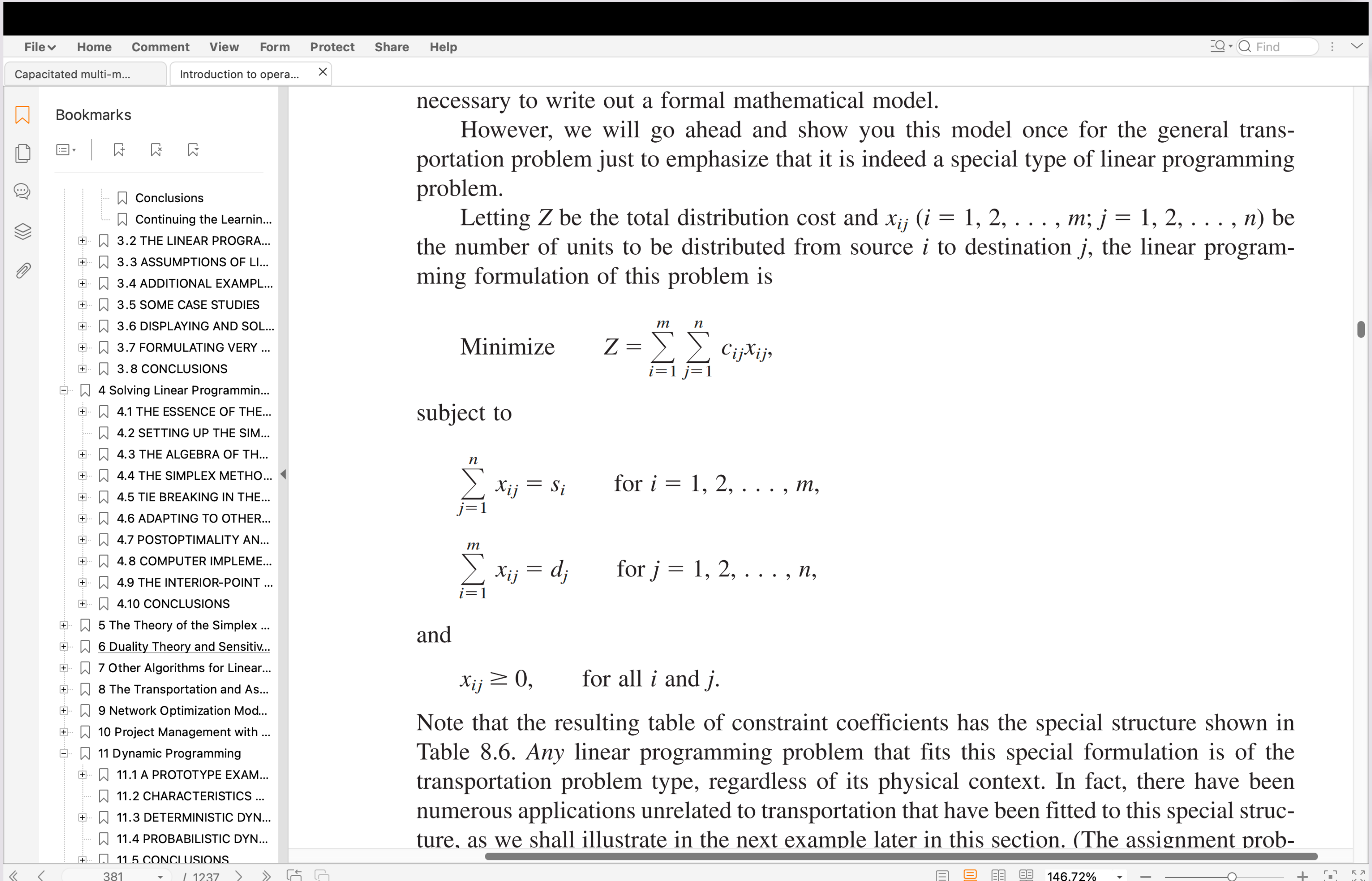
Operations Research Models and Algorithms

Transportation – Hungarian algorithm

- Subtract min value from all values in **row**
- Subtract min value from all values in **column**
- Zero assignment (select unique cell with zero)
- Calculate cost
- Source

LP of transportation problem



necessary to write out a formal mathematical model.

However, we will go ahead and show you this model once for the general transportation problem just to emphasize that it is indeed a special type of linear programming problem.

