$ expanded discretized time horizon, $\{0, 1, \ldots, |\mathcal{T}|\}$

$\bar{\mathcal{T}}$ expanded discretized time horizon, $\{0, 1, \ldots, |\mathcal{T}| + 1\}$

$\mathcal{R}$ locations with commodity sources, $\{r \in \mathcal{R} \colon \mathcal{R} \subseteq V_1\}$

Subsets

$\mathcal{P}(k)$ all paths that a modular production unit, $k \in \mathcal{K}$, can traverse, $\{1, 2, \ldots, |\mathcal{P}_k|\}$, in the subgraph, $G_2(V_2(k), E_2(k))$

$V_1(c)$ locations, $v \in V_1(c) \subseteq V_1$, in the supply chain that allow a specific commodity, $c \in \mathcal{C}$, to be consumed, produced, sourced, transported "from", and/or transported "to"

$E_1(c)$ edges, $(v, \bar{v}) \in E_1(c) \subseteq E_1$, in the graph, G_1, that represent the locations in the supply chain that a specific commodity, $c \in \mathcal{C}$, can be routed "from" and "to" respectively

$V_1^+(c, v)$ locations in the supply chain that a specific commodity, $c \in \mathcal{C}$, can be routed "to", $\{\bar{v} \in V_1(c) \colon (v, \bar{v}) \in E_1(c)\}$, given that it is being sent "from" a specific location

$V_1^-(c, v)$ locations in the supply chain that a specific commodity, $c \in \mathcal{C}$, can be routed "from", $\{\bar{v} \in V_1(c) \colon (\bar{v}, v) \in E_1(c)\}$, given that it is headed "to" a specific location

$V_2(k)$ tuples, $n \in V_2(k) \subseteq V_2$, of time periods and locations where a specific modular production unit, $k \in \mathcal{K}$, can reside – it should be noted that this set has been preprocessed by only allowing the vertices in the graph $G_2(V_2(k), E_2(k))$, in which the modular production unit under consideration can either produce or consume the materials or products that are available or demanded respectively at the corresponding location in the supply chain

$E_2(k)$ edges, $(n, \bar{n}) \in E_2(k) \subseteq E_2$, in the graph, G_2, that represent the time periods and facilities in the supply chain that a specific modular production unit, $k \in \mathcal{K}$, can be routed "from" and "to" respectively

$V_2^+(k, n)$ time periods and facilities that a modular production unit, $k \in \mathcal{K}$, can be routed "to" for a specific time period and facility, $\{\bar{n} \in V_2(k) \colon (n, \bar{n}) \in E_2(k)\}$, given that is it currently being routed "from" a specific facility at a given time period

$V_2^-(k, n)$ time periods and facilities that a modular production unit, $k \in \mathcal{K}$, can be routed "from" for a specific time period and facility, $\{\bar{n} \in V_2(k) \colon (\bar{n}, n) \in E_2(k)\}$, given that it will arrive at a specific location at a given time period

Parameters

$\alpha_{t,v}^c$ nominal availability of a commodity, $c \in \mathcal{C}$, at a source location, $v \in V_1(c) \cap \mathcal{R}$, during a time period, $t \in \mathcal{T}$

$\beta_{t,v}^c$ nominal demand of a commodity, $c \in \mathcal{C}$, at a sink location, $v \in V_1(c) \cap \mathcal{D}$, during a time period, $t \in \mathcal{T}$

$\gamma_{t,v,\bar{v}}^c$ maximum amount of a commodity, $c \in \mathcal{C}$, that can be transferred from a location, v, to a different, $\bar{v}$, during a time period, $t \in \mathcal{T}$, where $(v, \bar{v}) \in V_1(c)$

$\delta_{\bar{f},f}^k$ time it takes to route and integrate a modular production unit, $k \in \mathcal{K}$, from one facility in the supply chain, $\bar{f} \in \mathcal{F}$, to another facility, $f \in \mathcal{F}$

$\eta^{k,c}$ ratio of a commodity, $c \in \mathcal{C}$, a modular production unit, $k \in \mathcal{K}$, produces or consumes compared to its basis

$\theta_{(t,f)}^k$ equal to: (i) 1 if the modular production unit, $k \in \mathcal{K}$, is originally located the facility, $f \in \mathcal{F}$, and $t \equiv 0$; (ii) -1 if modular production unit, $k \in \mathcal{K}$, is originally located the facility, $f \in \mathcal{F}$, and $t \equiv |\mathcal{T}| + 1$; (iii) and 0 otherwise

$\rho_{t,v}^c$ maximum amount of a commodity, $c \in \mathcal{C}$, that can be purchased at a location, $v \in V_1(c)$, during a time period, $t \in \mathcal{T}$

σ_v^c maximum storage capacity of a commodity, $c \in \mathcal{C}$, a location, $v \in V_1(c)$

π_v^c initial amount of a commodity, $c \in \mathcal{C}$, stored at a location, $v \in V_1(c)$

$\upsilon_{t,v}^c$ maximum amount of a commodity, $c \in \mathcal{C}$, that can be disposed of at a location, $v \in V_1(c)$, during a time period, $t \in \mathcal{T}$

χ_k maximum production capacity of a modular production unit, $k \in \mathcal{K}$

$\phi_{(t,f)}^{k,p}$ equal to 1 if the modular unit, $k \in \mathcal{K}$, is located at the production facility, $f \in \mathcal{F}$, during the time period, $t \in \mathcal{T}$, when it traverses a path, $p \in \mathcal{P}(k)$; otherwise, 0

ψ_k initial location of a modular production unit, $k \in \mathcal{K}$

ω discount factor

Objective Coefficients

$\Gamma^c_{t,v,\bar{v}}$ variable cost to transport a commodity, $c \in \mathcal{C}$, from a location, v, to location, $\bar{v}$, where $(v, \bar{v}) \in E_1(c)$, during the time period, $t \in \mathcal{T}$

$P^c_{t,v}$ variable cost to purchase a commodity, $c \in \mathcal{C}$, at a location, $v \in V_1$, during the time period, $t \in \mathcal{T}$

$S^c_{t,v}$ variable cost to store a commodity, $c \in \mathcal{C}$, at a location, $v \in V_1(c)$, during the time period, $t \in \mathcal{T}$

$\bar{T}^k_p$ cost of the operation path, $p \in \mathcal{P}(k)$ for the modular production unit, $k \in \mathcal{K}$

$Y^c_{t,v}$ variable cost to dispose of a commodity, $c \in \mathcal{C}$, at a location, $v \in V_1$, during the time period, $t \in \mathcal{T}$

$X^k_{(t,f)}$ variable cost to operate a modular production unit, $k \in \mathcal{K}$, during the time period, $t \in \mathcal{T}$, at the production facility, $f \in \mathcal{F}$

$\bar{X}^k_{(t,f)}$ fixed cost to operate a modular production unit, $k \in \mathcal{K}$, during the time period, $t \in \mathcal{T}$, at the production facility, $f \in \mathcal{F}$

$\bar{Z}^k_{n,\bar{n}}$ fixed cost to transport a modular production unit, $k \in \mathcal{K}$, between two locations, f, and $\bar{f}$, where $((t, f), (\bar{t}, \bar{f})) \in (n, \bar{n}) \in E_2(k)$

Binary Variables

$\bar{t}^k_p$ 1 if the modular production unit, $k \in \mathcal{K}$, traverses the path $p \in \mathcal{P}(k)$; otherwise, 0

$\bar{x}^k_n$ 1 if the modular production unit, $k \in \mathcal{K}$, is operational at a facility, f, during a time period, t, such that $(t, f) \equiv n \in V_2(k)$; otherwise, 0

$\bar{z}^k_{n,\bar{n}}$ 1 if the modular production unit, $k \in \mathcal{K}$, is routed from a facility, f, at time period, t, to a facility, $\bar{f}$, at time period, $\bar{t}$, such that $((t, f), (\bar{t}, \bar{f})) \equiv (n, \bar{n}) \in E_2(k)$; otherwise, 0

Continuous Variables

$g^c_{t,v,\bar{v}}$ amount of a commodity, $c \in \mathcal{C}$, that is transported from a location, v, to location, $\bar{v}$, during a time period, $t \in \mathcal{T}$ where $(v, \bar{v}) \in E_1(c)$

$p^c_{t,v}$ amount of a commodity, $c \in \mathcal{C}$, that is purchased at a location, $v \in V_1$, during a time period, $t \in \mathcal{T}$, from the slack raw commodity source

$s^c_{t,v}$ amount of a commodity, $c \in \mathcal{C}$, that is stored at a location, $v \in V_1(c)$, during a time period, $t \in \mathcal{T}$

$y^c_{t,v}$ amount of a commodity, $c \in \mathcal{C}$, that is disposed of at a sink location, $v \in V_1$, during a time period, $t \in \mathcal{T}$, from the slack commodity sink

x^k_n operational capacity of a modular production unit, $k \in \mathcal{K}$, that is located at a facility, f, during a time period, t, where $(t, f) \equiv n \in V_2(k)$

3.2 Flow-Based Formulation

The MILP formulation of the "flow-based" formulation is given by F. The "flow-based" formulation is similar to other formulations in the literature that utilize a single unit flow to spatially and temporally track an object [Agra et al., 2013].

