
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# Robust Planning of Airport Platform Buses

CS 357: OPTIMIZATION, ALGORITHMS AND TECHNIQUES LAB


| Name | Roll Number |
|---|---|
| Rishika Sharma | 210002063 |
| Agrima Bundela | 210002009 |
| Niranjana R. Nair | 210003049 |

**Course Instructor:**

Dr. Kapil Ahuja


November 29, 2023

# Contents

# 1 Abstract

Efficient ground handling processes at airports are of pivotal importance for seamless air travel experiences. A critical aspect of these processes involves the transportation of passengers between terminals and aircraft stands, especially in scenarios where air bridges are not available. The utilization of platform buses for remote stands introduces the Bus Planning Problem, where the challenge lies in optimizing the allocation of limited buses to various routes and times.

This report addresses the need for a robust scheduling plan for platform buses, recognizing the cascading impact of flight delays on ground handling operations. The domino effect triggered by delays underscores the significance of a well-designed and adaptable bus scheduling framework. Our objective is to develop an Integer Linear Programming (ILP) formulation that maximizes robustness, aiming to minimize re-planning efforts and enhance recovery mechanisms in the face of unforeseen disruptions.

Beyond immediate operational benefits, the proposed ILP formulation extends its impact to broader considerations. By optimizing bus scheduling, the transportation system becomes more environment-friendly and energy-efficient. This aligns with contemporary concerns about sustainable practices in the aviation industry, contributing to the larger discourse on greener and more resilient airport operations.

# 2 Rationale

## 2.1 Operational Efficiency

1) Streamlining bus scheduling minimizes delays and optimizes resource utilization.
2) Addresses the Bus Planning Problem, reducing cascading delays and enhancing operational efficiency.

## 2.2 Environmental Considerations

1) Optimized scheduling aligns with the aviation industry's commitment to environmental sustainability.
2) Reduces fuel consumption and emissions, contributing to environmentally friendly airport operations.

## 2.3 Energy Efficiency

1) Recognizes the role of transportation in the airport's overall energy profile.
2) Enhances energy efficiency through precise bus scheduling.

## 2.4 Adaptability and Resilience

1) Robust scheduling contributes to adaptability and resilience in ground handling processes.
2) Minimizes the need for frequent re-planning, enhancing the airport's capacity to recover swiftly from disruptions.

# 3 Problem Description

## 3.1 Glossary

### 3.1.1 Shift

A shift is characterized by four attributes:
1) Start time of the shift.
2) End time of the shift.
3) Bus type operating during the shift (Peak bus or Cobus).
4) Number of buses utilized in the shift.

### 3.1.2 Trip

For departures, passengers board the bus at the terminal's bus gate, travel to the aircraft, and then disembark to board the aircraft. The arrival process is similar but with reversed source and destination. A sequence of these events is termed a trip.

### 3.1.3 Peak Bus

Standard buses used for city public transport, with a capacity of up to 50 passengers.

### 3.1.4 Cobus

Airport-exclusive buses designed with lower floors, dual-side doors, and a capacity of up to 70 passengers.

## 3.2 Cost Function

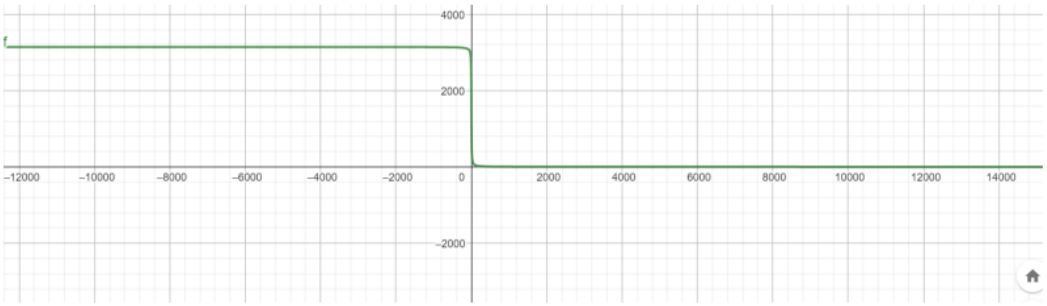
To enhance robustness, optimizing bus idle time is crucial. The idle time between two consecutive trips (t and t') assigned to the same bus, with t being the first trip, is calculated as follows:

Idle time=$T_{t'}^{start}$ -$T_t^{end}$-driving time(t,t')

Here, driving time is the duration taken to travel from the endpoint of the first trip to the starting point of the second trip.