Letting Z be the total distribution cost and x_{ij} ($i = 1, 2, \dots, m$; $j = 1, 2, \dots, n$) be the number of units to be distributed from source i to destination j, the linear programming formulation of this problem is

$$\text{Minimize} \qquad Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij},$$

subject to

$$\sum_{j=1}^{n} x_{ij} = s_i \qquad \text{for } i = 1, 2, \dots, m,$$

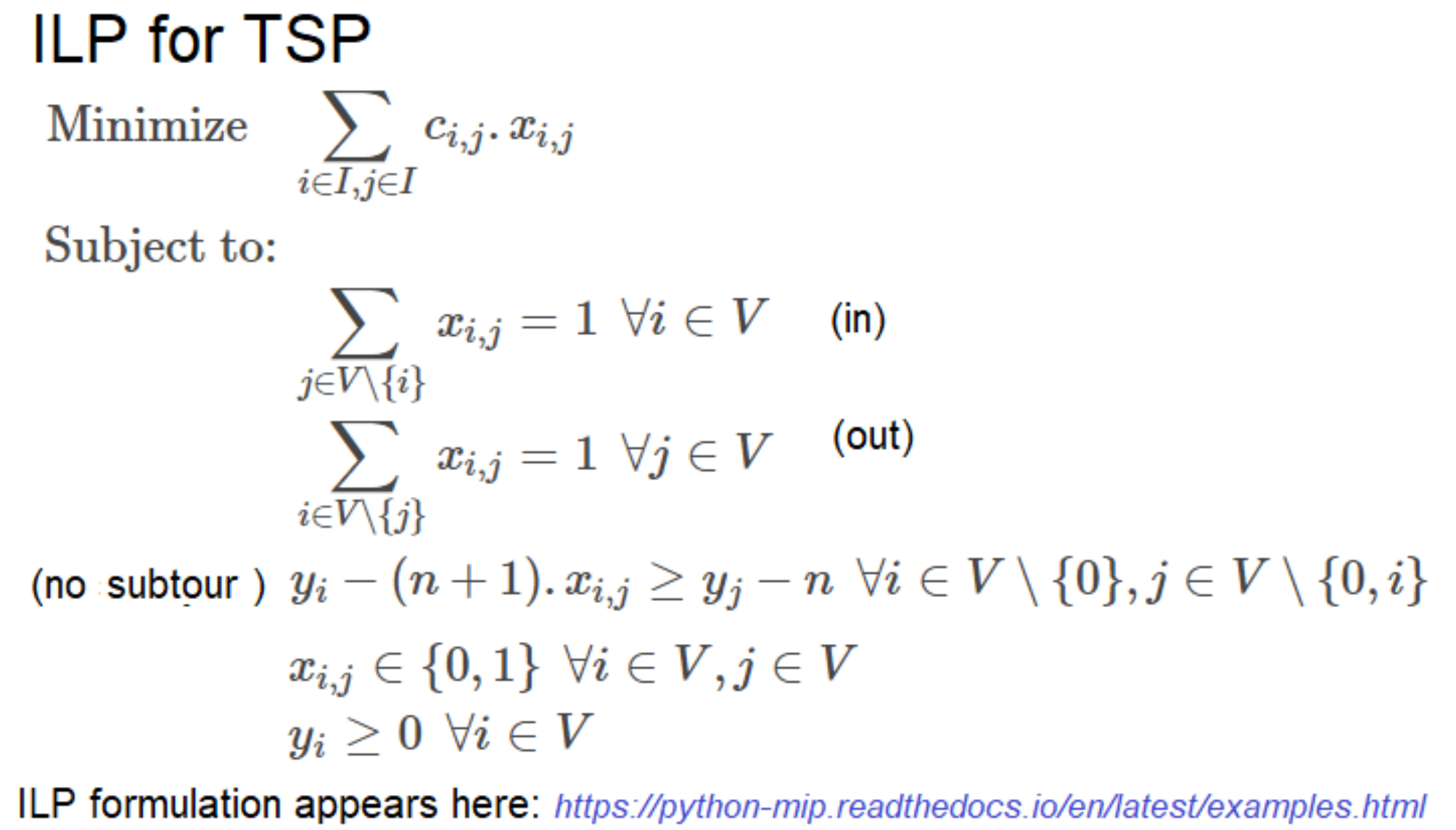
$$\sum_{i=1}^{m} x_{ij} = d_j \qquad \text{for } j = 1, 2, \dots, n,$$

and

$$x_{ij} \ge 0, \qquad \text{for all } i \text{ and } j.$$

Note that the resulting table of constraint coefficients has the special structure shown in Table 8.6. *Any* linear programming problem that fits this special formulation is of the transportation problem type, regardless of its physical context. In fact, there have been numerous applications unrelated to transportation that have been fitted to this special structure, as we shall illustrate in the next example later in this section. (The assignment prob-

TSP



ILP for TSP

$$\text{Minimize} \quad \sum_{i \in I, j \in I} c_{i,j} . x_{i,j}$$

Subject to:

$$\sum_{j \in V \setminus \{i\}} x_{i,j} = 1 \ \forall i \in V \quad \text{(in)}$$

$$\sum_{i \in V \setminus \{j\}} x_{i,j} = 1 \ \forall j \in V \quad \text{(out)}$$

$$\text{(no subtour)} \quad y_i - (n+1). x_{i,j} \geq y_j - n \ \forall i \in V \setminus \{0\}, j \in V \setminus \{0, i\}$$

$$x_{i,j} \in \{0,1\} \ \forall i \in V, j \in V$$

$$y_i \geq 0 \ \forall i \in V$$

ILP formulation appears here: https://python-mip.readthedocs.io/en/latest/examples.html

