Optimal Scheduling for Cross-Facility Workflows

XWF for CDT

Mark Asch

2026-01-27

Table of contents

Welcome

This (online) book-document contains a complete presentation of *optimal scheduling* applied to Exascale and post-Exascale workflows. These workflows combine data acquisition, data storage, data transfer and data analysis. The HPC components of such workflows can incorporate a diverse range of models, including partial differential equation solvers, algebraic solvers, AI/ML-based models, and data analytics components, all integrated into a single process (Ferreira da Silva et al. 2024), (Unat et al. 2025).

The objective here is to address the inherent cross-facility, multi-domain character of these workflows. This requires a very carefully-crafted theoretical foundation that can be readily generalized and extended to these contexts. The material covers both the theory and its application to practical use-cases, including real contexts with uncertainties emanating from different causes. To understand these well, numerous examples are provided in the form of python code snippets and jupyter notebooks. All of these are intgerated in a *digital twin*[1] of the underlying cyberinfrastructure, the so-called *digital continuum*.

The Continuum Digital Twin (CDT) is defined and described in the series of forthcoming papers (Garénaux-Gruau, Bodin, and Asch 2025a, 2025b, 2025c). For optimization and scheduling, there are numerous excellent references. Among these we point out particularly (Birge and Louveaux 2011), (Pinedo 2022), and (Powell 2022). Most of the codes are based on the wonderful `pyomo` framework (Bynum et al. 2021), (Hart, Watson, and Woodruff 2011) and (Postek et al. 2025). General background on exascale workflows can be found in (M. Asch et al. 2018).

[1] Known as the CDT

Finally, in (Mark Asch 2022) there are basic explanations of optimization, uncertainty quantification, inverse problems and their use for digital twins.

This book explains **all** the tools needed for formulating and implementing digital twins.

Author

Mark Asch is Emeritus Professor of the Université de Picardie Jules Verne, Mathematics department.

https://markasch.github.io/DT-tbx-v1/

https://github.com/markasch/

http://masch.perso.math.cnrs.fr/

mark.asch@u-picardie.fr

Citation

Asch, Mark. Optimal Scheduling for Cross-Facility Workflows. Online (2026) https://markasch.github.io/RCPSP4CDT/

```
@book{Asch2026
    title = {Optimal {S}cheduling for {C}ross-{F}acility {W}orkflows},
    author = {Asch, Mark},
    url = {https://markasch.github.io/RCPSP4CDT/},
    year = {2026},
    publisher = {Online}
}
```

License

This online book is frequently updated and edited. It's content is free to use, licensed under a Creative Commons licence, and the code can be found on GitHub. A physical copy of the book will be available at a later date.

References

Asch, Mark. 2022. *A Toolbox for Digital Twins: From Model-Based to Data-Driven*. Philadelphia, PA: Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9781611976977.

Asch, M, T Moore, R Badia, M Beck, P Beckman, T Bidot, F Bodin, et al. 2018. "Big Data and Extreme-Scale Computing: Pathways to Convergence-Toward a Shaping Strategy for a Future Software and Data Ecosystem for Scientific Inquiry." *The International Journal of High Performance Computing Applications* 32 (4): 435–79. https://doi.org/10.1177/1094342018778123.

Birge, John R., and François Louveaux. 2011. *Introduction to Stochastic Programming*. Second edition. Springer New York, NY. https://doi.org/10.1007/978-1-4614-0237-4.

Bynum, Michael L., Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff. 2021. *Pyomo–Optimization Modeling in Python*. Third. Vol. 67. Springer Science & Business Media.

Ferreira da Silva, Rafael, Rosa M. Badia, Deborah Bard, Ian T. Foster, Shantenu Jha, and Frederic Suter. 2024."Frontiers in Scientific Workflows: Pervasive Integration With High-Performance Computing ." *Computer* 57 (08): 36–44. https://doi.org/10.1109/MC.2024.3401542.

Garénaux-Gruau, Marius, François Bodin, and Mark Asch. 2025a. "Continuum Digital Twin Implementation." https://arxiv.org/abs/25xx.xxxx.

———. 2025b. "Continuum Digital Twin Use Cases." https://arxiv.org/abs/25xx.xxxx.

———. 2025c. "Continuum Digital Twin—Mathematical Models." https://arxiv.org/abs/25xx.xxxx.

Hart, William E, Jean-Paul Watson, and David L Woodruff. 2011. "Pyomo: Modeling and Solving Mathematical Programs in Python." *Mathematical Programming Computation* 3 (3): 219–60.