In maximizing schedule robustness, the goal is to maximize idle times between pairs of consecutively assigned trips to the same bus. To model this, a cost function for idle time is introduced. The cost function associates significant costs with small idle times and minor costs with large idle times. The function is designed to be steep for small idle times and gradually flattens out, reflecting the diminishing returns for larger idle times. The specified cost function is:

$C^B(idle)=1000(\arctan(-0.21*idle)+\pi/2)$




# 4 Mathematical Formulation of an Integer Linear Programming Problem (ILP)

To formulate the Integer Linear Programming (ILP) problem, bus plans are introduced, each corresponding to a shift. For a feasible bus plan:
1) All trips within the plan must occur within the shift's specified start and end times.
2) The type of bus assigned to the shift must perform all trips in the bus plan.
3) Trips within the plan should not conflict.
Given a complete set of bus plans, the goal is to select a subset such that each trip is included in exactly one bus plan. Additionally, the selected subset must contain as many bus plans for a shift as there are buses within that shift. The objective is to minimize the total cost of the chosen subset.

## 4.1 Variables

Binary variable $y_j$ for each bus plan j, $y_j$=1 if bus plan j is selected and is 0 otherwise

$c_j$: cost of a bus plan j
M: total number of available bus plans
$S_t$: total number of times trip t needs to be covered
$UAT_t$: allows for trip t to be unassigned in case not enough buses are present to drive all the trips at a certain time
$R_j$: cost coefficient for unassigned trips
$T_b$: number of buses shift b consists of
$h_{tj}$: it is a binary variable which is 1 when trip t is present in bus plan j and is 0 otherwise
$f_{jb}$: it is a binary variable which is 1 if bus plan j corresponds to shift b and is 0 otherwise

## 4.2 Objective Function

Summing the costs of idle times between successive pairs of trips in a bus plan yields the total cost of the plan. The objective is to minimize the total cost of selected bus plans in the subset.

Minimize $\Sigma_{j=1}^{M} c_j y_j + \Sigma_{t=1}^{T} R_t UAT_t$

## 4.3 Constraints

The objective function is subject to the following constraints:

$UAT_t + \Sigma_{j=1}^{M} h_{tj} y_j = S_t$ for t=1,2...T
This constraint ensures the selection of a subset from the complete set in a manner that guarantees

the presence of each trip in exactly one of the bus plans.
$\Sigma_{j=1}^{M} f_{jb} y_j = T_b$ for b=1,2...B
This constraint ensures that the number of bus plans selected for a given shift are equal to the number of buses within that shift.
$y_j \epsilon \{0,1\}$ for j=1,2..n
As mentioned above the binary variable $y_j$ takes values 0 and 1.

## 4.4 Transformation of equations into vector form

**Bus Plan Selection Vector:**

$$\mathbf{y} \text{ of size } M \times 1 :$$
$$\mathbf{y}^T = \begin{bmatrix} y_1 & y_2 & \cdots & y_M \end{bmatrix}$$

**Unassigned Trip Vector:**

$$\mathbf{UAT} \text{ of size } T \times 1 :$$
$$\mathbf{UAT}^T = \begin{bmatrix} UAT_1 & UAT_2 & \cdots & UAT_T \end{bmatrix}$$

**Shift Assignment Vector:**

$$\mathbf{Tb} \text{ of size } B \times 1 :$$
$$\mathbf{Tb}^T = \begin{bmatrix} Tb_1 & Tb_2 & \cdots & Tb_B \end{bmatrix}$$

**Bus Type Selection Vectors:**

$$\mathbf{cobus} \text{ and } \mathbf{peakbus} \text{ of size } M \times 1 :$$
$$\mathbf{cobus}^T = \begin{bmatrix} cobus_1 & cobus_2 & \cdots & cobus_M \end{bmatrix}$$
$$\mathbf{peakbus}^T = \begin{bmatrix} \text{peakbus}_1 & \text{peakbus}_2 & \cdots & \text{peakbus}_M \end{bmatrix}$$