$$
\begin{aligned}
F = \min\ & J_1 + J_2 \\
\text{s.t. } & s^c_{t-1,v} + \sum\nolimits_{\bar{v}\in V_1(c,v)^-} g^c_{t,\bar{v},v} + \sum\nolimits_{k\in\mathcal{K}} \eta^{k,c}\cdot x^k_{(t,v)} + p^c_{t,v} + \alpha^c_{t,v} = \beta^c_{t,v} + \ldots && (1)\\
& y^c_{t,v} + \sum\nolimits_{\bar{v}\in V_1(c,v)^+} g^c_{t,v,\bar{v}} + s^c_{t,v}\ \forall\, c\in\mathcal{C},\, t\in\mathcal{T},\, v\in V_1(c); \\
& s^c_{t,v} = \pi^c_v\ \forall\, t\in\{0\},\, c\in\mathcal{C},\, v\in V_1(c) && (2)\\
& x^k_n \le \chi_k\cdot \bar{x}^k_n\ \forall\, k\in\mathcal{K},\, n\in V_2(k) && (3)\\
& \sum\nolimits_{\bar{n}\in V_2^+(k,n)} \bar{z}^k_{n,\bar{n}} - \sum\nolimits_{\bar{n}\in V_2^-(k,n)} \bar{z}^k_{\bar{n},n} = \theta^k_n\ \forall\, k\in\mathcal{K},\, n\in V_2(k) && (4)\\
& \bar{x}^k_n \le \sum\nolimits_{\bar{n}\in V_2^-(k,n)} \bar{z}^k_{\bar{n},n}\ \forall\, k\in\mathcal{K},\, n\in V_2(k) && (5)\\
& \bar{x}^k_n \in \{0,1\}\ \forall\, k\in\mathcal{K},\, n\in V_2(k) && (6)\\
& \bar{z}^k_{n,\bar{n}} \in \{0,1\}\ \forall\, k\in\mathcal{K},\, (n,\bar{n})\in E_2(k) && (7)\\
& g^c_{t,v,\bar{v}} \in [0,\gamma^c_{v,\bar{v}}]\ \forall\, c\in\mathcal{C},\, t\in\mathcal{T},\, (v,\bar{v})\in E_1(c) && (8)\\
& p^c_{t,v} \in [0,\rho^c_{t,v}]\ \forall\, c\in\mathcal{C},\, t\in\mathcal{T},\, v\in V_1(c) && (9)\\
& s^c_{t,v} \in [0,\sigma^c_v]\ \forall\, c\in\mathcal{C},\, t\in\widehat{\mathcal{T}},\, v\in V_1(c) && (10)\\
& x^k_n \in [0,\infty)\ \forall\, k\in\mathcal{K},\, n\in V_2(k) && (11)\\
& y^c_{t,v} \in [0,\upsilon^c_v]\ \forall\, c\in\mathcal{C},\, t\in\mathcal{T},\, v\in V_1(c) && (12)
\end{aligned}
$$

The objective functions, J_1 and J_2, in F are given by Eqs. (13) and (14) respectively. Equation (13) sums the variable costs to store commodities at all of the locations, operate all of the modular production units, transport commodities between locations, purchase commodities at all of the locations, and to dispose or sell products at all of the locations in the supply chain over the course of the time horizon. Equation (14) sums the fixed costs to operate and transport the modular production units in the supply chain over the course of the time horizon respectively.

$$
\begin{aligned}
J_1 = & \sum\nolimits_{c\in\mathcal{C}}\sum\nolimits_{t\in\mathcal{T}}\sum\nolimits_{v\in V_1(c)} S^c_{t,v}\cdot s^c_{t,v} + \ldots && (13)\\
& \sum\nolimits_{c\in\mathcal{C}}\sum\nolimits_{t\in\mathcal{T}}\sum\nolimits_{(v,\bar{v})\in E_1(c)} \Gamma^c_{t,v,\bar{v}}\cdot g^c_{t,v,\bar{v}} + \ldots \\
& \sum\nolimits_{c\in\mathcal{C}}\sum\nolimits_{t\in\mathcal{T}}\sum\nolimits_{v\in V_1(c)} P^c_{t,v}\cdot p^c_{t,v} + \ldots \\
& \sum\nolimits_{k\in\mathcal{K}}\sum\nolimits_{n\in V_2(k)} X^k_n\cdot x^k_n + \ldots \\
& \sum\nolimits_{c\in\mathcal{C}}\sum\nolimits_{t\in\mathcal{T}}\sum\nolimits_{v\in V_1(c)} Y^c_{t,v}\cdot y^c_{t,v} \\
J_2 = & \sum\nolimits_{k\in\mathcal{K}}\sum\nolimits_{n\in V_2(k)} \bar{X}^k_n\cdot \bar{x}^k_n + \ldots && (14)\\
& \sum\nolimits_{k\in\mathcal{K}}\sum\nolimits_{(n,\bar{n})\in E_2(k)} \bar{Z}^k_{n,\bar{n}}\cdot \bar{z}^k_{n,\bar{n}}
\end{aligned}
$$

The first set of constraints, Eqs. (1) - (3), in the "flow-based" formulation are utilized to model acquisition, production, inventory management, and distribution decisions. It should be highlighted that these constraints are similar to those found in *multi-item, multi-plant capacitated lot-sizing problem* formulations [Pochet and Wolsey, 2006]. Equation (1) is a general material balance constraint for each location in the supply chain that ensures the conservation of mass holds between the commodity demand sinks, commodity sources, modular production units, and storage units. Equation (3) ensures that if a modular production unit is operational, its operational set point must be less than its nameplate capacity.

The second set of constraints, Eq. (4), are utilized to model the relocation of the modular production units between the production facilities, and are based upon single unit network flow formulations [Belvaux and Wolsey, 2001; Agra et al., 2013]. In Eq. (4) the parameter, θ_n^k, is employed to insert a single unit of flow at the artificial "source-node" and to remove a single unit of flow at the artificial "sink-node". These constraints are similar to those employed to track changes over in the *capacitated lot-sizing problem* utilizing small bucket models Belvaux and Wolsey [2001] and to track ships in the *maritime inventory routing problem* [Agra et al., 2013].

The final set of constraints, Eq. (5), integrates the relocation constraints with the lot-sizing constraints by linking the binary variables, i.e., the constraints ensure that the only way the a modular production unit can be operational at a facility is if it is located there.

Equations (6) and (7) describe the binary variables that track whether or not a modular production unit is operational and whether or not a modular unit traversed between two edges in the graph G_2, respectively. Equations (8) - (12) describe the non-negative continuous variables that indicate if a commodity is transported between two locations in the supply chain, the amount of a commodity purchased from the auxiliary commodity source at a location in the supply chain, the amount of a commodity stored at a location in the supply chain, the production set point of a modular production unit at a location in the supply chain, and the amount of a commodity that is disposed of at an auxiliary commodity sink at a location in the supply chain, respectively.

3.3 Path-Based Formulation

The "path-based" MILP formulation for the DMC-MPU is given by $R(\cdot)$, here it is assumed that $\bar{\mathcal{P}}$ is equal to $\mathcal{P}$. The "path-based" formulation is similar to those put forth for modeling the *vehicle routing problem* Skitt and Levary [1985] and its affiliate problems, such as the *inventory routing problem* Gronhaug et al. [2010] and the *production routing problem* Bard and Nananukul [2010]. The primary difference between the two formulations, is that in the "path-based" approach every possible route that a modular production unit, $k \in \mathcal{K}$, can traverse through the subgraph, $G_2(V_2(k), E_2(k))$, is explicitly considered, as well as whether or not the modular production unit is operational or not when it is located at a production facility. It is worth noting that the cardinality of the set of possible operational paths, $\mathcal{P}(k)$, for a specific modular production unit, $k \in \mathcal{K}$, is on the order of $\mathcal{O}\big(|\mathcal{F}|^{|\mathcal{T}|} \cdot |\mathcal{F}|^{|\{\text{“off”},\text{“on”}\}|}\big)$, where the elements "on" and "off" indicate if the modular production unit is operational or not.

$$R(\bar{\mathcal{P}}) = \min\ J_1 + J_3$$

$$\text{s.t.} \sum\nolimits_{p\in\bar{\mathcal{P}}(k)} \bar{t}_p^k = 1 \;\forall\, k \in \mathcal{K} \tag{15}$$

$$x_n^k \leq \sum\nolimits_{p\in\bar{\mathcal{P}}(k)} \chi_k \cdot \phi_n^{k,p} \cdot \bar{t}_p^k \;\forall\, k \in \mathcal{K},\, n \in V_2(k) \tag{16}$$

$$\bar{t}_p^k \in \{0,1\} \;\forall\, k \in \mathcal{K},\, p \in \bar{\mathcal{P}}(k) \tag{17}$$

$$\text{Eqs. (1), (2), (8) - (12)}$$

The objective functions, J_1 and J_3, in R are given by Eqs. (13) and (18) respectively. Equation (18) sums the fixed cost to transport and operate the modular production units in the supply chain based on the operational paths that they traverse.

$$J_3 = \sum\nolimits_{k\in\mathcal{K}} \sum\nolimits_{p\in\bar{\mathcal{P}}(k)} \bar{T}_p^k \cdot \bar{t}_p^k \tag{18}$$

The first set of equations, Eq. (15), are set partitioning constraints and ensure that one operational path must be selected for each of the modular production units. The second set of constraints, Eq. (16), integrates the binary relocation and operational decisions, $\{\bar{t}_p^k\}_{k\in\mathcal{K},\, p\in\bar{\mathcal{P}}(k)}$, with the continuous operational decisions, $\{x_n^k\}_{k\in\mathcal{K},\, n\in V_2(k)}$. This integration is accomplished via the parameter, $\phi_{(t,f)}^{k,p}$, which is equal to 1 if the modular production unit, $k \in \mathcal{K}$, is located and operational at the facility, $f \in \mathcal{F}$, during the time period, $t \in \mathcal{T}$, when it traverses the operational path, $p \in \bar{\mathcal{P}}(k)$; otherwise it is equal to 0. It should be noted that the set of constraints in Eq. (16) in the "path-based" formulation are analogous to the constraints in Eq. (3) and Eq. (5) in the "flow-based" formulation. Moreover, the relationship between the binary relocation and operational variables in the two formulations can be explicitly given by $\sum_{\bar{n}\in V_2^-(k,n)} \bar{z}_{\bar{n},n}^k \geq \bar{x}_n^k = \sum_{p\in\bar{\mathcal{P}}(k)} \phi_n^{k,p} \cdot \bar{t}_p^k$.