IEA, Paris. 2025. "Energy and AI." https://www.iea.org/reports/energy-and-ai.

Pinedo, Michael L. 2022. *Scheduling: Theory, Algorithms, and*

Systems. 6th edition. Springer Cham.

Postek, Krzysztof, Alessandro Zocca, Joaquim Gromicho, and Jeffrey Kantor. 2025. *Hands-On Mathematical Optimization with Python*. Cambridge University Press. https://doi.org/10.1017/9781009493512.

Powell, Warren B. 2022. *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions*. John Wiley & Sons.

Unat, Didem, Anshu Dubey, Emmanuel Jeannot, and John Shalf. 2025. "The Persistent Challenge of Data Locality in the Post-Exascale Era." *Computing in Science & Engineering* 27 (4): 19–27. https://doi.org/10.1109/MCSE.2025.3567586.

Introduction

A wide-scale, cross-facility workflow is a complex beast. It reunites users, jobs, and facilities, each with its resources and constraints. In the workflows that interest us, the facilites include data centres, HPC[2] centres, and the network connections between these.

The basic problem can be resumed as follows: find an optimal schedule S for a collection of jobs J to be executed on a set of facilities F, subject to constraints on resources, availability, precedences. The optimization can be performed for various objectives, or combinations of these, such as cost, project duration, facility availability, environmental impact. The presence of *uncertainty* plays a central role and is included in the optimization process.

The Exa-AtoW Project

This work is part of the Exa-AtoW project, a member of the NumPEx consortium. The Exa-AToW project aims at providing solutions for the efficient management of large-scale workflows composed of HPDA, AI, and HPC tasks that are distributed over a continuum of resources ranging from the Exascale, HPC, and Data infrastructures.

https://numpex.org

Exa-AToW focuses on effective end-to-end solutions, at scale, by considering not only functional dimensions such as workflows and data logistics but also resource federation governance, cybersecurity, energy, and sustainability.

[2]High Performance Computing

Continuum Digital Twin or Shadow

Given the inherent complexity of Exascale workflows, they will inevitably be executed on a cross-facility infrastructure. Such a multi-component basis will inevtibaly be subject to uncertainties in availability, maintenance, cost, etc. In addition, the cybersecurity constraints will impose strict access conditions that make workflow testing basically impossible. And, more recently, the issue of sustainability and energy consumption by cyberinfrastructure (IEA 2025) is a critical issue, in particular for so-called hyperscalers and data centers used for AI training and inference.

For all these reasons, the presence of a digital twin, or shadow, is indispensable for testing and planning workflow executions *before* actually executing them. The computer science setup, based on micro-services and ontologies, is fully described in (Garénaux-Gruau, Bodin, and Asch 2025a) and extended use-cases are presented in (Garénaux-Gruau, Bodin, and Asch 2025b).

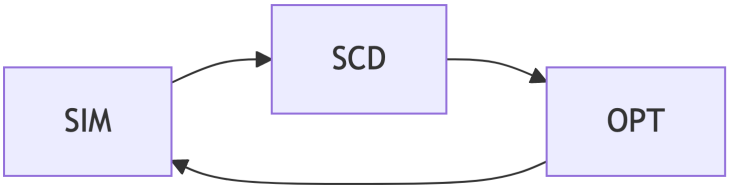

Figure 1: Three modules of the CDT: simuulation (SIM), scheduling (SCD), optimization (OPT). An initial workflow definition enters at the left and an optimal workflow is produced.

The core of the CDT is a set of three modules: simulation, scheduling and optimization, as shown in Figure 1. They can be used independently, or be chained together. These three are fed by a shared database, based on a common ontology that contains descriptions of all the jobs to perform, resources available, constraints to be respected, and any objectives to be attained. This database is connected to, and communicates with the real world—see Figure 2. This communication can take the form of a MADPP (machine actionable data project plan),

a user-interface, or a combination of the two (Garénaux-Gruau, Bodin, and Asch 2025a).

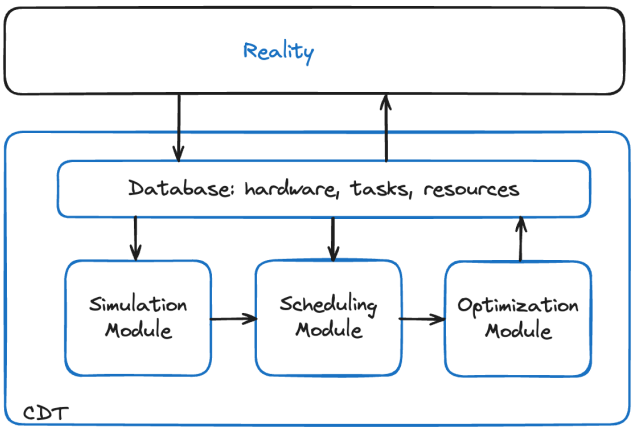

Figure 2: Global software architecture of the CDT, consisting of three modules: simulation (SIM), scheduling (SCD), optimization (OPT).