**Driver Break Vector:**

$$\mathbf{driver\_breaks} \text{ of size } T \times 1 :$$
$$\mathbf{driver\_breaks}^T = \begin{bmatrix} \text{driver\_break}_1 & \text{driver\_break}_2 & \cdots & \text{driver\_break}_T \end{bmatrix}$$

**Dual Multiplier Vectors:**

$$\mathbf{dual\_multipliers\_t} \text{ and } \mathbf{dual\_multipliers\_b} \text{ of size } T \times 1 \text{ and } B \times 1 \text{ respectively:}$$
$$\mathbf{dual\_multipliers\_t}^T = \begin{bmatrix} \text{dual\_multipliers\_t}_1 & \text{dual\_multipliers\_t}_2 & \cdots & \text{dual\_multipliers\_t}_T \end{bmatrix}$$
$$\mathbf{dual\_multipliers\_b}^T = \begin{bmatrix} \text{dual\_multipliers\_b}_1 & \text{dual\_multipliers\_b}_2 & \cdots & \text{dual\_multipliers\_b}_B \end{bmatrix}$$

## 4.5 Formulation into a standard LP

**Objective Function:**

Minimize

$$\mathbf{c}^T\mathbf{x}$$

**Decision Variables:**

$$\mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \mathbf{UAT} \\ \mathbf{f} \end{bmatrix} \quad \text{(dimension: } n + T + B\text{)}$$

**Constraints:**

$$\mathbf{Ax} = \mathbf{b} \quad \text{(dimension: } T + B\text{)}$$

Where:

$$\mathbf{A} = \begin{bmatrix} \mathbf{H} & \mathbf{0} \\ \mathbf{I}_T & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_B \end{bmatrix} \quad \text{(dimension: } T + B\text{)} \times (n + T + B\text{)}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{S} \\ \mathbf{T}_1 \\ \vdots \\ \mathbf{T}_B \end{bmatrix} \quad \text{(dimension: } T + B\text{)}$$

Where:

$$\mathbf{S} = \begin{bmatrix} S_1 \\ \vdots \\ S_T \end{bmatrix} \quad \text{(dimension: } T\text{)}$$

$$\mathbf{T}_b = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{(dimension: } B\text{)}$$

**Coefficient Vector:**

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}_{\text{y}} \\ \mathbf{c}_{\text{UAT}} \\ \mathbf{c}_{\text{f}} \end{bmatrix} \quad \text{(dimension: } n + T + B\text{)}$$

# 5 Branch and Bound

## 5.1 Definition

Branch and Bound is an algorithmic paradigm for solving optimization problems, systematically exploring the solution space by dividing it into smaller sub-problems, pruning branches that cannot lead to optimal solutions.

## 5.2 Application to the Bus Scheduling Problem

Branch and Bound can be applied to explore different combinations of bus plans systematically, efficiently traversing the solution space to find the optimal schedule by pruning sub-optimal solutions.

## 5.3 Drawbacks

1) Exponential Growth: The solution space for bus scheduling problems can grow exponentially with the number of buses and trips, making it challenging for Branch and Bound to handle large instances.

2) Complex Constraints: The intricate constraints, such as time dependencies and bus compatibility, might result in a large number of branching decisions, leading to increased computation time.

# 6 Cutting Planes Algorithm

## 6.1 Definition

The Cutting Planes Algorithm is a linear programming-based optimization method that iteratively refines solutions by intersecting the feasible region with cutting planes, converging to an optimal solution.

## 6.2 Application to the Bus Scheduling Problem

The algorithm can be adapted to formulate a linear programming model with decision variables representing bus assignments, trip schedules, and driver breaks. The cutting planes can then be derived from constraints related to capacity, time, and resource usage, progressively improving the schedule's feasibility and optimality.

## 6.3 Drawbacks

1) Real-time Adaptability: The cutting planes algorithm may face difficulties in adapting to real-time changes, such as dynamic passenger demand, traffic variations, and unexpected disruptions. Ensuring the algorithm's responsiveness to these changes is crucial for practical bus scheduling scenarios.

