aircondMulti: A Multi-Product Extension of the aircond Model

David L. Woodruff
Graduate School of Management, UC Davis
`DLWoodruff@UCDavis.edu`

February 2026

Abstract

We describe `aircondMulti`, a multi-product extension of the `aircond` model documented in [1]. The extension introduces P products that share a single regular-time production capacity while maintaining independent demand streams, inventories, and overtime production. The model is implemented as an mpi-sppy scenario creator and supports bundling. See `https://github.com/DLWoodruff/aircond`.

1 Overview

The single-product `aircond` model [1] is a multi-stage stochastic production planning problem with regular-time and overtime production, inventory carry-forward, and optional start-up costs. The `aircondMulti` extension generalises this to P products ($P \geq 1$) that *share* a common regular-time capacity constraint while each product has its own demand process, inventory state, and overtime production.

The number of products and the degree of cost differentiation are controlled by two new parameters: `num_products` (P, default 2) and `cost_spread` (α, default 0.1). Product p (indexed from 0) pays costs that are a factor of $(1 + p\alpha)$ times the base regular-time and overtime costs.

2 Variables

We retain the notation of [1] with the addition of a product index $p \in \{0, \dots, P - 1\}$. At each stage t:

$x_{t,p} \in \mathbb{R}_{\geq 0}$	regular-time production of product p
$w_{t,p} \in \mathbb{R}_{\geq 0}$	overtime production of product p
$y_{t,p} \in \mathbb{R}$	inventory of product p (positive or negative)
$y^{+}_{t,p} \in \mathbb{R}_{\geq 0}$	positive part of $y_{t,p}$
$y^{-}_{t,p} \in \mathbb{R}_{\geq 0}$	negative part of $\lvert y_{t,p}\rvert$
$\delta_t \in \{0, 1\}$	start-up indicator (shared across products)

3 Parameters

All parameters from [1] are retained with the following modifications:

- K_t (regular-time capacity): now shared across all products, i.e. $\sum_p x_{t,p} \leq K_t$.
- C_t and O_t (base regular and overtime costs): product p pays $C_t(1 + p\alpha)$ and $O_t(1 + p\alpha)$, respectively.
- y_0 (beginning inventory) and D_1 (starting demand) are each divided equally across P products.
- $D_{t,p}$ (demand): each product has an independent demand stream; see Section 5.

Two new parameters are introduced:

- `num_products` (P): the number of products (default 2).
- `cost_spread` (α): the per-product cost increment factor (default 0.1).

4 Objective and Constraints

The stage-t cost for product p is

$$C_t(1+p\alpha)\,x_{t,p} + O_t(1+p\alpha)\,w_{t,p} + \begin{cases} I_t\,y^+_{t,p} + N_t\,y^-_{t,p} + Q_t(y^-_{t,p})^2, & t<T \\ N^{\text{last}}_T\,y^+_{T,p} + N_T\,y^-_{T,p}, & t=T \end{cases}$$

where $N^{\text{last}}_T < 0$ is a salvage value for leftover inventory in the final stage.

The total stage-t cost sums over products, plus an optional shared start-up cost:

$$\phi_t = \sum_{p=0}^{P-1} \left[C_t(1+p\alpha)\,x_{t,p} + O_t(1+p\alpha)\,w_{t,p} + \text{inventory cost}_{t,p}\right] + S_t\,\delta_t.$$

Constraints. For each stage t and product p:

$$\sum_{p=0}^{P-1} x_{t,p} \le K_t \tag{1}$$

$$y_{t-1,p} + x_{t,p} + w_{t,p} - y_{t,p} = D_{t,p} \tag{2}$$

$$y^+_{t,p} - y^-_{t,p} = y_{t,p} \tag{3}$$

$$M\,\delta_t \ge \sum_{p=0}^{P-1}(x_{t,p} + w_{t,p}) \tag{4}$$

Constraint (1) is the shared regular-time capacity. Constraint (2) is the per-product material balance ($y_{0,p} = y_0/P$). Constraint (4) links the shared start-up binary to total production across all products.