The third set of equations, Eq. (17), describe the binary variables that track whether or not a modular production unit traverses a specific operational path.

The final set of equations, Eq. (1), Eq. (2), and Eqs. (8) - (12), are equivalent to those found in the "flow-based" approach and are utilized to describe acquisition, production, inventory management, and distribution decisions.

4 Decomposition Procedures

In this section, we present two decomposition procedures for the DMC-MPU. The first method presented is a column generation procedure for solving the linear programming (LP) relaxation of the "path-based" formulation of the DMC-MPU. The aim of this procedure is to prevent the a priori generation of all the operational paths, $\mathcal{P}$, that the modular production units could possibly undertake; thereby, dramatically reducing the overall complexity of constructing and solving the problem. Then, a Lagrangian relaxation procedure is presented to decompose the "path-based" formulation, which decouples the problem into a single LP problem and a set of $|\mathcal{K}|$ independent subproblems.

4.1 Column Generation Procedure

In this subsection, we present the column generation procedure to generate a subset of the operational paths, $\bar{\mathcal{P}} \subseteq \mathcal{P}$, utilized for constructing and solving the "path-based" formulation of the DMC-MPU. We then show that the column generation subproblems can be recast from

a mathematical programming problem with integer variables to a shortest path problem that can be solved in $\mathcal{O}(|E_2|)$ time.

4.1.1 Dantzig-Wolfe Decomposition

The restricted master problem for the column generation procedure is given by $R(\cdot)$. The restricted master problem is parameterized by a subset of the operational paths that the modular production units can traverse, $\bar{\mathcal{P}} \equiv \cup_{k \in K} \bar{\mathcal{P}}(k) \subseteq \mathcal{P}$, where $\bar{\mathcal{P}}(\cdot)$ is a subset of operational paths that belong to a specific modular production unit. If all of the operational paths in which the modular production units can traverse, $\mathcal{P}$, are known a priori and are embedded into the restricted master problem then $R(\mathcal{P}) = R(\bar{\mathcal{P}})$. However, for all but small problem instances constructing and solving the corresponding optimization problem, $R(\mathcal{P})$, is impractical because the cardinality of the set of all operational paths that the modular production units can traverse, $\mathcal{P}$, is on the order of $\mathcal{O}\big(|\mathcal{K}| \cdot |\mathcal{F}|^{|\mathcal{T}|} \cdot |\mathcal{F}|^{|\{\text{“off”},\text{“on”}\}|}\big)$.

Consequently, we utilize a column generation approach to generate a subset of operational paths, $\bar{\mathcal{P}} \subseteq \mathcal{P}$ – please refer to [Desaulniers et al., 2006] for a detailed treatise on column generation. This is accomplished by first solving the LP relaxation of the restricted master problem, $\bar{R}(\cdot)$, using a subset of feasible operational paths, $\bar{\mathcal{P}} \subseteq \mathcal{P}$ – there must be at least one feasible operational path for each modular production unit, i.e., $\bar{\mathcal{P}} \equiv \big\{p \in \cup_{k \in K} \bar{\mathcal{P}}(k) \colon |\bar{\mathcal{P}}(k)| \geq 1 \; \forall \; k \in \mathcal{K}\big\}$.

$$
\begin{aligned}
\bar{R}(\bar{\mathcal{P}}) = \min \; & J_1 + \sum\nolimits_{k \in \mathcal{K}} \sum\nolimits_{p \in \bar{\mathcal{P}}(k)} \bar{T}_p^k \cdot t_p^k \\
\text{s.t. } & x_n^k \leq \sum\nolimits_{p \in \bar{\mathcal{P}}(k)} \chi_k \cdot \phi_n^{k,p} \cdot t_p^k \; \forall \; k \in \mathcal{K}, \, n \in V_2(k) && (19) \\
& \sum\nolimits_{p \in \bar{\mathcal{P}}(k)} t_p^k = 1 \; \forall \; k \in \mathcal{K} && (20) \\
& t_p^k \in [0, \infty) \; \forall \; k \in \mathcal{K}, \, p \in \bar{\mathcal{P}}(k) && (21) \\
& \text{Eqs. (1), (2), (8) - (12)}
\end{aligned}
$$

Once the relaxed restricted master problem, $\bar{R}(\cdot)$, has been solved, a set of $|\mathcal{K}|$ column generation subproblems are solved. The column generation subproblems, $S_k(\cdot, \cdot)$, generate additional operational paths for each of the modular production units and are parameterized by two sets of dual variables. These dual variables, π and ψ, correspond to the constraints given in Eq. (19) and Eq. (20) of the relaxed restricted master problem, $\bar{R}(\cdot)$.

$$
\begin{aligned}
S_k(\pi, \psi) = \min \; & \sum\nolimits_{n \in V_2(k)} \Big(\bar{X}_n^k - \chi_k \cdot \pi_n^k\Big) \cdot \bar{x}_n + \sum\nolimits_{(n,\bar{n}) \in E_2(k)} \bar{Z}_{n,\bar{n}}^k \cdot \bar{z}_{n,\bar{n}} - \psi_k \\
\text{s.t. } & \sum\nolimits_{\bar{n} \in V_2^+(k,n)} \bar{z}_{n,\bar{n}} - \sum\nolimits_{\bar{n} \in V_2^-(k,n)} \bar{z}_{\bar{n},n} = \theta_n^k \; \forall \; n \in V_2(k) && (22) \\
& \bar{x}_n \leq \sum\nolimits_{\bar{n} \in V_2^-(k,n)} \bar{z}_{\bar{n},n} \; \forall \; n \in V_2(k) && (23) \\
& \bar{x}_n \in \{0, 1\} \; \forall \; n \in V_2(k) && (24) \\
& \bar{z}_{n,\bar{n}} \in \{0, 1\} \; \forall \; (n, \bar{n}) \in E_2(k) && (25)
\end{aligned}
$$

After the column generation subproblems have been solved, the subset of operational paths, $\bar{\mathcal{P}} \subseteq \mathcal{P}$, is updated to include the newly generated operational paths. This process continues

in an iterative fashion until the reduced cost for all the modular production units are strictly non-negative.

It should be noted, that at any point in the column generation procedure the bounds on the LP relaxation of the "path-based" formulation of the DMC-MPU, $\bar{R}$, can be easily computed. This is accomplished by utilizing the current objective values of the relaxed restricted master problem, $\bar{R}(\cdot)$, and the column generation subproblems, $S_k(\cdot,\cdot)$.

$$\sum\nolimits_{k\in\mathcal{K}} S_k(\pi,\psi) + \bar{R}(\bar{\mathcal{P}}) \leq \bar{R} \leq \bar{R}(\bar{\mathcal{P}})$$

4.1.2 Reformulation of the Subproblems

In the following, we present the process utilized to recast the column generation subproblems from mathematical programming problems with binary variables to shortest path problems. We then present a dynamic programming algorithm that is able to solve each of the shortest path problems in $\mathcal{O}(|E_2|)$ time.

The original integer programming formulation of the pricing problem, $S_k(\cdot,\cdot)$, can be recast to a reduced integer programming problem, $\bar{S}_k(\cdot,\cdot)$, where $[\cdot]_-$ is a function that is equivalent to $\min\{\cdot, 0\}$.

$$\begin{aligned}\bar{S}_k(\pi,\psi) = \min\ & \sum\nolimits_{(n,\bar{n})\in E_2(k)} \left(\left[\bar{X}^k_{\bar{n}} - \chi_k\cdot\pi^k_{\bar{n}}\right]_- + \bar{Z}^k_{n,\bar{n}}\right)\cdot\bar{z}_{n,\bar{n}} - \psi_k \\ & \text{s.t. Eqs. (22), (25)}\end{aligned}$$

Once the nonredundant formulation, $\bar{S}_k(\cdot,\cdot)$, has been solved, the associated operational variables, $\{\bar{x}_n\}_{n\in V_2(k)}$, can be solved for a posteriori using the following decision making rule in conjunction with the optimal allocation variables, $\{\bar{z}_{\bar{n},n}\}_{(\bar{n},n)\in E_2(k)}$.

$$x_n = \begin{cases} 1, & \texttt{isnegative}\left(\bar{X}^k_n - \chi_k\cdot\pi^k_n\right)\ \wedge\ \texttt{isone}\left(\sum_{\bar{n}\in V_2(k,n)^-}\bar{z}_{\bar{n},n}\right) \\ 0, & \text{otherwise}\end{cases}$$

The function `isnegative`$(\cdot)$ returns `true` if its input is negative; otherwise it returns `false`. In a similar fashion, the function `isone`$(\cdot)$ returns `true` if its input is equal to one; otherwise it returns `false`.