MCLP

P-median vs p-center

Portfolio Optimization

- Markowitz’s Model
- Budget limitation
- Demand constraint

- Minimize X_i r_i
- S.t.
  - $\sum_i c_i X_i \leq B$
  - $\sum_i c_i X_i \geq D$

  - $X_i \geq 0$, i=1,2,3,…,n

Mean-variance (Markowitz) portfolio selection model

- Maximize expected return of the portfolio
- S.t. all capital is invested
- estimated risk must not exceed a prespecified maximal admissible level of variance $\bar{\sigma}^2$

$$\max_{\boldsymbol{x}} \boldsymbol{\mu}^{\boldsymbol{T}} \boldsymbol{x}$$

s.t.

$$\sum_i x_i = 1$$

$$x^{\mathrm{T}} \Sigma \mathrm{x} \leq \bar{\sigma}^2$$

Where μ = vector of expected return

x_i is the proportion of the capital invested

$\Sigma = variance\ matrix$

Min cost

Simplex

- Steps

I. Standard form

II. Introducing slack variables

III. Creating the tableau

IV. Pivot variables

V. Creating a new tableau

VI. Checking for optimality

VII. Identify optimal values

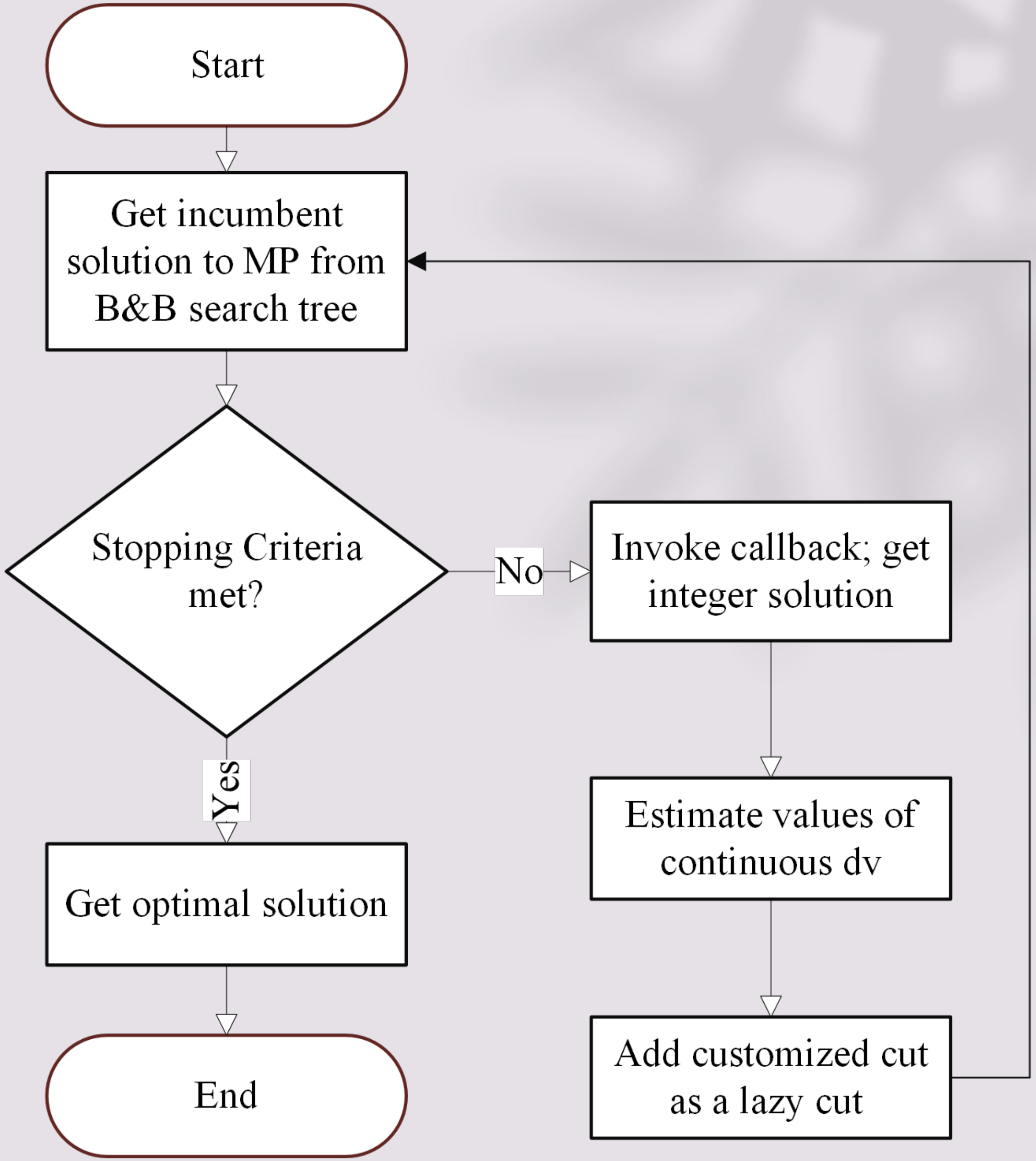
Branch and Bound

- What is it?

