What is Column Generation?

Column Generation is an optimization technique commonly used to solve large-scale linear and integer programming problems with a vast number of variables. In traditional approaches, all possible variables (or "columns") are included in the model from the start, which can make solving the problem computationally infeasible for large datasets.

Column generation addresses this by:

1. Starting with a restricted set of variables (columns).
2. Iteratively adding new variables that have potential to improve the solution.
3. Stopping when adding more columns no longer significantly improves the objective.

This technique is particularly useful for problems with many potential options (such as bus-route assignments) because it reduces memory requirements and computation time by only focusing on variables that contribute meaningfully to the solution.

1. Why Column Generation?

Imagine you have an optimization problem with thousands or millions of variables (e.g., each possible bus-route assignment in a transit network). Solving a problem with all these variables at once can be computationally expensive or even infeasible.

Instead of considering all variables from the beginning, column generation starts with a small subset of the variables, then gradually adds new variables only if they improve the solution.

2. How Column Generation Works

1. Restricted Master Problem (RMP): This is a smaller version of the original problem, containing only a subset of the total variables.
2. Pricing Problem (Subproblem): This problem helps identify which additional variables (columns) might improve the solution. The pricing problem evaluates all possible variables and suggests new ones with the highest improvement potential, often based on “reduced costs” from the RMP.

The basic steps are:

1. Solve the RMP with the current columns (variables).
2. Use the Pricing Problem to identify additional columns with potential cost reduction.
3. Add new columns to the RMP based on the pricing problem's suggestions.
4. Repeat until convergence: Stop adding columns when no more significant improvements are possible.

Column Generation in the Bus Routing Optimization Code

In the context of the bus routing problem, column generation is used to find an optimal set of bus-route assignments while minimizing operational costs. Here's how each part of the column generation process is applied in the code.

Step 1: Initialize the Restricted Master Problem (RMP)

In the code, _initialize_columns creates an initial, limited set of bus-route assignments, reducing the size of the problem. This subset is used to create the RMP, which is solved first to get an initial feasible solution.

```
def _initialize_columns(self, initial_fraction=0.1):
    """
    Initialize a subset of bus-route assignments to create a restricted model.
    """
    num_a  (variable) initial_assignments: list[tuple]  en(self.trips) * initial_fraction)
    self.initial_assignments = random.sample(
        [(e.bus_id, t.trip_id) for e in self.electric_buses for t in self.trips],
        num_assignments
    )
```

Step 2: Solve the Restricted Master Problem (RMP)

After initializing, _solve_restricted_problem solves the RMP. This solution provides a baseline, allowing us to evaluate the initial assignments and determine where improvements are needed.

```python
def _solve_restricted_problem(self):
    """
    Solve the restricted problem using the initial subset of assignments.
    """
    # Add only the initial assignments to the model
    for assignment in self.initial_assignments:
        e_id, t_id = assignment
        self.model.x_e[e_id, t_id] = Var(domain=Binary)

    # Define constraints only for the restricted set
    self._apply_constraints(initial_assignments=self.initial_assignments)

    # Solve the restricted problem
    self.solve()
```

Step 3: Identify and Add New Columns (Pricing Problem)

In _add_columns, the code identifies "unserved trips" — trips not assigned to any bus in the current solution. These represent gaps or cost-saving opportunities. The subproblem (pricing problem) adds new bus-route assignments for these trips, expanding the RMP iteratively.

```python
def _add_columns(self):
    """
    Add new columns (bus-route assignments) to the model based on solution feedback.
    """
    # Find unserved trips
    unserved_trips = [t for t in self.trips if not any(self.model.x_e[e.bus_id, t.trip_id].value == 1 for e in self.electric_buses)]

    # Randomly assign an electric bus to each unserved trip to create a new assignment
    for trip in unserved_trips:
        bus = random.choice(self.electric_buses)
        new_assignment = (bus.bus_id, trip.trip_id)

        # Only add if it's not already in the model
        if new_assignment not in self.initial_assignments:
            self.initial_assignments.append(new_assignment)
            self.model.x_e[bus.bus_id, trip.trip_id] = Var(domain=Binary)
```