From inspection of the column generation subproblems, $\bar{S}_k(\cdot,\cdot)$, there is only one set of algebraic constraints, *Eq.* (22). These algebraic constraints are network flow balance constraints and have been mapped from the graph $G_2(V_2, E_2)$. In these constraints a single unit of flow is inserted at the artificial "source-node" and is removed at the artificial "sink-node", which in turn allow the location of the modular production unit to be tracked as it traverses the time horizon

Consequently, the column generation subproblems, $\bar{S}_k(\cdot,\cdot)$, can be posed as minimum cost network flow problems. They can then be recast as shortest path problems because there is only one unit of flow inserted and one unit of flow removed. To solve the shortest path problems, we have developed a problem specific two-stage dynamic programming algorithm with a complexity of $\mathcal{O}(|E_2|)$. The proposed algorithm allows for possible negative edge weights in the graph, which

can arise from the negative reduced operational costs.

An overview of the bespoke two-stage dynamic programming algorithm we have developed for solving the column generation subproblems is presented in Algorithm 1. The algorithm accepts as input the modular production unit under consideration as well as the optimal dual variables, π and ψ, that were found by solving the restricted master problem, $\bar{R}(\cdot)$. The algorithm returns three different items: (i) ϕ, a data structure of 0's and 1's, which indicates when and where the modular production unit is operational; (ii) *primalCost*, the objective value of the solution excluding the reduced operational cost; and (iii) *reducedCost*, the objective value of the solution including the reduced operational cost.

Algorithm 1: Dynamic Programming Algorithm

input : k − modular production unit under consideration
π − dual variables associated with the constraints given in Eq. (19)
ψ − dual variables associated with the constraints given in Eq. (20)

output: ϕ − indicates if the modular production unit is operational at a node
primalCost − primal costs of the operational path
reducedCost − reduced costs of the operational path given the dual variables

Function `DynamicProgrammingAlgorithm(`k, π, ψ`):`

`/* foward pass –` $V_2(\cdot)$ `is a lexicographically sorted list */`

$distance \leftarrow$ `Dict(`$n \leftarrow$ (`isone(`i`)` ? ψ_k : `inifinity)` **for** $(i, v) \in$ `enumerate(`$V_2(k)$`))`

$predecessor \leftarrow$ `Dict(`$n \leftarrow$ `null` **for** $v \in V_2(k)$`)`

foreach $(i, n) \in$ `enumerate(`$V_2(k)$`)` **do**

  if ¬`isone(`i`)` **then**

    $\bar{n} \leftarrow \arg\min \left\{ distance[\bar{n}] + \bar{Z}^k_{\bar{n},n} : \bar{n} \in V_2^-(k, n) \right\}$

    $distance[n] \leftarrow distance[\bar{n}] + \bar{Z}^k_{\bar{n},n} + \left[\bar{X}^k_n - \chi_k \cdot \pi^k_n \right]_-$

    $predecessor[n] \leftarrow \bar{n}$

`/* backwards pass */`

$\phi \leftarrow$ `Dict(`$n \leftarrow 0$ **for** $v \in V_2(k)$`)`

$primalCost \leftarrow 0.0$

$reducedCost \leftarrow distance[V_2(k)$[`last`]]

with $V_2(k)$[`last`] **as** n **do**

  while $n \neq V_2(k)$[`first`] **do**

    if `isnegative`$\left(\bar{X}^k_n - \chi_k \cdot \pi^k_n\right)$ **then**

      $\phi[n] \leftarrow 1$

    $primalCost \leftarrow primalCost + \bar{Z}^k_{predecessor[n],n} + \bar{X}^k_n \cdot \phi[n]$

    $n \leftarrow predecessor[n]$

  return ϕ, *primalCost*, *reducedCost*

End Function

In the first phase of the algorithm, a forward pass is undertaken from the artificial "source-node" to the artificial "sink-node". The forward pass exploits the time dependant nature that exists between nodes in the acyclic graph, $G_2(V_2(\cdot), E_2(\cdot))$, to construct the minimal distance and optimal precedence between all of the nodes, $V_2(\cdot)$, in the graph and the artificial "source-node". This intrinsic time dependant nature is apparent in Figure 2. From further inspection of Figure 2 it is clear that each node in the graph has at most $|\mathcal{F}|$ in neighbors. Practically,

each of these in neighbors correspond to a production facility that the modular production unit could be coming from. Consequently, if the set, $V_2(\cdot)$, is sorted in a lexicographical order by time period then the distance and optimal precedence relationships can be computed at worst in $\mathcal{O}(|V_2| \cdot |\mathcal{F}|)$, or equivalently $\mathcal{O}(|E_2|)$, time – this result can be seen in lines 4-8 in Algorithm 1. The function `enumerate`$(\cdot)$ accepts an input in the form of a list and returns a list of tuples, where the first item in the tuple is a counter and the second item is the value from the input list.

In the second phase of the algorithm, a backwards pass is undertaken from the artificial "sink-node", which is the last element in $V_2(\cdot)$, to the artificial "source-node", which is the first element in $V_2(\cdot)$, through the *predecessor* data structure. The backwards pass computes the objective value of the solution excluding the reduced operational cost, the objective value of the solution including the reduced operational cost, and the shortest path to the artificial "source-node" in at worst $\mathcal{O}(|\mathcal{T}|)$ time.

4.2 Lagrangian Relaxation Procedure

The Lagrangian relaxation procedure decomposes the restricted "path-based" formulation, $R(\cdot)$, into a single LP problem and a set of $|\mathcal{K}|$ independent subproblems by relaxing the constraints given in Eq. (19). The constraints given by Eq. (19) integrate the variables that govern if a modular production unit is operational or not at a production facility at a given time period, with the production set point of the modular production unit at the corresponding production facility and time period. The resulting Lagrangian problem is given by $\bar{R}(\cdot,\cdot)$ and is parameterized by the Lagrangian multipliers π, which correspond to the dual variables for the relaxed constraints given in Eq. (19), and a subset of the operational paths, $\bar{\mathcal{P}} \subseteq \mathcal{P}$.

$$\begin{aligned}\bar{R}(\pi, \bar{\mathcal{P}}) = \min\ & J_1 + \sum\nolimits_{k\in\mathcal{K}} \sum\nolimits_{p\in\bar{\mathcal{P}}(k)} \bar{T}_p^k \cdot t_p^k + \dots \\ & \sum\nolimits_{k\in\mathcal{K}} \sum\nolimits_{n\in V_2(k)} \pi_n^k \cdot \left(x_n^k - \sum\nolimits_{p\in\bar{\mathcal{P}}(k)} \chi_k \cdot \phi_n^{k,p} \cdot t_p^k\right) \\ \text{s.t. } & \text{Eqs. (1), (2), (8) - (12), (20), (21)}\end{aligned}$$

From inspection of the Lagrangian problem, $\bar{R}(\cdot,\cdot)$, it is evident that the problem can be decomposed into a single LP problem, given by $\dot{\bar{R}}(\cdot,\cdot)$, and as a set of $|\mathcal{K}|$ independent subproblems, given by $\ddot{\bar{R}}_k(\cdot,\cdot)$. The distribution, production, and storage decisions in Lagrangian problem, $\bar{R}(\cdot,\cdot)$, are captured by $\dot{\bar{R}}(\cdot)$, while the operational and routing decisions are captured by the independent subproblems $\ddot{\bar{R}}_k(\cdot,\cdot)$.

$$\begin{aligned}\dot{\bar{R}}(\pi) = \min\ & J_1 + \sum\nolimits_{k\in\mathcal{K}} \sum\nolimits_{n\in V_2(k)} \pi_n^k \cdot x_n^k \\ \text{s.t. } & \text{Eqs. (1), (2), (8) - (12)}\end{aligned}$$

$$\begin{aligned}\ddot{\bar{R}}_k(\pi, \bar{\mathcal{P}}) = \min\ & \sum\nolimits_{p\in\bar{\mathcal{P}}(k)} \left(\bar{T}_p^k - \sum\nolimits_{n\in V_2(k)} \chi_k \cdot \pi_n^k \cdot \phi_n^{k,p}\right) \cdot t_p \\ \text{s.t. } & \sum\nolimits_{p\in\bar{\mathcal{P}}(k)} t_p = 1 \qquad (26) \\ & t_p \in [0,\infty)\ \forall\ p \in \bar{\mathcal{P}}(k) \qquad (27)\end{aligned}$$

From further inspection of the independent subproblems, $\ddot{\bar{R}}_k(\cdot,\cdot)$, it is clear that they can be reformulated from mathematical programming problems to a form that allows them to be solved algorithmically in $\mathcal{O}(|\bar{\mathcal{P}}(\cdot)|)$ time. This reformulation is possible because in its mathematical programming problem form, $\ddot{\bar{R}}_k(\cdot,\cdot)$, has only one algebraic constraint, Eq. (26), which is a set partitioning constraint on a relaxed variable. Consequently, the variable corresponding to the operational path, $p \in \bar{\mathcal{P}}(k)$, with the minimum objective coefficient is selected; thereby, ensuring that the solution to the operational and routing decisions are integer feasible.

$$\ddot{\bar{R}}_k(\pi, \bar{\mathcal{P}}) = \min_{p \in \bar{\mathcal{P}}(k)} \left\{ \bar{T}_p^k - \sum\nolimits_{n \in V_2(k)} \chi_k \cdot \pi_n^k \cdot \phi_n^{k,p} \right\}$$