- It recursively splits the feasible search space into smaller spaces, then minimizing *f*(*x*) on these smaller spaces; the splitting is called *branching*.
- Branching alone would amount to brute-force enumeration of candidate solutions and testing them all. To improve on the performance of brute-force search, a B&B algorithm keeps track of *bounds* on the minimum that it is trying to find, and uses these bounds to "prune" the search space, eliminating candidate solutions that it can prove will not contain an optimal solution

B&C

- Branch-and-cut methods combine branch-and-bound and cutting-plane methods. The cutting-planes are generated throughout the branch-and-bound tree. The underlying idea is to work on getting as tight as possible bounds in each node of the tree and thus reducing the number of nodes in the search tree.


Start
Get incumbent solution to MP from B&B search tree
Stopping Criteria met?
No
Invoke callback; get integer solution
Estimate values of continuous dv
Add customized cut as a lazy cut
Yes
Get optimal solution
End

Column generation

Add subset of variables iteratively

- The overarching idea is that many linear programs are too large to consider all the variables explicitly. The idea is thus to start by solving the considered program with only a subset of its variables. Then iteratively, variables that have the potential to improve the objective function are added to the program. Once it is possible to demonstrate that adding new variables would no longer improve the value of the objective function, the procedure stops.

The algorithm then proceeds as follow:

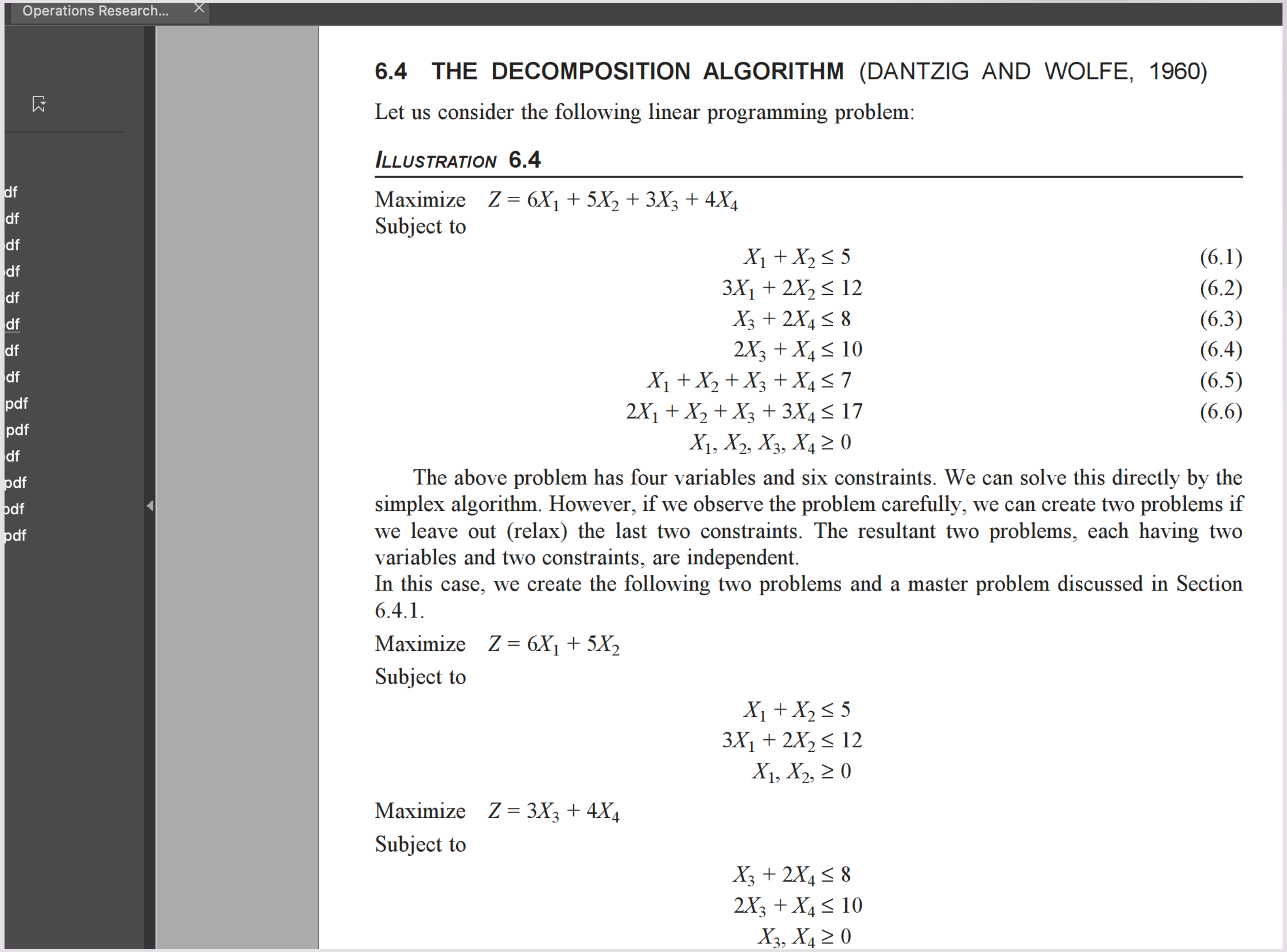
1. Initialise the master problem and the subproblem
2. Solve the master problem
3. Search for an improving variable with the subproblem
4. If an improving variable is found: add it to the master problem then go to step 2
5. Else: The solution of the master problem is optimal. Stop.