All details can be found in (Garénaux-Gruau, Bodin, and Asch 2025c).

Uncertainty

Deterministic schedules optimize for a world that, in theory, never materializes. Processing times vary, machines fail, networks are overloaded, demand shifts. As a result, a schedule that is "optimal" under perfect information often performs poorly when reality intervenes. Stochastic formulations, on the other hand, explicitly hedge against variability. The resulting schedules may sacrifice some of the theoretical efficiency, but maintain feasibility when *disruptions* occur, thus avoiding costly rescheduling or missed deadlines. Rather than a single makespan or cost figure, one can obtain distributions: "We meet the deadline with 95% confidence" which is far more informative than "the expected completion is Tuesday."

Hence, when uncertainty in job durations or data transfer arrivals is modeled, the true value of buffer capacity, parallel facilities, or overtime flexibility becomes visible. Deterministic

models systematically undervalue these. We can compute the value of the stochastic solution (VSS), which quantifies what can be gained by solving the full stochastic program rather than just optimizing against average conditions. Note that averaging yields a deterministic problem.

The Value of Stochastic Solution (VSS) is defined as the difference between the Expectation of the Expected Value Solution (EEVS) and the optimal objective value of the Recourse Problem (RP). The VSS quantifies the benefit of using a stochastic model in place of a deterministic one in decision-making problems under uncertainty. A large VSS signals that the system is sensitive to variability, hedging decisions matter, and the deterministic approximation is potentially dangerous.

In conclusion, when faced with expensive (financially and environmentally), time-consuming Exascale workflows, the need to consider uncertainty in the scheduling program is vital. Stochastic-based scheduling can take into account any uncertain resources and all uncertain costs—a good example would be variable energy costs. The solution thus obtained will permit a *hedging* strategy. We can, in addition, perform risk analysis, where we compare alternative deployments of the workflow as a function of our *risk profile*.

$$\mathrm{VSS} = \mathrm{EEV} - \mathrm{RP}$$

Theory

We propose a *hierarchical* model for the Continuum Digital Twin[3], composed of a Supply Chain Network Design[4] coupled with a Resource Constrained Process Scheduling[5]. In such a hierarchical system, the supply chain model allocates jobs to the best facilities, and the scheduling model then (optimally) sequences jobs within a given facility. Supply chain optimization emphasizes "where" (facility location, data transfer mode, sourcing). Scheduling emphasizes "when" and "in what order."

Recall the basic problem: given a directed graph of N storage and processing jobs $\mathcal{J}$, find an optimal allocation A and a feasible schedule S. Further details below.

$\mathcal{J} = \{J_1, J_2, ... J_N\}$,

Supply Chains and Scheduling

Let us begin by defining supply chains and scheduling in the context of the CDT.

Definition 0.1 (Supply Chain)**.** A *supply chain* is a network (directed graph) of jobs, facilities, data storage, networks, processing and end-product delivery to users. The design poroblem is to optimally allocate an ordered list of data storage and data processing jobs to an optimal choice of storage and processing centres.

Definition 0.2 (Scheduling)**.** *Scheduling* is a decision-making process that deals with the allocation of (limited) resources to tasks over given time periods. Its goal is to optimize one or more

[3]CDT
[4]SCND
[5]RCPSP

objectives. The resulting *schedule* is a job sequence determined for every machine (facility) of the processing system.

These two models are complementary:

- the SCND can be used for global, over a *fixed period*, modelling—for example annualized;
- the RCPSP can be used for time-dependent models, especially when uncertainty is introduced, as in a two-stage, or multi-stage, model—see below Section **?@sec-stoch**.