At every iteration of the Lagrangian relaxation procedure for the "path-based" formulation, the Lagrangian multipliers, π, are updated utilizing a subgradient method. The direction of the subgradient at the i^{th} iteration is given by the vector ζ^i, which is computed by utilizing the optimal values of the production and operational decisions found in the most recent solution of $\bar{R}(\cdot,\cdot)$.

$$\zeta_n^{k,i} = x_n^k - \sum\nolimits_{p \in \bar{\mathcal{P}}(k)} \chi_k \cdot \phi_n^{k,p} \cdot t_p^k \;\; \forall\, k \in \mathcal{K},\, n \in V_2(k)$$

Consequently, given the subgradient directions, ζ, the vector of Lagrangian multipliers, π, can be updated at the iteration to subsequent to i by,

$$\pi_n^{k,i+1} = \left[\pi_n^{k,i} + \lambda^i \cdot \frac{R^* - \bar{R}(\pi^i, \bar{\mathcal{P}}^i)}{\sum_{\bar{k} \in \mathcal{K}} \sum_{\bar{n} \in V_2(k)} \left(\zeta_{\bar{n}}^{\bar{k},i} \right)^2} \cdot \zeta_n^{k,i} \right]_+ \quad \forall\, k \in \mathcal{K},\, n \in V_2(k)$$

where R^* is the objective value of the best feasible solution found thus far, $\bar{R}(\pi^i, \bar{\mathcal{P}}^i)$ is the objective value of the Lagrangian problem using the most recent Lagrangian multipliers and operational paths, and λ^i is the stepsize utilized in the subgradient method, where $[\cdot]_+$ is a function that is equivalent to $\max\{\cdot, 0\}$ [Jena et al., 2017].

5 Path-Based Matheuristic

In this section, we present the iterative three-stage matheuristic, which is given in Algorithm 2. The matheuristic utilizes a Lagrangian relaxation based heuristic to construct a set of locally optimal operational paths via the "path-based" formulation and then utilizes a randomized path-relinking procedure to possibly further improve the solution. The algorithm accepts four inputs: (i) the first input argument is the greediness parameter that is utilized in constructing a random low cost solution; (ii) the second input argument governs how many times the Lagrangian multipliers are updated before a new set of locally optimal operational paths is generated; (iii) the third input argument is a limitation on the run time of the algorithm; and (iv) the final argument is convergence tolerance that can be utilized to break out of the algorithm before the time limitation is met. The algorithm returns a locally optimal solution to the problem. At the beginning of the algorithm, the Lagrangian multipliers, π, an empty solution pool, $\bar{S}$, to store sets of locally optimal operational paths are initialized, and an initial set of operational paths, $\bar{P}$, is initialized.

Once the three data structures have been initialized, the iterative procedure begins. In

Algorithm 2: Path Based Matheuristic

input : *greediness* − greediness of the procedure
iterations − number of iterations Lagrangian multipliers are updated
timeLimit − maximum run time of the algorithm
tolerance − convergence tolerance

output: *solution* − optimal solution to the problem

Function `PathBasedMatheuristic(`*greediness, iterations, timeLimit, tolerance*`)`**:**
  $\bar{P} \leftarrow$ initialize a set of operational paths
  $\bar{S} \leftarrow$ empty solution pool to store sets of locally optimal operational paths
  $\pi \leftarrow$ initialize set of lagrangian multipliers
  repeat
    $S \leftarrow$ set of locally optimal operational paths found by solving $\bar{R}(\pi, \bar{P})$
    `/* Generate new operational paths */`
    foreach $k \in \mathcal{K}$ **do**
      $\psi_k \leftarrow$ objective coefficent for the optimal path selected when solving $\ddot{\bar{R}}_k(\pi, \bar{P})$
      $\bar{P} \leftarrow \bar{P} \cup \{$operational path found from $S_k(\pi_k, \psi_k)\}$
    `/* Path relinking procedure */`
    if $\bar{S} \neq \emptyset$ **then**
      $S \leftarrow$ `pathRelinking`$(S, \bar{S}, greediness)$
      $\bar{S} \leftarrow \bar{S} \cup S$
    else
      $\bar{S} \leftarrow \{S\}$
    `/* Solve the Lagrangian dual problem` *iterations* `iterations */`
    foreach $i \in \{1, \ldots, iterations\}$ **do**
      Solve $R(\pi, \bar{P})$ to compute new lower bound
      $\pi \leftarrow$ update Lagrangian multipliers using the subgradient method
  until the *timeLimit* or the convergence *tolerance* is reached
  solution $\leftarrow \arg\min\{$`UpperBound`$(s)\colon s \in \bar{S}\}$
  return *solution*
End Function

the first stage, the Lagrangian relaxation procedure for "path-based" formulation using the current Lagrangian multipliers, π, is initiated and a locally optimal integer feasible solution to the operational paths are found by solving $\dot{\bar{R}}(\cdot,\cdot)$ and the set of independent subproblems $\ddot{\bar{R}}_k(\cdot,\cdot)$. Then, new operational paths are computed for each of the modular production units by calling the respective column generation subproblem, $\bar{S}(\cdot,\cdot)$. It should be noted that the dual variables for the constraints in Eq. (20) can be approximated by utilizing the objective coefficient found by solving the independent subproblem $\ddot{\bar{R}}_k(\cdot,\cdot)$. In the second phase, the solution pool, $\bar{S}$, gets the solution to the locally optimal operational path that was just found by solving the Lagrangian relaxation procedure. In the final phase, the Lagrangian multipliers are updated *iterations* times utilizing the subgradient method. If the time limit, *timeLimit*, has been reached or the convergence gap between the best known upper bound to the problem computed via `upperBound`$(\cdot)$ and the best known lower bound to the problem computed via $F(\cdot)$ is less than the convergence tolerance, *tolerance*, the algorithm breaks out of the loop;

otherwise, a subsequent iteration is initiated.

If a second iteration of the matheuristic is initiated, the Lagrangian relaxation procedure is repeated using the newly updated set of Lagrangian multipliers, π, and a new set of operational paths are generated for each of the modular production units, which are then passed to the path-relinking procedure, Algorithm 3.

Algorithm 3: Path Relinking Procedure

input : P − locally optimal operational paths
$\bar{S}$ − solution pool that stores sets of locally optimal operational paths
greediness − greediness of the procedure
output: $\ddot{P}$ − best solution found through the relinking procedure

Function `pathRelinking`(P, S, *greediness*)**:**
  $\ddot{P} \leftarrow P$
  $K \leftarrow \mathcal{K}$
  $\dot{P} \leftarrow$ `randomLowCostSolution`($\bar{S}$, *greediness*)
  repeat
    $\bar{K} \leftarrow$ random subset of elements from K
    $P \leftarrow$ `randomLowCostSolution`($\{(P \setminus \{P[k]\}) \cup \{\dot{P}[k]\} \colon k \in \bar{K}\}$, *greediness*)
    $\ddot{P} \leftarrow \arg\min\{$`upperBound`$(p) \colon p \in \{\ddot{P}, P\}\}$
    $K \leftarrow K \cap \{k \in \bar{K} \colon P[k] = \dot{P}[k]\}$
  until $K = \emptyset$
  return $\ddot{P}$
End Function

The path-relinking procedure accepts three arguments: (i) the first input, P, is a locally optimal initial solution; (ii) the second input, $\bar{S}$, is a set of locally optimal operational paths that acts as a solution pool; and (iii) the final input, *greediness*, is the greediness parameter that is utilized in constructing a random low cost solution.

At the beginning of the path-relinking procedure three data structures are initialized. The first data structure, $\ddot{P}$, is initialized with the locally optimal initial solution and is utilized to store the best solution found in the path-relinking procedure. The second data structure, K, is initialized with the initial set of modular production units, $\mathcal{K}$. The final data structure, $\dot{P}$, is the guiding solution for the path-relinking procedure and it is selected in a semi-greedy fashion from the solution pool, $\bar{S}$, via the `randomLowCostSolution`$(\cdot, \cdot)$ function.

Algorithm 4: Random Low Cost Solution

input : $\bar{S}$ − solution pool that stores sets of locally optimal operational paths
greediness − greediness of the procedure
output: P − operational paths chosen in a semi-greedy fashion

Function `randomLowCostSolution`($\bar{S}$, *greediness*)**:**
  $best \leftarrow \min\{$`upperBound`$(p) \colon p \in \bar{S}\}$
  $worst \leftarrow \max\{$`upperBound`$(p) \colon p \in \bar{S}\}$
  $P \leftarrow$ `randomElement`$(\{p \in \bar{S} \colon$ `upperBound`$(p) \leq best + greediness \cdot (worst - best)\})$
  return P
End Function

The random low cost solution function, which is explicitly presented in Algorithm 4, accepts a solution pool that stores sets of locally optimal operational paths and a greediness parameter. The function returns a set of locally optimal operational paths chosen in a semi-greedy fashion based upon the greediness parameter.

After the three data structures are initialized in the path-relinking procedure, the initial solution, P, is slowly merged to the guiding solution, $\dot{P}$. This merging occurs in an iterative fashion and takes $|\mathcal{K}|$ iterations to be completed. At every iteration in this procedure, a random subset, $\bar{K}$, of the modular production units whose paths have not been merged from the guiding solution, $\dot{P}$, to the initial solution, P, is selected. From this set of candidates a random low cost solution is selected by merging one the operational paths for one of the candidate modular production units, $k \in \bar{K}$, from the guiding solution, $\dot{P}$, to the merged solution, P. Then the procedure checks if this new merged solution, P, provides a better upper bound than the best one found thus far, $\ddot{P}$, and if so it updates the best solution found thus far accordingly. Finally, the modular production unit selected from the set of candidates chosen to merge is removed from the set K. This iterative procedure is executed until the set of candidate modular production units, K, is empty. The best solution found in the path-relinking procedure, $\ddot{P}$, is then returned to the matheuristic.