Step 4: Iterate Until Convergence

The main iteration loop in solve_with_column_generation checks if adding new columns significantly improves the objective. If the improvement is below a tolerance level, the process stops, achieving a converged solution with an optimized set of assignments.

```
=== Iteration 1 ===
Solving restricted problem with current columns.
Restricted problem solved.
Objective value: 1450.5, Improvement: inf
Adding new columns (assignments).
Added 75 new columns for unserved trips.

=== Iteration 2 ===
Solving restricted problem with current columns.
Restricted problem solved.
Objective value: 1300.8, Improvement: 149.7
Adding new columns (assignments).
Added 30 new columns for unserved trips.
```

```
=== Iteration 3 ===
Solving restricted problem with current columns.
Restricted problem solved.
Objective value: 1285.3, Improvement: 15.5
Convergence achieved.

Problem solved.

Displaying solution:
Electric Bus E1 assigned to Trip T3
Electric Bus E5 assigned to Trip T8
Diesel Bus D2 assigned to Trip T15
Diesel Bus D7 assigned to Trip T20
```

Buses

Let's call the buses B1,B2,B3,B4,B5

Routes

Let's call the routes R1,R2,R3,R4,R5

Objective

Our objective is to minimize operational costs. Each bus-route assignment (e.g., B1 assigned to R1) has a cost associated with it. The goal is to assign each route to one bus in a way that minimizes the total cost.

Variables

Each possible bus-route assignment is a variable. We'll define:

- $x_{i,j}=1$ if bus Bi is assigned to route Rj, and $x_{i,j} = 0$ otherwise.

In a traditional model, we would start with all 25 possible variables ($x_{1,1}, x_{1,2}, \ldots, x_{5,5}$), but with column generation, we'll only work with a subset of these at first.

Column Generation Steps

<u>Step 1: Initialize the Restricted Master Problem (RMP)</u>

We start with a restricted subset of bus-route assignments. Let's say we randomly choose only 5 assignments out of the possible 25.

- Initial Assignments:
    - $X_{1,1}$ Bus B1 assigned to Route R1
    - $X_{2,2}$ Bus B2 assigned to Route R2
    - $X_{4,4}$ Bus B4 assigned to Route R4
    - $X_{5,5}$ Bus B5 assigned to Route R5

This means our initial problem (RMP) only considers these 5 assignments. We'll try to optimize costs with this limited set of assignments first.

Step 2: Solve the Restricted Master Problem (RMP)

With only the initial 5 assignments, we try to find the best solution. However, since each route must be served by exactly one bus, we might find that not all routes are served by this restricted model.

Let's assume:

- The RMP solution shows that routes R1,R2,R4, and R5 are covered but R3 is unserved.
- The total cost of the current assignments is, say, $500.

The dual values from this solution tell us how much it "costs" to leave each route unserved, which will guide us in choosing new assignments.

Step 3: Identify and Add New Columns (Pricing Problem)

Since route R3 is unserved, the subproblem (pricing problem) will look for new assignments that could cover R3 or reduce costs.

- Pricing Problem Solution: Finds that assigning B1 to R3 has a low cost.
- New Column: X1,3 (Bus B1 assigned to Route R3).

So, we add X1,3 to our model, expanding the RMP to now include:

- Initial assignments:
    - X1,1 Bus B1 assigned to Route R1
    - X2,2 Bus B2 assigned to Route R2
    - X4,4 Bus B4 assigned to Route R4
    - X5,5 Bus B5 assigned to Route R5
- New assignment: X1,3

Step 4: Re-solve the RMP with the New Column

With this new assignment, we re-solve the RMP:

- This time, the solution assign B1 to R3 to cover the unserved route.
- Let's say the new total cost is $450, showing an improvement from the previous cost of $500.

Step 5: Check for Convergence

- Improvement: The new solution with the added column X1,3 has reduced the cost by $50.
- Repeat or Stop: Since there was a significant improvement, we'll look for additional columns in the next iteration.

Iteration 2: Adding More Columns

1. Identify Unserved Routes or High-Cost Assignments: After the second RMP solution, we find that R2 has a high cost when served by B2.
2. Add New Column for Potential Improvement:
    - Add X3,2 (assigning B3 to R2), which could lower costs by switching routes.
3. Re-solve the RMP with the expanded set of columns.
4. Check Improvement: If adding this new assignment further reduces costs, we repeat the process.

Step 6: Convergence

After a few iterations, we might reach a point where:

- Adding new columns no longer significantly improves the solution (e.g., cost improvement is less than a specified tolerance).
- At this point, column generation converges, meaning we have reached an optimal or near-optimal solution with the current set of bus-route assignments.