2) Integration of Multiple Objectives: Incorporating multiple optimization objectives, such as minimizing costs, maximizing robustness, and meeting capacity constraints, requires careful formulation and may increase the complexity of the linear programming model.

3) Scalability Issues: As the size of the bus scheduling problem grows, scalability becomes a concern. The computational demands of solving large-scale linear programming problems may limit the algorithm's efficiency in handling extensive schedules and constraints.

# 7 Column Generation Technique

1) Master Problem Solution: Solve the LP-relaxation of the master problem.

The master problem is a linear programming (LP) relaxation of the original optimization problem. The LP-relaxation allows fractional values for decision variables, providing a continuous relaxation of the problem. The objective is to minimize the total cost, considering both the fixed costs associated with bus plans and the costs related to unassigned trips and driver breaks. The constraints ensure that the number of selected bus plans meets the demand for each trip and the overall shift capacity.

2) Pricing Problem: Solve the pricing problem to find a new column with negative reduced cost.

The pricing problem is essentially a subproblem designed to find a new bus plan (column) that can improve the current solution. The objective is to minimize the reduced cost of the new column, where the reduced cost is the difference between the cost of adding the new column to the master problem and the associated dual values (shadow prices) obtained from the solution of the master problem. The pricing problem involves decision variables for the new bus plan, bus type, driver breaks, and unassigned trips.

3) Column Addition: If a new column with negative reduced cost is found, add it to the master problem.

If the solution to the pricing problem yields a new column with a negative reduced cost, it means that adding this column to the current solution would decrease the overall cost. Therefore, the new column is added to the master problem, expanding the set of feasible solutions. This process is crucial for finding an optimal or near-optimal solution efficiently, especially in large-scale optimization problems.

4) Iteration: Repeat steps 1-3 until no more negative reduced cost columns can be found or a maximum number of iterations is reached.

The process of solving the master problem, generating a new column through the pricing problem, and adding it to the master problem is repeated iteratively. The goal is to iteratively improve the solution until no more columns with negative reduced cost can be found or a predefined maximum number of iterations is reached. This iterative approach is a key characteristic of column generation, allowing the algorithm to focus on the most promising columns and avoid considering an exhaustive set of columns.

# 8 Convergence and Stability

## 8.1 Convergence

### 8.1.1 Optimality Condition (Primal)

In the context of bus scheduling, the optimality condition ensures that each new bus plan (column) added to the master problem has a non-negative reduced cost. The reduced cost represents the difference between the cost of adding a new column and the associated dual values obtained from the solution of the master problem.

For each bus plan j, if the cost associated with selecting this plan ($c_j$) minus the contribution of this plan in the current dual solution ($A^T y$) is negative, then this column is added to the master problem.

### 8.1.2 Complementary Slackness (Primal-Dual)

In the bus scheduling context, complementary slackness ensures that the selected bus plans (columns) and their corresponding dual variables are consistent. Specifically, for each primal constraint (corresponding to a trip), either the corresponding dual variable is zero, or the slack in the primal constraint is zero. This ensures that only relevant dual variables are non-zero in an optimal solution.

Illustration: Consider a simplified bus scheduling problem with two trips (T1 and T2) and two bus plans (Bus1 and Bus2). The objective is to minimize the total cost of bus plans while satisfying trip assignments and capacity constraints. We'll focus on one constraint for illustration: the trip assignment constraint for Trip 1.

Primal Constraint (Trip Assignment for T1):
$h_{11}y_{Bus1}+h_{12}y_{Bus2}+UAT_{T1}=S_{T1}$

Dual Variable (Corresponding to Trip Assignment for T1): Let $\lambda_{T1}$ be the dual variable corresponding to the trip assignment constraint for Trip 1.

Complementary slackness ensures that the dual variable is "active" (non-zero) only when the corresponding primal constraint is "binding" (the total demand is fully satisfied). If there is excess capacity (slack), the dual variable becomes zero.

