



UNIVERSITÀ DI PISA

ML for Combinatorial Optimization

Mauriana Pesaresi Seminar Series - PhD Program in Computer Science

Andrea Cardia, 17/04/2026



Outline



- ▶ Examples of Combinatorial Problems
- ▶ Column Generation for CSP
- ▶ ORQ: How could ML speed the execution?



Vehicle Routing Problem

- ▶ A fleet of m vehicles must serve a group of n customers, departing from a depot and returning to the depot
- ▶ By minimizing a certain cost, typically the total distance or the total transportation cost
- ▶ Generalization of the Traveling Salesman Problem (TSP)



Sets

- ▶ $V = \{0, 1, 2, \dots, n\}$ Nodes
- ▶ $N = \{1, 2, \dots, n\}$ Customers
- ▶ $K = \{1, 2, \dots, m\}$ Vehicles

Indices

- ▶ $i, j \in V$
- ▶ $k \in K$

Parameters

- ▶ c_{ij} cost to go from i to j
- ▶ d_i customers demands
- ▶ Q vehicle capacities (CVRP)

Decision Variables

- ▶ $x_{ij}^k = \begin{cases} 1, & \text{if vehicle } k \text{ traverses arc } (i, j) \\ 0, & \text{otherwise} \end{cases}$
- ▶ u_i auxiliary variables used for MTZ constraints

Vehicle Routing Problem

- The mathematical model

$$\min \sum_{k \in K} \sum_{i \in V} \sum_{j \in V, j \neq i} c_{ij} x_{ij}^k$$

- A huge number of possible routes
- But only a small fraction of them are actually useful!

$$\sum_{k \in K} \sum_{j \in V, j \neq i} x_{ij}^k = 1, \forall i \in N$$

$$\sum_{j \in N} x_{0j}^k \leq 1, \forall k \in K$$

$$\sum_{i \in N} x_{i0}^k = \sum_{j \in N} x_{0j}^k, \forall k \in K$$

$$\sum_{j \in V, j \neq i} x_{ij}^k = \sum_{j \in V, j \neq i} x_{ji}^k, \forall i \in N, \forall k \in K$$

$$\sum_{i \in N} d_i \sum_{j \in V, j \neq i} x_{ij}^k \leq Q, \forall k \in K$$

$$u_i - u_j + Q \sum_{k \in K} x_{ij}^k \leq Q - d_j, \forall i \neq j$$

$$d_i \leq u_i \leq Q, \forall i \in N$$

$$x_{ij}^k \in \{0,1\}, \forall i, j \in V, i \neq j, \forall k \in K$$

$$u_i \geq 0, \forall i \in N$$

Crew Scheduling Problem

- ▶ We have a set of flights
- ▶ Each flight has to be covered by a crew
- ▶ Each crew has to satisfy compatibility constraints among flights
- ▶ Legal constraints, connection times, rest requirements (etc.)
- ▶ Objective: to build complete **pairings** with minimum total cost
- ▶ Set Partitioning Problem (SPP)



Sets

- ▶ $F = \{1, 2, \dots, n\}$ Flights
- ▶ P : Set of all feasible pairings

Parameters

- ▶ c_p cost of the pairing $p \in P$
- ▶ $a_{fp} = \begin{cases} 1, & \text{if pairing } p \text{ covers flight } f \\ 0, & \text{otherwise} \end{cases}$

Decision Variables

- ▶ $x_p = \begin{cases} 1, & \text{if pairing } p \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$

Crew Scheduling Problem

- The mathematical model



$$\min \sum_{p \in P} c_p x_p$$

$$\sum_{p \in P} a_{fp} x_p = 1, \forall f \in F$$
$$x_p \in \{0,1\}, \forall p \in P$$

- Very simple model!
- The real problem is $|P|$
- How big is $|P|$?