The new set of locally optimal operational paths generated via the path-relinking procedure is then appended to the solution pool, $\bar{S}$. Finally, the Lagrangian multipliers are updated *iterations* times utilizing the subgradient method. This process repeats until the time limit, *timeLimit*, is reached or the difference between the best known upper bound to the problem computed via `upperBound`$(\cdot)$ and the best known lower bound to the problem computed via $\bar{R}(\cdot,\cdot)$ is less than the convergence tolerance, *tolerance*.

6 Numerical Experiments

In this section, we present a computational case study using randomly generated test instances to show the effectiveness of the proposed matheuristic. In this case study the matheuristic is benchmarked against a "state-of-the-art" MILP solver, which solves the full-space "flow-based" mathematical programming model.

6.1 Test Instance Generation

In order to ensure a fair comparison between the performance of the proposed matheuristic and a "state-of-the-art" MILP solver we generated a set of 750 test instances. Since there was no available public data or data within the literature, we had to generate the 750 test instances randomly. To reduce the appearance of bias, we randomly generated the parameters utilized in the test instances using uniform distributions that have wide lower and upper limits. The specific manner in which these test instances were generated is described in detail in the supplementary information.

6.2 Computational Study

In this subsection, we present the results to the computational case study in which we benchmark the proposed matheuristic against a "state-of-the-art" MILP solver, which solves the full-space "flow-based" mathematical programming formulation of the test instances. We

classify the test instances utilized in the computational case study into three different groups. The groups are categorized by the amount of time it takes the "state-of-the-art" MILP solver to solve them to an optimality gap of less than 1%. The three groups are: (i) test instances that take less than 1,000 seconds to solve; (ii) test instances that take between 1,000 and 10,000 [seconds] to solve; and (iii) test instances that cannot be solved within 10,000 [seconds]. The optimality gap is defined as, $\min\{|U - L^*| \,/\, U,\, 1\}$, where U is the current upper bound and L^* is the best known lower bound.

The algorithms were implemented in Julia 1.7.0, the mathematical programming models were constructed utilizing JuMP 0.22.2 [Dunning et al., 2017], and the mathematical programming problems were solved using Gurobi 9.5 [Gurobi Optimization, LLC, 2022]. The experiments were performed on a machine at the Texas A&M High Performance Research Computing (HPRC) with an Intel Xeon 6248R 3.0GHz 24-core processor with its usable threads capped at 4. The maximum run time for the matheuristic and the "state-of-the-art" MILP solver was set to 1,000 seconds and 10,000 seconds, respectively, while the optimality gap was set to 1.0% for the matheuristic and the "state-of-the-art" MILP solver.

In the following paragraphs, we present the results for the test instances using the benchmark method. These results have been computed by solving the full-space "flow-based" mathematical programming formulation of the test instances using a "state-of-the-art" MILP solver. Of the the 750 randomly generated test instances, approximately 29% of them were solved within 1,000 seconds, approximately 39% of them were solved between 1,000 seconds and 10,000 seconds, and approximately 32% were not solved within 10,000 seconds.

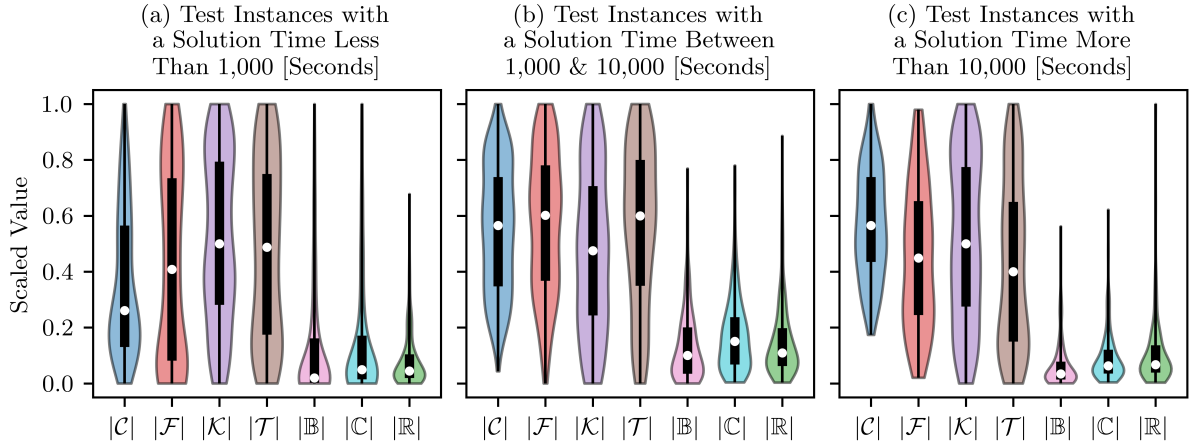

Figure 4: Descriptive statistics for the scaled sizes of the grouped test instance attributes.

Figure 4 presents three sets of violin plots that illustrate the descriptive statistics for the scaled sizes of the grouped test instance attributes, where the white dot represents the mean, the thicker of the vertical bars represents the confidence interval between 25 [%] and 75 [%], and the horizontal thickness of the violin plots represents the kernel density. These plots are utilized to show how the computational time required to solve the problem is affected by the number of commodities in the supply chain, $|\mathcal{C}|$, the number of production facilities in the supply chain, $|\mathcal{F}|$, the number of modular production units utilized, $|\mathcal{K}|$, and the number of time periods decisions can be made, $|\mathcal{T}|$, as well as the number of binary variables, $|\mathbb{B}|$, number of linear algebraic constraints, $|\mathbb{C}|$, and number of continuous variables, $|\mathbb{R}|$, utilized in the mathematical

programming model.

In these three figures, the y-axis has been normalized to values between 0 and 1 due to the fact that the cardinalities of the sets, the number of variables in the mathematical programming model, and the number of linear algebraic constraints in the mathematical programming model are all on different orders of magnitude. The extreme values of the sets and the statistics of the mathematical programming model are presented in Table 1. From inspection of these three sub-figures, it can be seen that the only apparent test instance attribute that indicates the problem will be difficult to solve is the number of commodities utilized in the supply chain, $|\mathcal{C}|$. Specifically, when looking at Fig. 4.a it is clear that mean number of commodities utilized in the supply chain, $|\mathcal{C}|$, is approximately 7. However, in the more difficult test instances to solve, as seen in Fig. 4.b and Fig. 4.c, the mean number of commodities utilized in the supply chain is closer to 14. For all of the other test instance attributes, the mean and confidence interval is roughly the same across all bucketed test instances.

Table 1: Bounds on the un-scaled values of the test instance attributes.

	$\|\mathcal{C}\|$	$\|\mathcal{F}\|$	$\|\mathcal{K}\|$	$\|\mathcal{T}\|$	$\|\mathbb{B}\|$	$\|\mathbb{C}\|$	$\|\mathbb{R}\|$
Minimum	2	1	10	10	420	5,839	2,184
Maximum	25	50	50	50	3,868,855	12,951,216	2,548,561

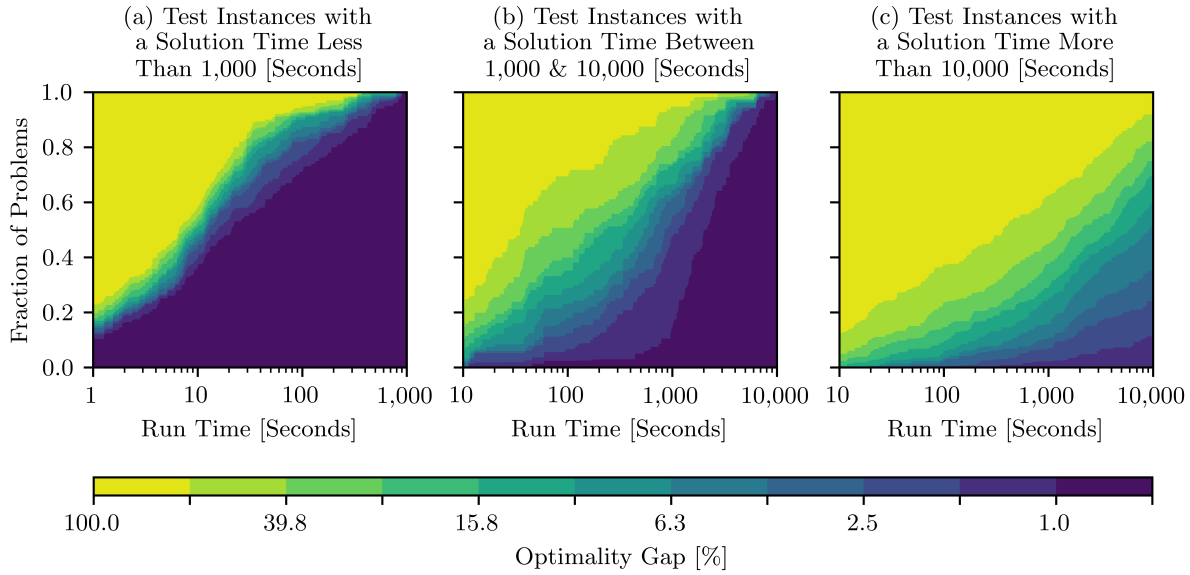

Figure 5: Convergence profiles for the aggregated test instances.