## 8.2 Stability

### 8.2.1 Objective Function Perturbation:

The perturbed objective function $z'(x', c')$ is given by:

$$z'(x', c') = z(x^*, c) + \delta c_j \cdot y_j^* + \text{Reduced Cost}_j \cdot \delta c_j \tag{1}$$

where

1. **Original Objective Function:**
$$z(x, c)$$
2. **Perturbation in Cost Coefficient:**
$$\delta c_j$$
3. **Binary Variable for Column Selection:**
$$y_j^*$$
4. **Reduced Cost of Column $j$:**
$$\text{Reduced Cost}_j = -\frac{\partial z}{\partial c_j}$$

### 8.2.2 Analysis

1. **Original Objective Contribution ($z(x^*, c)$):** Represents the original objective function value with the optimal solution $x^*$ and original cost coefficients $c$.
2. **Perturbation Impact ($\delta c_j \cdot y_j^*$):** Captures the effect of perturbing the cost coefficient $c_j$ on the binary variable $y_j^*$, indicating the selection status of column $j$.

3. **Reduced Cost Contribution (Reduced Cost$_j \cdot \delta c_j$):** Reflects the influence of the reduced cost associated with column $j$ on the perturbed objective function, where reduced cost measures the sensitivity of the objective function to changes in the cost coefficient.

#### 8.2.3 Overall Stability Considerations

The stability of the perturbed objective function is influenced by the interplay between the original objective function value, the perturbation impact, and the contribution from the reduced cost.

Column generation algorithm, by utilizing reduced costs, ensures that changes in the cost coefficients have a controlled impact on the objective function. Reduced costs guide the selection of columns, contributing to a stable solution.

#### 8.2.4 Finite Termination of the algorithm implemented

The algorithm terminates when no more negative reduced cost columns can be found. In the bus scheduling problem, this means that the addition of a new bus plan does not result in the further reduction of the overall cost. The algorithm stops when it converges to a solution where all the added columns have non-negative reduced costs.

#### 8.2.5 Bound on the number of iterations of the algorithm implemented

To ensure the stability of the algorithm, a maximum iteration limit is introduced. This prevents the algorithm from running indefinitely and ensures that it converges within a reasonable number of iterations. In the context of bus scheduling, this maximum iteration limit guards against potential infinite loops and allows the algorithm to provide a solution even if optimality is not reached.

## 9 Input and Output

### 9.1 Input

Input: The input data for the bus scheduling optimization problem is provided through a JSON file named data.json. The structure of the input data includes the following components:
General Information:
M: Total number of bus plans.
T: Total number of trips.
B: Total number of shifts.
cobus_capacity: Capacity of a Cobus. peakbus_capacity: Capacity of a Peak Bus.

Cost Information:
c: Dictionary representing the cost associated with each bus plan.
UAT_values: Dictionary representing the cost of unassigned trips.

Demand and Assignment Information:
S: Dictionary representing the demand for each trip.
Tb: Dictionary representing the demand for each shift.
h: Dictionary representing the assignment of trips to bus plans.
The input data is loaded from the data.json file using the load_data function.

```
"T": 8,
"B": 2,
"c": {"1": 1597.3034339824678, "2": 1680.158222100208, "3": 1672.761902341755, "4": 1203.655077980327
"S": {"1": 2, "2": 1, "3": 3, "4": 2, "5": 1, "6": 2, "7": 1, "8": 2},
"Tb": {"1": 2, "2": 2},
"UAT_values": {"1": 10, "2": 10, "3": 10, "4": 10, "5": 10, "6": 10, "7": 10, "8": 10},
"cobus_capacity": 70,
"peakbus_capacity": 50,
"h": {"1": {"1": 1, "2": 0, "3": 1, "4": 0},
      "2": {"1": 0, "2": 1, "3": 1, "4": 0},
      "3": {"1": 1, "2": 1, "3": 1, "4": 0},
      "4": {"1": 0, "2": 0, "3": 0, "4": 1},
      "5": {"1": 1, "2": 0, "3": 0, "4": 0},
      "6": {"1": 0, "2": 1, "3": 0, "4": 0},
      "7": {"1": 0, "2": 0, "3": 1, "4": 0},
      "8": {"1": 0, "2": 0, "3": 0, "4": 1}}}
```