Crew Scheduling Problem

- ▶ Simple Example with $|F| = 4$

- ▶ $|P|$ is the number of valid sequences

- ▶ Length 1:

A, B, C, D 📌 4

- ▶ Length 2:

Permutations of 2 elements

AB, AC, AD

BA, BC, BD

CA, CB, CD

DA, DB, DC 📌 12

- ▶ Length 3:

Permutations of 3 elements

ABC, ACB, ABD, ADB, ACD, ADC

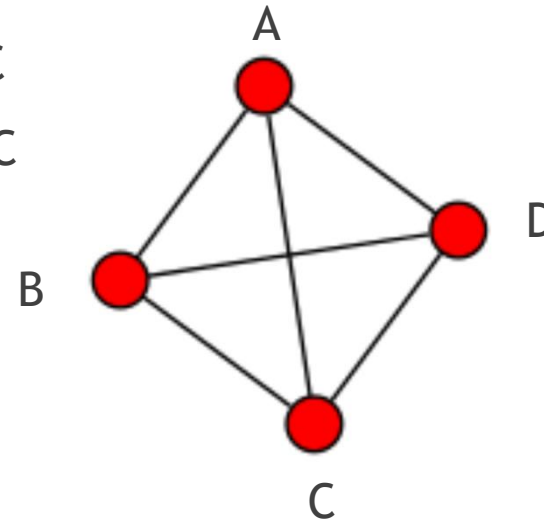
BAC, BCA, BAD, BDA, BCD, BDC

... 📌 24

- ▶ Length 4:

All the permutations

$4! = 24$



- ▶ Total: $4 + 12 + 24 + 24$

- ▶ In general $|P| = O\left(\sum_{k=1}^{|F|} \frac{|F|!}{(|F|-k)!}\right) = O\left(|F| + |F|(|F|-1) + \dots + \frac{|F|!}{2} + |F|! + |F|!\right)$
 $|P| = O(|F|!)$

Crew Scheduling Problem

- ▶ In practice, how can we build a pairing? (Just an intuition)
- ▶ Compatibility constraints - a pairing must comply with:
 - time compatibility (a flight must depart after the previous one has arrived)
 - basic requirements (departure and arrival at the same airport)
 - flight hour limits
 - limits on the number of traits
 - rest periods
- ▶ Geographic and temporal structure of the CSP $\rightarrow |P|$ becomes exponential in $|F|$

Column Generation Algorithm (applied to the CSP)

- ▶ It avoids generating all pairings at the start
- ▶ Each column corresponds to one feasible pairing

$$\min \sum_{p \in P} c_p x_p$$

Becomes

$$\min \sum_{p \in P'} c_p x_p$$

$$\sum_{p \in P} a_{fp} x_p = 1, \forall f \in F$$

$$x_p \in \{0,1\}, \forall p \in P$$

$$\sum_{p \in P'} a_{fp} x_p = 1, \forall f \in F$$

$$x_p \geq 0, \forall p \in P'$$

- ▶ A **primal solution** (values of x_p)
- ▶ A set of **dual variables**, one for each flight constraint: $\pi_f, \forall f \in F$

Column Generation Algorithm (applied to the CSP)

- ▶ How do we enlarge P' ?

- ▶ Pricing Problem

c_p : Real cost of the pairing

π_f : Price of covering the flight

$\sum_{f \in F} \pi_f a_{fp}$: Benefit of covering the set of flights

Definition (Reduced Cost):

$$\bar{c}_p = c_p - \sum_{f \in F} \pi_f a_{fp}$$

- ▶ $\bar{c}_p < 0$: Convenient pairing
- ▶ $\bar{c}_p \geq 0$: Does not improve the solution
- ▶ Pricing Problem: $\min \bar{c}_p$ (Shortest Path Problem - with RC)