When to use B&C vs Column generation?

Column generation

- When large number of decision variables involved
- CG is successfully applied in problems like vehicle routing problems, airplane crew scheduling problems or machine scheduling problems

Decomposition



6.4 THE DECOMPOSITION ALGORITHM (DANTZIG AND WOLFE, 1960)

Let us consider the following linear programming problem:

ILLUSTRATION 6.4

Maximize $Z = 6X_1 + 5X_2 + 3X_3 + 4X_4$

Subject to

$$X_1 + X_2 \leq 5 \tag{6.1}$$
$$3X_1 + 2X_2 \leq 12 \tag{6.2}$$
$$X_3 + 2X_4 \leq 8 \tag{6.3}$$
$$2X_3 + X_4 \leq 10 \tag{6.4}$$
$$X_1 + X_2 + X_3 + X_4 \leq 7 \tag{6.5}$$
$$2X_1 + X_2 + X_3 + 3X_4 \leq 17 \tag{6.6}$$
$$X_1, X_2, X_3, X_4 \geq 0$$

The above problem has four variables and six constraints. We can solve this directly by the simplex algorithm. However, if we observe the problem carefully, we can create two problems if we leave out (relax) the last two constraints. The resultant two problems, each having two variables and two constraints, are independent.

In this case, we create the following two problems and a master problem discussed in Section 6.4.1.

Maximize $Z = 6X_1 + 5X_2$

Subject to

$$X_1 + X_2 \leq 5$$
$$3X_1 + 2X_2 \leq 12$$
$$X_1, X_2, \geq 0$$

Maximize $Z = 3X_3 + 4X_4$

Subject to

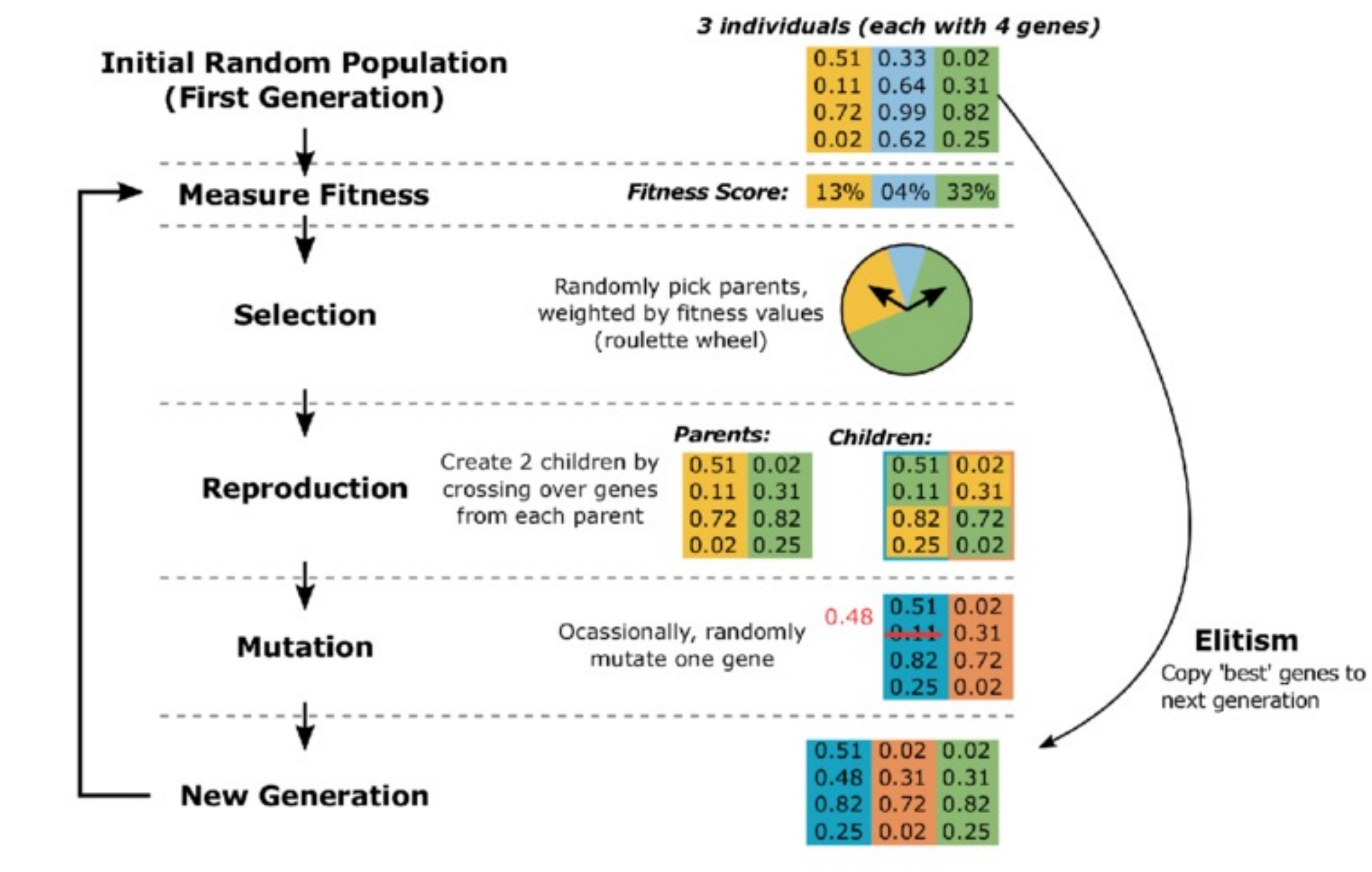
$$X_3 + 2X_4 \leq 8$$
$$2X_3 + X_4 \leq 10$$
$$X_3, X_4 \geq 0$$