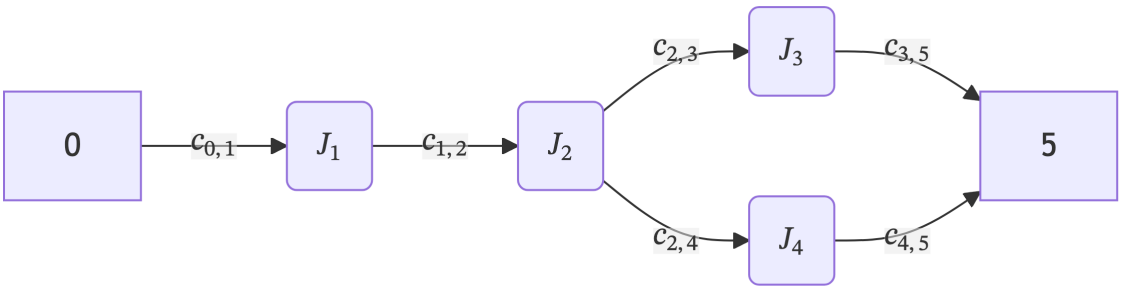

Figure 3: Simple directed graph (DAG) for a genomics workflow with 4 processing jobs $J_1, \cdots, J_4$, and data transfers $c_{i,j}$ between them, including initial upload and final download. The tasks 0 and 5 are dummy tasks, representing the start and the end of the workflow.

Supply Chain Network Design

Problem Definition

Let $\mathcal{J}$, $\mathcal{F}$, and $\mathcal{U}$ be respective (finite) sets of jobs and datasets, HPC and datacentres (facilities), and end-users. The union

$$\mathcal{N} \doteq \mathcal{J} \cup \mathcal{F} \cup \mathcal{U}$$

of these sets is viewed as the set of nodes of a directed graph $(\mathcal{N}, \mathcal{A})$, where $\mathcal{A}$ is a set of arcs (directed links) connecting these nodes in a way representing flow of the jobs—see Figure 3 and Figure 4. The processing facilities $\mathcal{F}$ can include HPC centres

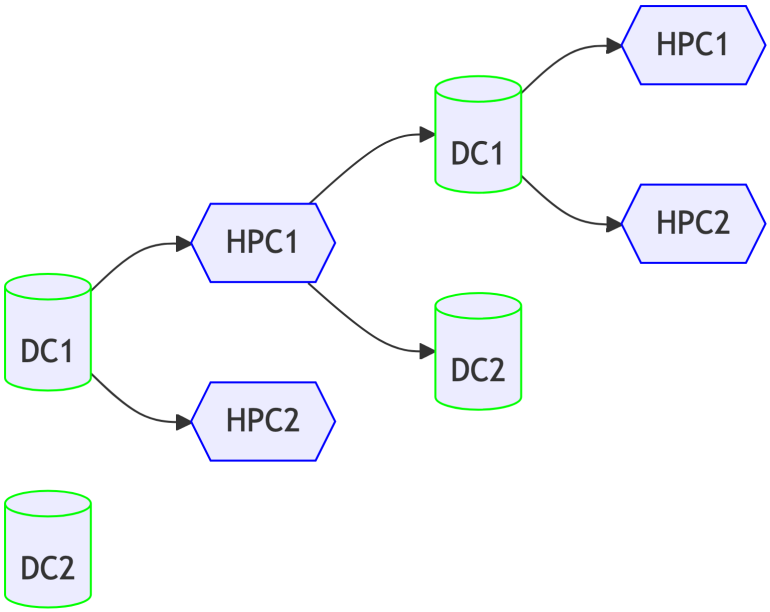

Figure 4: Supply chain model for the CDT. For a given job, the dataset (stored in one of two Datacentres, DC_1 or DC_2) can be processed on one of two HPC centres (HPC_1 or HPC_2), and ouput data is sent to (and stored on) one of the two Datacentres, ready as input for the next job. This is repeated for each job in the ordered job list (DAG).

$\mathcal{H}$, data storage centres $\mathcal{D}$, and post-processing centres $\mathcal{P}$, i.e. $\mathcal{F} = \mathcal{H} \cup \mathcal{D} \cup \mathcal{P}$. Furthermore, HPC ($i \in \mathcal{H}$), data ($i \in \mathcal{D}$), and post-processing ($i \in \mathcal{P}$) centers can each be composed of sets of individual machines, $\mathcal{M}_i$ and the set $\mathcal{F}$ includes the processing centers as well as the machines in each center. Finally, let $\mathcal{K}$ be the set of jobs flowing through the "supply chain,'' the digital continuum in our case.