5 Demand Generation

Each product has an independent demand stream generated by the same Gaussian random walk as in [1], but with a per-product seed offset. The demand for product p at tree node with index ν is generated using seed

$$\text{seed} = s_0 + p \cdot 100{,}000 + b \cdot 100 + \nu$$

where s_0 is the base `start_seed`, b is the bundle index (zero for non-bundled problems), and ν is the tree-node index computed by `sputils.node_idx()`.

The large product offset (100 000) ensures that demand streams for different products never collide. The bundle offset (100) ensures that different bundles see distinct demand realisations even though mpi-sppy's bundling mechanism passes identical sub-tree branching factors to every bundle (see Section 6).

Starting demand for each product is D_1/P (deterministic at the root node).

6 Bundling Support

When mpi-sppy creates bundles for a tree with branching factors $[B_1, B_2, \ldots, B_T]$, the number of scenarios per bundle, s, is specified on the command line. If $s = B_2 \cdot B_3 \cdots B_T$ then the bundler passes sub-tree branching factors $[1, B_2, \ldots, B_T]$ to the scenario creator for each bundle. Because $B'_1 = 1$ in the sub-tree, all bundles would otherwise share identical node indices and therefore identical demand realisations.

To break this degeneracy, the scenario creator computes

$$b = \left\lfloor \texttt{scennum} \,/\, \textstyle\prod_i B'_i \right\rfloor$$

and adds $b \cdot 100$ to the demand seed. For the non-bundled case, all scenario numbers satisfy `scennum` $< \prod B_i$, so $b = 0$ and behaviour is unchanged.

7 mpi-sppy Interface

The module provides the standard mpi-sppy scenario-creator interface:

- `scenario_creator(sname, **kwargs)`: builds a Pyomo `ConcreteModel` for the given scenario name.
- `scenario_names_creator(num_scens, start=None)`: generates scenario name strings.
- `inparser_adder(cfg)`: registers all command-line arguments.
- `kw_creator(cfg)`: extracts keyword arguments from the configuration.

Nonant variables are `RegularProd[p]` and `OvertimeProd[p]` for all products at all stages except the last. Supplementary EF variables include `Inventory[p]` for each product and the optional `StartUp` binary.

8 Default Parameter Values

The module defines the following defaults (matching the `parms` dictionary in the source code):

Parameter	Default	Description
`num_products`	2	number of products
`cost_spread`	0.1	cost differentiation factor α
`starting_d`	200	total starting demand (split across products)
`BeginInventory`	200	total initial inventory (split across products)
`Capacity`	200	shared regular-time capacity K
`RegularProdCost`	1.0	base regular-time cost C
`OvertimeProdCost`	3.0	base overtime cost O
`InventoryCost`	0.5	per-unit holding cost I
`NegInventoryCost`	5.0	per-unit backorder cost N
`LastInventoryCost`	-0.8	final-stage salvage value
`sigma_dev`	40	demand std. deviation σ
`mu_dev`	0	demand mean deviation μ
`start_ups`	False	include start-up costs (MIP)
`StartUpCost`	300	start-up cost S
`start_seed`	1134	base random seed s_0
`QuadShortCoeff`	0	quadratic backorder coefficient Q

References

[1] D. L. Woodruff, B. C. Knight, X. Chen, and S. Cazaux. aircond: An Example for Optimization Under Uncertainty. `https://github.com/DLWoodruff/aircond`, 2022.

[2] B. Knueven, D. Mildebrath, C. Muir, J. P. Watson, and D. L. Woodruff. A Parallel Hub-and-Spoke System for Large-Scale Scenario-Based Optimization Under Uncertainty. *Mathematical Programming Computation*, 15:591–619, 2023.