Figure 5 illustrates the convergence profiles for the three different aggregated groups of test instances. As aforementioned, the aggregation of the test instances into three buckets is as follows: (i) the first bucket includes test instances that took the commercial solver less than 1,000 seconds to solve the "flow-based" formulation, given by Fig. 5.a; (ii) the second bucket includes test instances that took the commercial solver between than 1,000 seconds and 10,000 seconds to solve the "flow-based" formulation, given by Fig. 5.b; and (iii) the final bucket includes test instances that took the commercial solver over than 10,000 seconds to solve the "flow-based" formulation, given by Fig. 5.c. In each of the filled contours plots in Figure 5,

the x-axis represents the run time, the y-axis represents the fraction of problems, and the filled color represents the optimality gap. It is worth pointing out that in these filled contours plots the x-axis and the filled color is on a log scale.

Performance of the Proposed Matheuristic

In the following paragraphs, we compare the computational performance of the "state-of-the-art" MILP solver to the proposed matheuristic with and without the use of the path-relinking procedure. It should be pointed out, that two initial operational paths for each modular production unit in the matheuristic, as given by $\bar{P}$ in line 2 of Algorithm 2, were generated by assuming: (i) that the modular production unit, $k \in \mathcal{K}$, is never moved from its initial location in the supply chain, $\psi_k \in \mathcal{F}$, and that it is operational in every time period; and (ii) that the modular production unit, $k \in \mathcal{K}$, is never moved from its initial location in the supply chain, $\psi_k \in \mathcal{F}$, and that it is never operational.

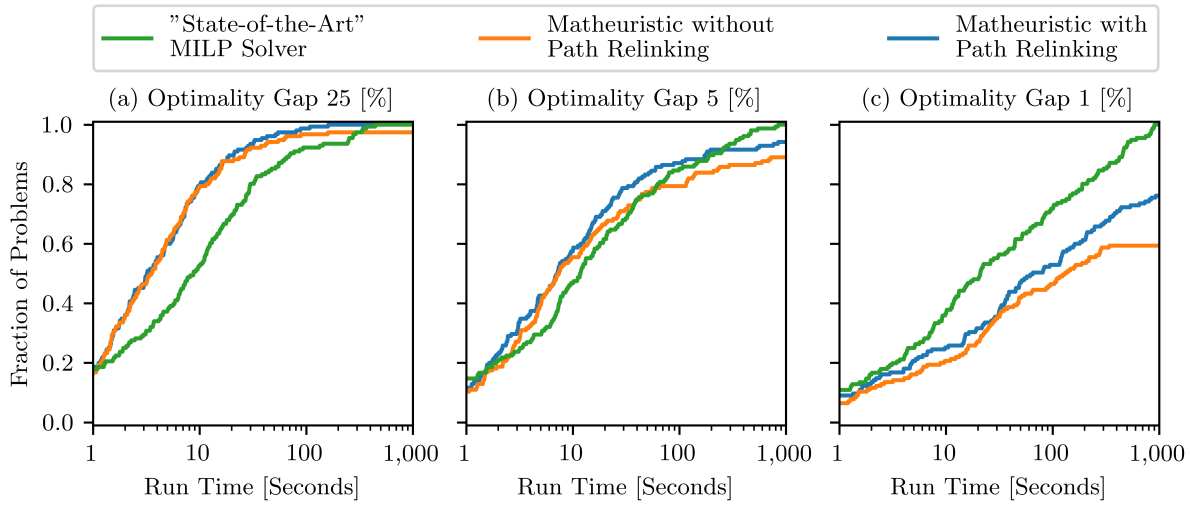

Figure 6: Performance curves for the test instances that take the "state-of-the-art" MILP solver less than 1,000 [seconds] to solve.

Figure 6 illustrates the performance curves of the "state-of-the-art" MILP solver and the matheuristics for the group of test instances that take the mathematical programming solver less than 1,000 seconds to solve to an optimality gap of 1%. From inspection of the first group of test instances, it is clear that the proposed matheuristics produce a slightly tighter initial bounds when compared to the "state-of-the-art" MILP solver. However, unlike the "state-of-the-art" MILP solver, the proposed matheuristics both end up getting stuck in local optimal solutions and are unable to solve all of the test instances to the desired optimality gap of 1%. It should be noted, that the path-relinking slightly improves the performance of the matheuristic.

Figure 7 illustrates the performance curves of the "state-of-the-art" MILP solver and the matheuristics for the group of test instances that take the mathematical programming solver between than 1,000 seconds and 10,000 seconds to solve to an optimality gap of 1%. From inspection of Figure 7 it is clear that both of the proposed matheuristics produce are able to dramatically outperform the "state-of-the-art" MILP solver in the first 1,000 seconds. Specifically the both types of matheuristics are able to solve roughly 75% of the problems to an optimality gap of at worst 25% in over 1.5 orders of magnitude in time faster than the "state-of-

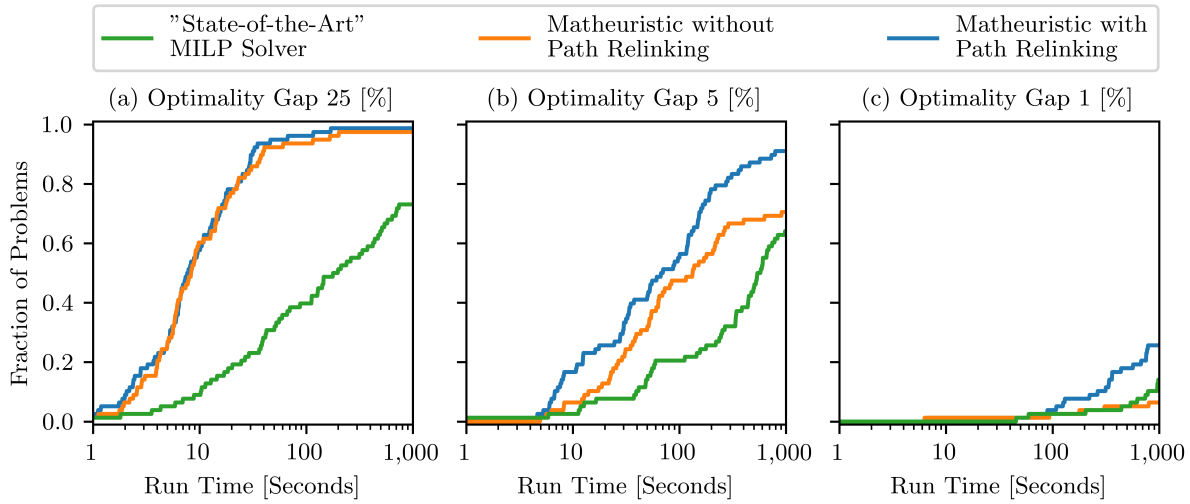

Figure 7: Performance curves for the test instances that take the "state-of-the-art" MILP solver between 1,000 and 10,000 seconds to solve.

the-art" MILP solver. Similarly, the proposed matheuristic with the path-relinking procedure is able to solve approximately 65% of the test instance in this group to an optimality gap of at worst 5% in a little over 1 order of magnitude faster than the "state-of-the-art" MILP solver. Along the same track, the proposed matheuristic with the path-relinking procedure is able to solve a larger portion of the problems to an optimally gap of 1% than the "state-of-the-art" MILP solver.

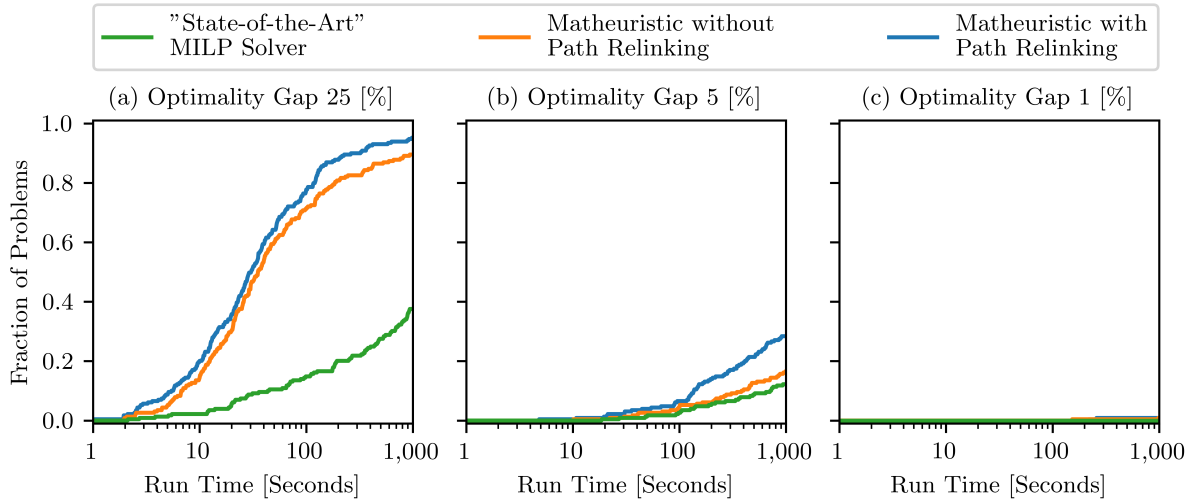

Figure 8: Performance curves for the test instances that take the "state-of-the-art" MILP solver more than 10,000 seconds to solve.