Genetic Algorithm

- **Step 1-** Choose an encoding technique, a selection operator, and a crossover operator
- **Step 2-** Choose a population size
- **Step 3-** Randomly choose the initial population
- **Step 4-** Select parental chromosomes
- **Step 5-** Perform Crossover (random crossover points)
- **Step 6-** Evaluation of offsprings
- **Step 7-** Repeat the process

- From Scratch

GA



3 individuals (each with 4 genes)

Initial Random Population (First Generation)

0.51	0.33	0.02
0.11	0.64	0.31
0.72	0.99	0.82
0.02	0.62	0.25

Measure Fitness

Fitness Score: 13% 04% 33%

Selection

Randomly pick parents, weighted by fitness values (roulette wheel)

Reproduction

Create 2 children by crossing over genes from each parent

Parents:

0.51	0.02
0.11	0.31
0.72	0.82
0.02	0.25

Children:

0.51	0.02
0.11	0.31
0.82	0.72
0.25	0.02

Mutation

Ocassionally, randomly mutate one gene

0.48

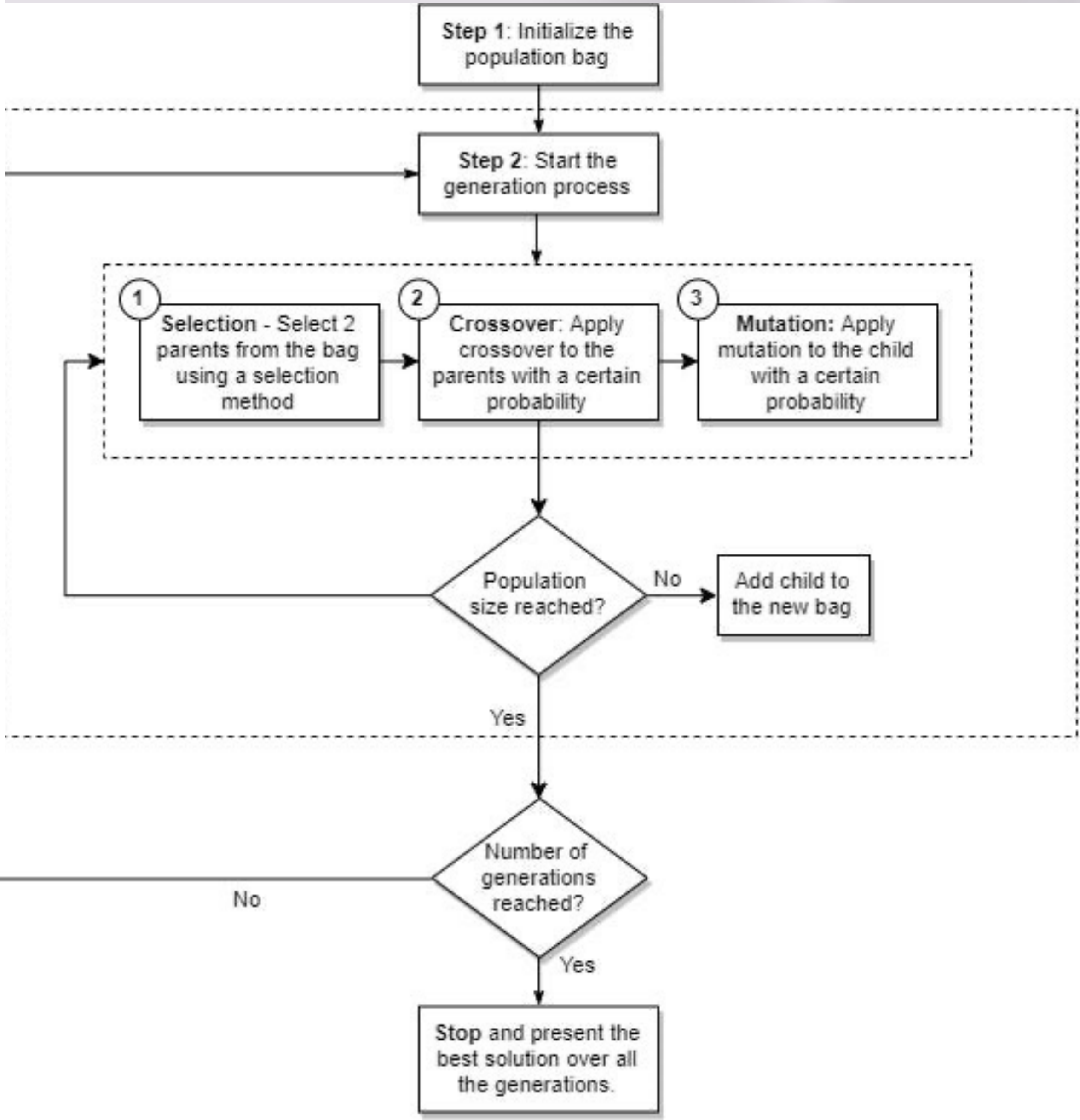
0.51	0.02
~~0.11~~	0.31
0.82	0.72
0.25	0.02