State Variables

The state of the system, or supply chain, at a given time t, is represented by state variables and coefficients:

- the available (open/used, or closed/unused) facilities:
  - compute centres $x_i^C \in \{0, 1\}$, $i = 1, \ldots, N_C$
  - data centres $x_j^D \in \{0, 1\}$, $j = 1, \ldots, N_D$
- the quantity/number of jobs, or datasets, or volumes transferred
  - y_{ji} from instrument (data centre) j to compute centre i,
  - y_{ij} from compute centre i to data centre j,
  - y_{ik} from compute centre i to user k,
  - y_{kj} from user k to data centre j,
- costs—both fixed and variable (eventually stochastic[6])—are associated with each of the decision variables;
- resource supply/availability and demand/requests are specified for each facility, and eventually for each machine in a given facility—this can includes the ephemeral buffers (Garénaux-Gruau, Bodin, and Asch 2025a).

Objective Function

The overall objective is to combine location decisions—which centres to use—with allocation decisions—how to distribute

[6] Either a discrete set or tabulated values drawn from a known probability distribution, or the PDF itself.

the workloads and jobs among the chosen centres (data and compute).

There can be a *single* objective such as minimizing (any function of) the total of fixed and variable costs, or minimizing the completion time (makespan). *Multiple* objective optimization[7] seeks a tradeoff between minimum cost and maximum sustainability (minimum environmental impact). Finally, *stochastic* optimization takes into account the uncertainties of resource availabilities and delays, maintenance and failures, resource allocations, variable energy costs, project costs (HR, budget).

The overall, deterministic cost function for total cost can be expressed as a sum (some terms can be ignored by setting the coefficients to zero, depending on the context)

$$\min_{x,y} \sum_{i=1}^{N_C} f_i^C x_i^C + \sum_{j=1}^{N_D} f_j^D x_j^D + \sum_{i=1}^{N_C}\sum_{j=1}^{N_D} c_{ji} y_{ji} + \sum_{j=1}^{N_D}\sum_{i=1}^{N_C} c_{ij} y_{ij} + \sum_{i=1}^{N_C}\sum_{k=1}^{N_U} c_{ik} y_{ik} + \sum_{k=1}^{N_U}\sum_{j=1}^{N_D} c_{kj} y_{kj}$$

- f_i^C, f_j^D are the fixed costs of using compute centre i, data centre j;
- c_{ji}, c_{ij}, c_{ik}, c_{kj} are the variable costs of the job "flow" from data centre j to compute centre i, etc. along the arc $(j, i) \in \mathcal{A}$, etc.
- y_{ji}, y_{ij}, y_{ik}, y_{kj} are the corresponding quantities "flowing" from data centre j to compute centre i, etc. along the arc $(j, i) \in \mathcal{A}$, etc.
- x_i, x_j are binary variables, indicating the choice/use of a particular compute or data centre;
- N_U is the number of end-users;
- weights, or priorities, can be attributed to each term, to either balance the contributions, or to give greater importance to certain terms.

Configuration Decisions

We suppose that supply chain configuration decisions consist of deciding which of the processing centers to engage, and

[7] MOO

eventually which processing and storage machines to use within each center.

Dynamic Networks

We can generalize the supply chain by considering a multi-stage network with an added time dimension. This enables flow and carrying of data and jobs across time periods, in addition to flow among facilities. For example, if certain facilites become unavailable, or less available, as the project evolves over time. This is a kind of recourse, which will be considered below in Section **?@sec-stoch**.

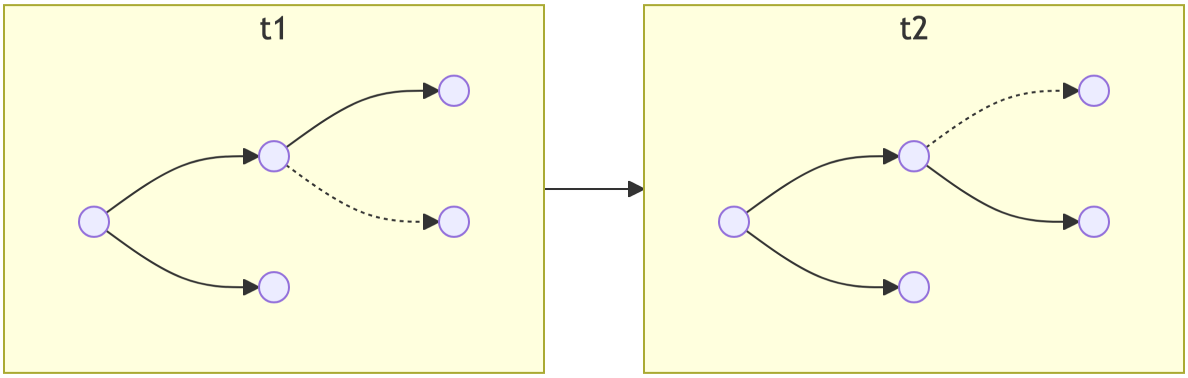

Figure 5: Multi-stage, dynamic network that evolves over time due to changing availability constraints.