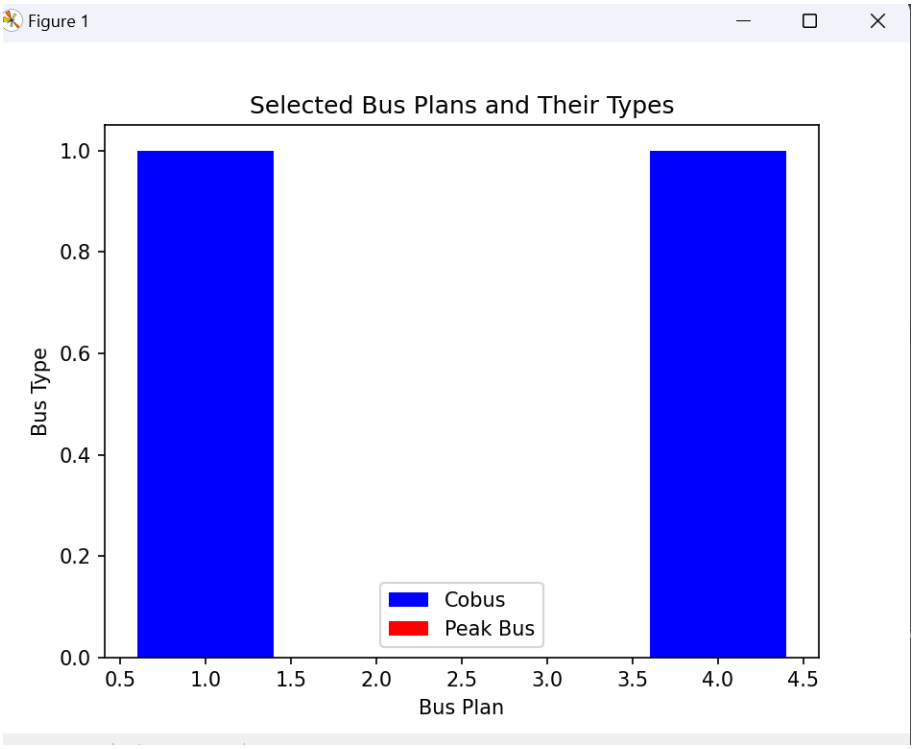

## 9.2 Output

The output of the bus scheduling optimization algorithm includes the following:
Selected Bus Plans:
A list of bus plans selected by the optimization algorithm.

Objective Function Value:
The objective function value corresponding to the total cost of the selected bus plans.

```
Optimal solution found (tolerance 1.00e-06)
Best objective 2.898958511963e+03, best bound 2.898958511963e+03, gap 0.0000%
Selected Bus Plans: [1, 4]
Objective Value: 2898.9585119627955
```

Graphical Representation:
Bar graph showing the selected bus plans and their types (Cobus or Peak Bus).




Bar graph displaying the unassigned trip costs for each trip.


### 9.3 Execution

To execute the code, ensure that the required Python packages (gurobipy, numpy, matplotlib) are installed. The input data is loaded from the data.json file, and the optimization algorithm is executed with the help of Gurobi solver. The results, including the objective function value, selected bus plans and graphical representations are then displayed.

## 10 Future Scope

1. Machine Learning-Based Heuristic Initialization: The current algorithm employs a random heuristic initialization for bus plans. A promising avenue for improvement is the integration of machine learning models to provide intelligent heuristic initialization. By training a model on historical data and patterns, the algorithm could benefit from more informed initial solutions, potentially enhancing convergence speed and solution quality.

2. Variable Aggregation for Complexity Reduction: Introducing variable aggregation techniques can be explored to reduce the problem's complexity. Aggregating variables that share common characteristics or have similar patterns might lead to a more compact representation of the problem. This not only simplifies the model but also has the potential to enhance computational efficiency.

3. Enhanced Dual Variable Initialization: The initialization of dual variables plays a crucial role in the efficiency of column generation algorithms. Investigating advanced techniques for initializing dual variables, such as machine learning-based predictions or adaptive strategies, could contribute to faster convergence and improved overall performance.

## References

[1] https://www.gurobi.com/documentation/current/refman/index.html

[2] https://www.sciencedirect.com/science/article/pii/S0305054811002267

[3] https://optimization.cbe.cornell.edu/index.php?title=Column_generation_algorithms

[4] https://www.cse.iitb.ac.in/~sundar/cs435/lecture15.pdf

[5] https://www.mathworks.com/help/optim/ug/mixed-integer-linear-programming-algorithms.html