New Generation

0.51	0.02	0.02
0.48	0.31	0.31
0.82	0.72	0.82
0.25	0.02	0.25

Elitism

Copy 'best' genes to next generation



Step 1: Initialize the population bag

Step 2: Start the generation process

1 **Selection** - Select 2 parents from the bag using a selection method

2 **Crossover**: Apply crossover to the parents with a certain probability

3 **Mutation:** Apply mutation to the child with a certain probability

Population size reached?

No

Add child to the new bag

Yes

Number of generations reached?

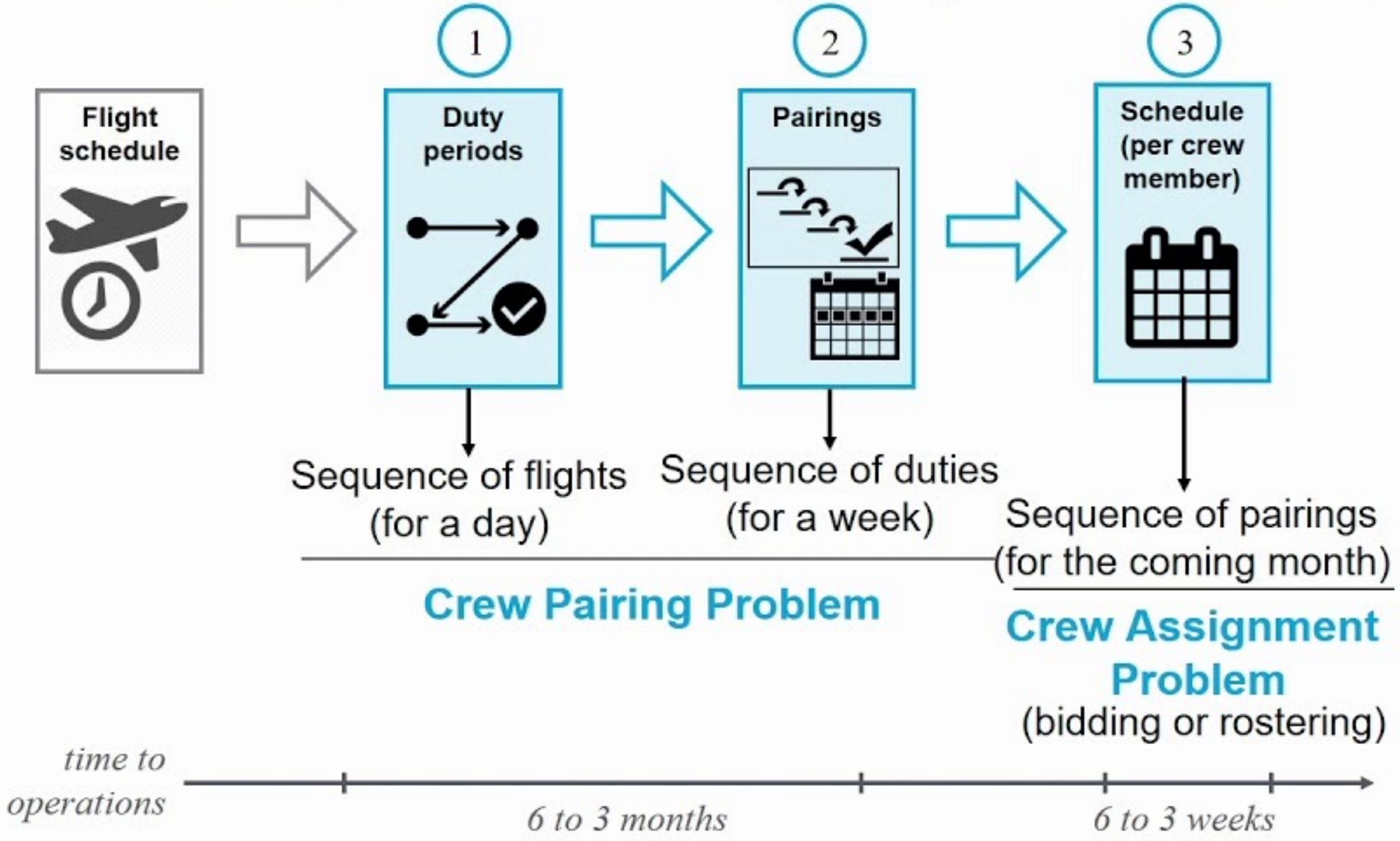
No

Yes

Stop and present the best solution over all the generations.