Mathematical Formulation - Deterministic Model

We begin with a deterministic formulation, and then introduce uncertainties to obtain a full stochastic model.

Assign a binary variable $x_i = 1$ if the processing facility i is engaged or machine i is used, and $x_i = 0$ otherwise. Operational decisions then consist of routing the flow of all the jobs $k \in \mathcal{K}$ from the instrument—or primary data reposirory—to the end-user, in a cost-optimal way, still to be defined.

Let y_{ij}^k denote the flow of job k from a node i to a node j of the network, where $(i, j) \in \mathcal{A}$ is an arc of the network. The flow

will normally be a voluminous data transfer, possibly by even physically tansporting disk drives from remote locations.

The optimization problem—objective function and constraints—can be written as follows. It covers a very general case, with both optimal allocation and eventual resource constraints.

$$\min_{x,y} \sum_{i\in\mathcal{F}} c_i x_i + \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} q_{ij}^k y_{ij}^k \tag{1}$$

$$\text{s.t.} \sum_{i\in\mathcal{N}} y_{ij}^k - \sum_{\ell\in\mathcal{N}} y_{j\ell}^k = 0, \quad j\in\mathcal{F},\ k\in\mathcal{K}, \tag{2}$$

$$\sum_{i\in\mathcal{N}} y_{ij}^k \geq d_j^k, \quad j\in\mathcal{U},\ k\in\mathcal{K}, \tag{3}$$

$$\sum_{i\in\mathcal{N}} y_{ij}^k \leq s_j^k, \quad j\in\mathcal{I},\ k\in\mathcal{K}, \tag{4}$$

$$\sum_{k\in\mathcal{K}} r_j^k \sum_{i\in\mathcal{N}} y_{ij}^k \leq m_j x_j, \quad j\in\mathcal{F}, \tag{5}$$

$$x\in\mathcal{X}, \quad y\geq 0, \tag{6}$$

- in (1), c_i is the cost of engaging facility i or using machine i, and q_{ij}^k is the per unit cost of processing job k at facility i and/or sending job k on arc $(i,j)\in\mathcal{A}$, (data transfer, transport and data storage costs)
- in (3), d_j^k is the "demand" of job k at node j, (eg. how many CPU's or how much data storage capacity is required)
- in (4), s_j^k is the "supply" of job k at node j, (eg. how many CPU's or how much data storage capacity is available)
- in (5), r_j^k is the per-unit processing requirement for job k at node j, and m_j is the capacity of facility j,
- in (6), $\mathcal{X}\subset\{0,1\}^{|\mathcal{F}|}$, $y\in\mathbb{R}^{|\mathcal{A}|\times|\mathcal{K}|}$ is a vector with components y_{ij}^k,
- all cost components are defined per given period (hour, day, week, month, etc.)

The above equations have the following interpretations.

- Objective function (1) minimizes the total of investment[8] plus operational[9] costs.

[8] CAPEX
[9] OPEX

- $\mathcal{K}$ represents logical dependencies and restrictions, such as $x_i \leq x_j$ for all $i \in \mathcal{M}_j$, where $j \in \{\mathcal{H}, \mathcal{D}, \mathcal{P}\}$, which means that machine i should be used only if facility j is engaged, and since the x's are binary the constraint $x_i \leq x_j$ implies that $x_i = 0$ if $x_j = 0$.
- Constraints (2) enforce the flow conservation of job k across each processing node j.
- Constraints (3) require that the total flow of job k to a customer node j should exceed the "demand" d_j^k at that node.
- Constraints (4) require that the total flow of job k from an instrument node j should be less than the "supply" s_j^k at that node.
- Constraints (5) enforce capacity constraints of the processing nodes.
  - Require that the total processing requirement of all jobs flowing into a processing node j should be smaller than the capacity m_j of facility$j$ if it is used ($x_j = 1$).
  - If facility j is not used ($x_j = 0$), the constraint will force all flow variables $y_{ij}^k = 0$ for all $i \in \mathcal{N}$.
- Finally, constraint (6) enforces the feasibility constraint $x \in \mathcal{X}$ and the non-negativity of the flow variables corresponding to an arc $(i, j) \in \mathcal{A}$, and job $k \in \mathcal{K}$.