V

Figure 8 illustrates the performance curves of the "state-of-the-art" MILP solver and the matheuristics for the group of test instances that take the mathematical programming solver is unable to solve to an optimality gap of 1% in 10,000 seconds. From inspection of Figure 8, it is clear that the proposed matheuristic with the path-relinking procedure is much more effective than the "state-of-the-art" MILP solver. Notably, the proposed matheuristics are able to solve approximately 30% of the test instances to an optimality gap of at worst 25% in over

1.5 orders of magnitude of time faster than the "state-of-the-art" MILP solver. Moreover, the matheuristics are able to solve roughly 3 times as many problems to an optimality gap of at worst 25% in 1,000 seconds, when compared to the "state-of-the-art" MILP solver. Furthermore, the proposed matheuristic with the path-relinking procedure is able to solve roughly 2 times as many problems to an optimality gap of at worst 5% in 1,000 seconds, when compared to the "state-of-the-art" MILP solver.

7 Concluding Remarks

In this work, we present a systematic framework to determine the optimal operation of multi-commodity supply chains that harness modular manufacturing techniques, which we refer to as the *dynamic multiple commodity supply chain problem with modular production units*. In the DMC-MPU the central planner has the ability to relocate the modular production units between production facilities to meet the spatial and temporal changes in the availabilities, demands, and prices of the underlying commodities. The framework includes two different mathematical programming formulations utilizing MILP to solve the DMC-MPC. The first MILP model utilizes a "flow-based" formulation to spatially and temporally track the relocation and operational decisions for the modular production units, while the second MILP model utilizes a "path-based" formulation whose LP relaxation can be found via a column generation procedure. The column generation procedure prevents the a priori generation of all the possible operational paths. Also, the procedure allows the bounds to the relaxed problem to be found by solving a much simpler LP problem and a set of column generation subproblems. It should be highlighted that we have developed a bespoke dynamic programming that can solve each of the column generation subproblems in $\mathcal{O}(|E_2|)$ time, where E_2 is the set of edges in the underlying graph, $G_2(V_2, E_2)$, utilized to spatially and temporally track the modular production units.

Due to the computational difficulties that arise when solving medium-scale to large-scale instances we have developed a three-stage matheuristic. In the first stage of the matheuristic, a feasible solution to the problem is generated by solving a Lagrangian relaxation based heuristic for the respective formulation. In the second stage of the matheuristic, we utilize a path-relinking procedure to act as a local procedure by intensifying known quality solutions. In the third and final stage of the matheuristic, the Lagrangian multipliers are updated utilizing a sub-gradient procedure. These three phases continue in an iterative fashion until the time limit has been reached or a convergence tolerance has been reached.

The effectiveness of the matheuristic is illustrated through numerical experiments with a set of randomly generated test instances. For the large-scale test instances, the results show that the matheuristic with the path relinking procedure produces quality solutions an order of magnitude faster than a "state-of-the-art" MILP solver , which is evident in Fig. 8.b. Moreover, we found that the matheuristic often produce quality integer feasible solutions to the DMC-MPU with loose certificates of optimality before the "state-of-the-art" MILP has the opportunity to finish calculating its initial LP relaxation. Moreover, the matheuristic allows the decision maker to develop models for larger supply chains and to conduct sensitivity analysis on various parameters much faster than with a "state-of-the-art" MILP solver, which in turn reduces the time it will take the decision maker to make the distribution, production, and relocation decisions. In a future work, it would be beneficial to see how the proposed matheuristic preforms at the

tail when the time limit is extended 1,000 seconds to 10,000 seconds as well as parallelize the implementation of the path relinking procedure and the column generation procedure.

Funding

This work was supported by: (i) the National Science Foundation under grant no. 1739977 (INFEWS); (ii) the U.S. Department of Energy under RAPID SYNOPSIS Project (DE-EE0007888-09-03); (iii) Royal Dutch Shell; (iv) and the Texas A&M Energy Institute.

Present Addresses

R. Cory Allen is currently located at the ExxonMobil Upstream Research Company in Spring, TX, USA.

References

Y. Adulyasak, J.-F. Cordeau, and R. Jans. The production routing problem: A review of formulations and solution algorithms. *Computers & Operations Research*, 55:141–152, 2015.

A. Agra, H. Andersson, M. Christiansen, and L. Wolsey. A maritime inventory routing problem: Discrete time formulations and valid inequalities. *Networks*, 62(4):297–314, 2013.

E. Alarcon-Gerbier and U. Buscher. Modular and mobile facility location problems: A systematic review. *Computers & Industrial Engineering*, page 108734, 2022.

R. C. Allen, D. Allaire, and M. M. El-Halwagi. Capacity planning for modular and transportable infrastructure for shale gas production and processing. *Industrial & Engineering Chemistry Research*, 58(15):5887–5897, 2018.

R. C. Allen, S. Avraamidou, and E. N. Pistikopoulos. Production scheduling of supply chains comprised of modular production units. *IFAC-PapersOnLine*, 53(2):11452–11457, 2020.

R. C. Allen, F. Iseri, C. D. Demirhan, I. Pappas, and E. N. Pistikopoulos. Improvements for decomposition based methods utilized in the development of multi-scale energy systems. *Computers & Chemical Engineering*, page 108135, 2023.

A. Allman and Q. Zhang. Dynamic location of modular manufacturing facilities with relocation of individual modules. *European Journal of Operational Research*, 2020.

A. Allman, C. Lee, M. Martin, and Q. Zhang. Biomass waste-to-energy supply chain optimization with mobile production modules. *Computers & Chemical Engineering*, 150:107326, 2021.

C. Archetti and M. G. Speranza. A survey on matheuristics for routing problems. *EURO Journal on Computational optimization*, 2(4):223–246, 2014.

A. Arora, J. Li, M. S. Zantye, and M. F. Hasan. Design standardization of unit operations for reducing the capital intensity and cost of small-scale chemical processes. *AIChE Journal*, 66 (2):e16802, 2020.

M. Baldea, T. F. Edgar, B. L. Stanley, and A. A. Kiss. Modular manufacturing processes: Status, challenges, and opportunities. *AIChE journal*, 63(10):4262–4272, 2017.

J. F. Bard and N. Nananukul. A branch-and-price algorithm for an integrated production and inventory routing problem. *Computers & Operations Research*, 37(12):2202–2217, 2010.

C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3): 316–329, 1998.

T. Becker, S. Lier, and B. Werners. Value of modular production concepts in future chemical industry production networks. *European Journal of Operational Research*, 276(3):957–970, 2019.

T. Becker, B. Bruns, S. Lier, and B. Werners. Decentralized modular production to increase supply chain efficiency in chemical markets. *Journal of Business Economics*, 91(6):867–895, 2021.

A. K. Beheshti and S. R. Hejazi. A novel hybrid column generation-metaheuristic approach for the vehicle routing problem with general soft time window. *Information Sciences*, 316: 598–615, 2015.

G. Belvaux and L. A. Wolsey. Modelling practical lot-sizing problems as mixed-integer programs. *Management Science*, 47(7):993–1007, 2001.

A. Bhosekar and M. Ierapetritou. A framework for supply chain optimization for modular manufacturing with production feasibility analysis. *Computers & Chemical Engineering*, 145: 107175, 2021.

J. Bielenberg and I. Palou-Rivera. The rapid manufacturing institute–reenergizing us efforts in process intensification and modular chemical processing. *Chemical Engineering and Processing-Process Intensification*, 138:49–54, 2019.

I. I. Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46 (53):157, 2009.

G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

I. Dunning, J. Huchette, and M. Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017. doi: 10.1137/15M1020575.

R. Gronhaug, M. Christiansen, G. Desaulniers, and J. Desrosiers. A branch-and-price method for a liquefied natural gas inventory routing problem. *Transportation Science*, 44(3):400–415, 2010.

S. D. Jena, J.-F. Cordeau, and B. Gendron. Lagrangian heuristics for large-scale dynamic facility location with generalized modular capacities. *INFORMS Journal on Computing*, 29 (3):388–404, 2017.

M. Mourgaya and F. Vanderbeck. Column generation based heuristic for tactical planning in multi-period vehicle routing. *European Journal of Operational Research*, 183(3):1028–1041, 2007.

Y. Pochet and L. A. Wolsey. *Production planning by mixed integer programming*. Springer Science & Business Media, 2006.

P. Punyim, A. Karoonsoontawong, A. Unnikrishnan, and V. Ratanavaraha. A heuristic for the two-echelon multi-period multi-product location–inventory problem with partial facility closing and reopening. *Sustainability*, 14(17):10569, 2022.

S. Roy. Consider modular plant design. *Chemical Engineering Progress*, 113(5):28–31, 2017.

Y. Shao and V. M. Zavala. Modularity measures: Concepts, computation, and applications to manufacturing systems. *AIChE Journal*, 66(6):e16965, 2020.

K. S. Shehadeh. Distributionally robust optimization approaches for a stochastic mobile facility routing and scheduling problem. *arXiv preprint arXiv:2009.10894*, 2020.

A. Silva, D. Aloise, L. C. Coelho, and C. Rocha. Heuristics for the dynamic facility location problem with modular capacities. *European Journal of Operational Research*, 2020.

R. A. Skitt and R. R. Levary. Vehicle routing via column generation. *European Journal of Operational Research*, 21(1):65–76, 1985.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2022. URL `http://www.gurobi.com`.

P. Toth and D. Vigo. *The vehicle routing problem*. SIAM, 2002.