Problem definition

- The crew scheduling problem is typically solved in three steps:



1 Duty periods → Sequence of flights (for a day)

2 Pairings → Sequence of duties (for a week)

3 Schedule (per crew member) → Sequence of pairings (for the coming month)

Flight schedule

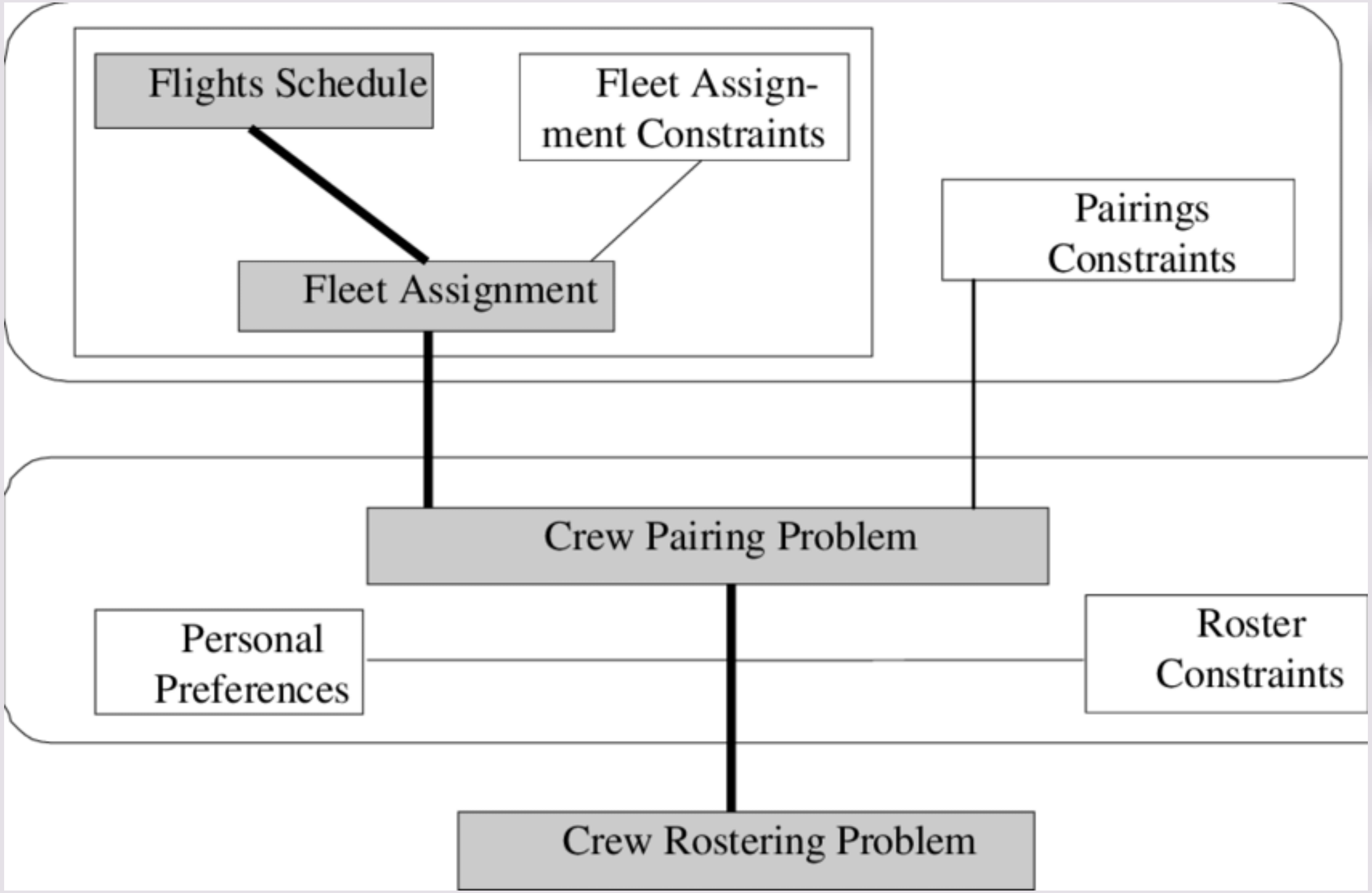
Crew Pairing Problem

Crew Assignment Problem (bidding or rostering)

time to operations

6 to 3 months

6 to 3 weeks


Flights Schedule
Fleet Assignment Constraints
Fleet Assignment
Pairings Constraints
Crew Pairing Problem
Personal Preferences
Roster Constraints
Crew Rostering Problem