We can rewrite the above in a very compact, matrix-vector form.

$$\begin{aligned} \min\ & c^\top x + q^\top y \\ \text{s.t. } & Ny = 0, \\ & Cy \geq d, \\ & Sy \leq s, \\ & Ry \leq Mx, \end{aligned}$$

where

- vectors c, q, d, and s represent fixed costs, processing or transmission costs, demands, supplies, respectively;
- matrices N, C, and S correspond to summations of the respective expressions, (nodes, demands, supplies);

- R is the matrix of r_j^k (requirements);
- $M =$ diag (m_j) (machine capacities).

Simplified Cases

The above formulation covers both the cost-optimal streaming of jobs across the cyberinfrastructure as well as the respect of individual facilities' constraints. The latter is a resource-constrained scheduling problem.

Unconstrained-Resource Problem

As a first simplification, we suppose that the facilites possess the necessary capacities to service the requirements of all the incoming data storage and processing jobs. Suppose that we still have fixed and variable costs, so the objective does not change. The "infinite-resource" problem simplifies to:

$$\min_{x,y} \sum_{i\in\mathcal{F}} c_i x_i + \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} q_{ij}^k y_{ij}^k \tag{1a}$$

$$\text{s.t.} \sum_{i\in\mathcal{N}} y_{ij}^k - \sum_{\ell\in\mathcal{N}} y_{j\ell}^k = 0, \quad j\in\mathcal{F},\ k\in\mathcal{K}, \tag{2a}$$

$$x\in\mathcal{X}, \quad y\geq 0, \tag{6a}$$

Resource-Constrained Project Scheduling Problem (RCPSP)

A second simplification is to consider the resource constraint problem, without facility or machine allocations. The Resource-Constrained Project Scheduling Problem (RCPSP) is a combinatorial optimization problem that consists of finding a feasible scheduling for a set of n jobs subject to resource and precedence constraints. Each job has a processing time, a set of successor jobs and a required amount of different resources. Resources may be scarce but are renewable at each time period. Precedence constraints between jobs mean that no jobs may start before all its predecessors are completed. The jobs must be scheduled non-preemptively, i.e., once started, their processing cannot be interrupted.

The RCPSP has the following input data:

- $\mathcal{J}$ set of jobs.
- $\mathcal{R}$ set of renewable resources.
- $\mathcal{S}$ set of precedences between jobs $(i, j) \in \mathcal{J} \times \mathcal{J}$.
- $\mathcal{T}$ planning horizon: set of possible processing times for jobs.
- p_j processing time of job j.
- $u_{(j,r)}$ amount of resource r required for processing job j.
- c_r capacity of renewable resource r.

A binary programming formulation was proposed by Pritsker et al. in 1986. In this formulation, decision variables $x_{jt} = 1$ if job j is assigned to begin at time t; otherwise, $x_{jt} = 0$. All jobs must finish in a single period of time without violating precedence constraints while respecting the amount of available resources. The model proposed by Pristker can be stated as follows:

$$
\begin{aligned}
\text{Minimize} & \sum_{t \in \mathcal{T}} t \cdot x_{(n+1,t)} \\
\text{Subject to:} & \sum_{t \in \mathcal{T}} x_{(j,t)} = 1 \ \ \forall j \in J, \\
& \sum_{j \in J} \sum_{t_2 = t - p_j + 1}^{t} u_{(j,r)} x_{(j,t_2)} \leq c_r \ \ \forall t \in \mathcal{T}, r \in R, \\
& \sum_{t \in \mathcal{T}} t \cdot x_{(s,t)} - \sum_{t \in \mathcal{T}} t \cdot x_{(j,t)} \geq p_j \ \ \forall (j, s) \in S, \\
& x_{(j,t)} \in \{0, 1\} \ \ \forall j \in J, t \in \mathcal{T}.
\end{aligned}
$$

Mutli-Project Multi-Mode RCPSP

Note that the above formulation is restricted to temporal scheduling on a *single* machine, and for a single project, or collection of jobs. This formulation has been generalized to deal with multiple machines and multiple projects. In this case, the formulation becomes very similar to that of a supply chain. It is referred to in the literature as the *multi-mode resource-constrained multi-project scheduling problem*, or MRCMPSP. We can then perform simultaneous scheduling of a set of multiple projects taking into account the availability of local and global resources under different time and resource constraints. This has practical importance, at national and

European levels, when cross-facility implies exploitation of cyberinfrastructure resources across different countries, for example, as would be the case for EuroHPC[10], at the European level, or GENCI[11] for France. Imagine being able to plan and pilot multiple exascale projects, on multiple sites, over multiple countries...[12]

This general formulation is then mathematically equivalent to the supply chain configuration problem[13] with resource constraints. The loop is closed.