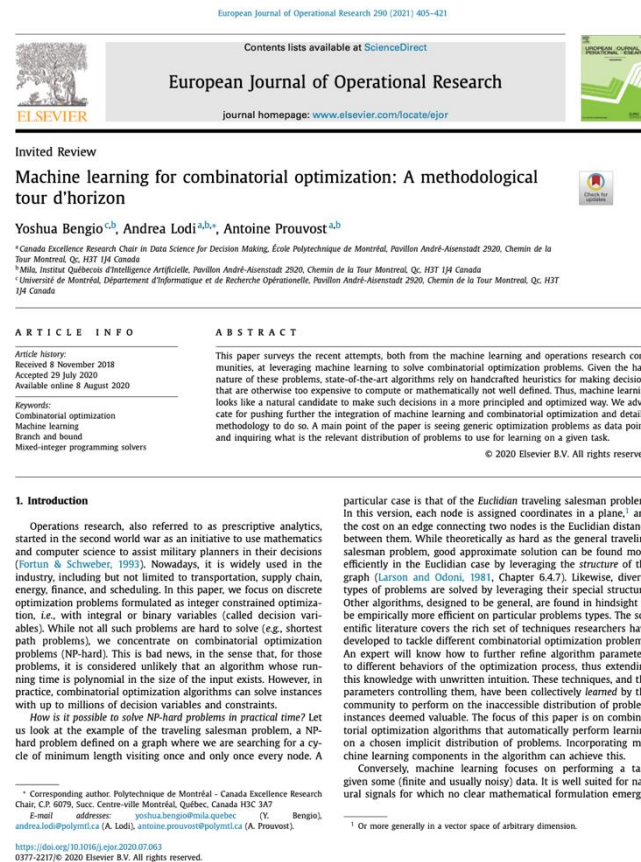


Other fields of application of CG

- ▶ Unit Commitment (Energetic Field)
- ▶ Portfolio Optimization (Quantitative Finance)
- ▶ Cutting Stock (Manufacturing Industry)
- ▶ Graph Coloring (Telecommunications)
- ▶ Crew Rostering (Transportation)

ORQ: How could ML speed the execution?

- ▶ ML + CO: ML does not replace algorithms, but learns their key components
- ▶ Two paradigms: end-to-end VS learning within optimization
- ▶ Most effective approach: ML integrated into OR algorithms (Branch & Bound, CG)
- ▶ In Column Generation: focus on the Pricing Problem
- ▶ ML used to predict promising columns and reduced costs
- ▶ Benefit: drastic reduction in computation time while maintaining quality



Arcs Selection in the PP Graph

- ▶ CG often dominated by PP
- ▶ Idea: learn to prune arcs in the pricing graph
- ▶ Formulate arc selection as **binary classification**
- ▶ Keep only arcs used in generated columns ($\approx <20\%$)
- ▶ Build reduced graph $\rightarrow$ faster shortest path computation
- ▶ Achieve **up to 40% speed-up** with negligible loss of quality

Machine-learning-based arc selection for constrained shortest path problems in column generation

Mouad Morabit^{1,2,3}, Guy Desaulniers^{1,2}, and Andrea Lodi^{1,2,3,4}

¹Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7

²GERAD, Montréal (Québec), Canada, H3T 2A7

³Canada Excellence Research Chair in Data Science for Real-Time Decision-Making, Polytechnique Montréal

⁴Jacobs Technion - Cornell Institute, Cornell Tech and Technion - IIT
mouad.morabit@gerad.ca
guy.desaulniers@gerad.ca
andrea.lodi@cornell.edu

Abstract

Column generation is an iterative method used to solve a variety of optimization problems. It decomposes the problem into two parts: a master problem, and one or more pricing problems (PP). The total computing time taken by the method is divided between these two parts. In routing or scheduling applications, the problems are mostly defined on a network, and the PP is usually an NP-hard shortest path problem with resource constraints. In this work, we propose a new heuristic pricing algorithm based on machine learning. By taking advantage of the data collected during previous executions, the objective is to reduce the size of the network and accelerate the PP, keeping only the arcs that have a high chance to be part of the linear relaxation solution. The method has been applied to two specific problems: the vehicle and crew scheduling problem in public transit and the vehicle routing problem with time windows. Reductions in computational time of up to 40% can be obtained.

Keywords : Column generation, machine learning, arc selection, heuristic pricing, vehicle routing and crew scheduling.



Thank you for your attention!

