It sounds like you're referring to the **Sudoku optimization problem**. Sudoku is typically solved using logic, but it can also be framed as an optimization problem using techniques from operations research and computer science. Here are some approaches:

### 1. **Constraint Satisfaction Problem (CSP) Approach**

Sudoku can be modeled as a **CSP**, where:

* **Variables**: Each cell in the 9x9 grid.
* **Domains**: Numbers 1 through 9 for each cell.
* **Constraints**:
  + Each row contains unique numbers 1-9.
  + Each column contains unique numbers 1-9.
  + Each 3x3 subgrid contains unique numbers 1-9.

**Solution Methods**:

* **Backtracking Algorithm**: Try values one by one, backtrack when a constraint is violated.
* **Forward Checking & Constraint Propagation**: Reduce possible values for cells as decisions are made.

### 2. **Integer Programming (IP) Approach**

Sudoku can also be modeled as an **Integer Programming** problem:

* **Decision Variables**:
* **Objective Function**:
  + Since Sudoku is a feasibility problem (not an optimization), the objective is simply to find a feasible solution that satisfies all constraints.
* **Constraints**:
  1. **Each cell has exactly one number**:
  2. **Each number appears exactly once in each row**:
  3. **Each number appears exactly once in each column**:
  4. **Each number appears exactly once in each 3x3 block**:
  5. **Pre-filled cells constraint**: For given cells, set for the pre-filled numbers.
* **Solving**: You can solve this using optimization solvers like **CPLEX**, **Gurobi**, or **GLPK**.

### 3. **Heuristic & Metaheuristic Approaches**

If you're interested in more flexible or faster methods:

* **Genetic Algorithms**: Use population-based search to evolve solutions over generations.
* **Simulated Annealing**: Start with a random solution and iteratively improve it, allowing occasional worse moves to escape local optima.
* **Tabu Search**: Explore the solution space while avoiding previously visited (tabu) solutions.

Are you looking for a specific method to solve Sudoku, or do you need help implementing one of these approaches?