Optimization

In summary, this book has no content whatsoever. In summary, this book has no content whatsoever.

MILP formulations for RCPSP and SCND

We consider single- and multi-mode resource-constrained project scheduling problems. In summary, this book has no content whatsoever.

Precedence Constraints

In summary, this book has no content whatsoever. In summary, this book has no content whatsoever.

Disjunctions

In summary, this book has no content whatsoever. In summary, this book has no content whatsoever.

[10] https://www.eurohpc-ju.europa.eu/
[11] https://www.genci.fr//
[12] These would be multiple states, in the USA context.
[13] SCCP

Uncertainty

In practice some of the scheduling parameters may be uncertain. The exact duration of an activity, for instance, might not be known at the beginning of the project. Similarly, the number of available resources is another parameter that may not be known before project execution. These uncertainties may be due to different sources, including estimation errors, unforeseen (weather) conditions, late "delivery" (unavailability) of some required resources, unpredictable incidents such as machine breakdown or worker accidents, etc.

Examples

In summary, this book has no content whatsoever.

Examples

In summary, this book has no content whatsoever.

```
import matplotlib.pyplot as plt
plt.plot([1,23,2,4])
plt.show()
```

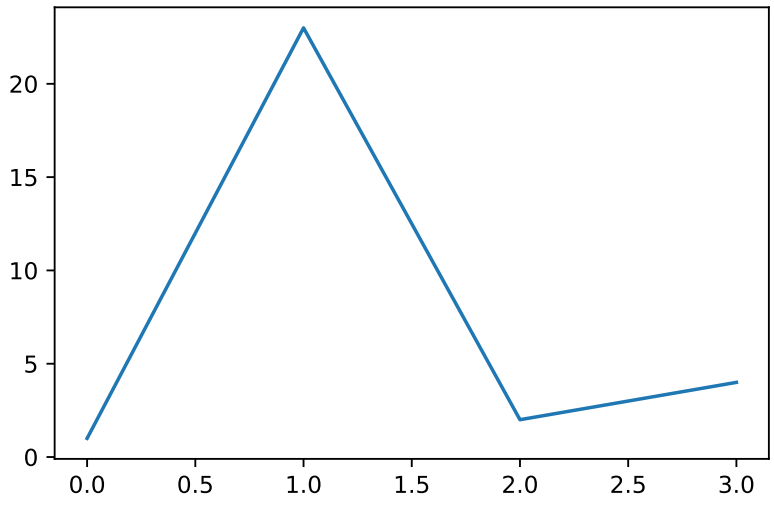

Figure 6: A line plot

We observe in Figure 6 that there is a clear trend, up and down.

Use-Cases

Setup and Linear Programming Example

As an introduction we load the necessary `pyomo` packages and solve an ultra-simple linear programming example.

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & 3x_1 + 4x_2 \geq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- `pyomo.environ` provides the framework for building the model
- `SolverFactory` allows to call the solver used to solve the optimization problem

```
import pyomo.environ as pyo
from pyomo.opt import SolverFactory

import pyomo.environ as pyo

model = pyo.ConcreteModel("Simple Linear")
# Decision variables and domains
model.x = pyo.Var([1,2], domain=pyo.NonNegativeReals)
# Objective function
model.OBJ = pyo.Objective(expr = 2*model.x[1] + 3*model.x[2])
# Constraint(s)
model.Constraint1 = pyo.Constraint(expr = 3*model.x[1] + 4*model.x[2] >= 1)

solver = 'appsi_highs'
SOLVER = pyo.SolverFactory(solver)
assert SOLVER.available(), f"Solver {solver} is not available."

# Solve and print solution
SOLVER.solve(model)
print(f"x = ({pyo.value(model.x[1]):.2f}, {pyo.value(model.x[2]):.2f})")
print(f"optimal value = {pyo.value(model.OBJ):.2f}")
```

```
x = (0.33, 0.00)
optimal value = 0.67
```

We can print out

- the complete model;
- a full trace of the optimization process.

(use collapsed display here - click to expand and view output)

Precedence Example

We illustrate how to set up a *precedence table*, made up of job pairs that define pairwise precedences.

Disjunctions

When faced with a choice among exclusive options, we resort to disjunctive programming, based on:

- Big-M formulation.
- Logical formulation.

Simplified Supply Chain Example

Putting together:

- precedence
- TBC

Simplified RCPSP Example

Putting together:

- precedence
- TBC