UNIVERSITY OF BIRMINGHAM

Solving Air Crew Scheduling Problems

A Simulated Annealing and Genetic Algorithm Approach

Evolutionary Computation

Author: Max Hart

mah422@student.bham.ac.uk

Professors: Dr Shan He and Dr Per Kristian Lehre

February 19, 2025

Contents

1 Introduction

Airline crew scheduling is an important optimisation problem, often framed as a Set Partitioning Problem (SPP). In this setup, each flight leg must be assigned to exactly one valid crew rotation, while satisfying various operational constraints. Since the number of possible assignments grows rapidly with problem size, purely exact methods can struggle to scale, especially for real-world scenarios. This has led researchers and practitioners to explore a range of metaheuristic algorithms that offer practical, near-feasible (and often near-optimal) solutions within reasonable time.

In this project, we implement three such algorithms from scratch:

- **Simulated Annealing (SA)**
- **Standard Binary Genetic Algorithm (BGA)**
- **Improved Binary Genetic Algorithm (BGA)**

The improved version draws on ideas from Chu and Beasley [1] and Runarsson and Yao [2], including pseudo–random initialization, heuristic improvement operators, and stochastic ranking for constraint handling. We test our methods on three OR–Library instances (sppnw41, sppnw42, sppnw43) over 30 independent runs, and then compare their performance in terms of feasibility, solution cost, and overall robustness. Our main objective is to investigate how these metaheuristic techniques manage the challenges of the SPP, while providing clear examples of how problem-specific enhancements can boost efficiency and constraint satisfaction.

2 Algorithm Descriptions

In this section, we provide a comprehensive overview of the three algorithms developed for solving the SPP. For each algorithm, we begin with a brief introduction that outlines its underlying principles and technical merits, followed by the corresponding pseudocode. Detailed flowcharts, which visually encapsulate the step-by-step processes, are presented on subsequent pages. This thorough explanation is intended to clarify both the operational mechanics and the innovative aspects of our methods. Ultimately, our goal is to bridge the gap between theoretical concepts and practical implementation.

2.1 Simulated Annealing (SA)

Simulated Annealing is a local search technique that iteratively improves a candidate solution by exploring its neighbourhood. It uses a temperature parameter to probabilistically accept inferior solutions, thereby enabling the algorithm to escape local optima. This simple yet effective approach balances exploration and exploitation through gradual cooling.

2.1.1 Pseudocode

The pseudocode below outlines the Simulated Annealing algorithm for the SPP, detailing random solution initialization, neighbor generation via bit-flipping, acceptance via direct improvement or the Metropolis criterion, and temperature cooling.

Algorithm 1 SimulatedAnnealing(T, α, maxIter, penaltyFactor)

$x \leftarrow$ RandomSolution() ▷ Generate a random binary solution
$F(x) \leftarrow$ PenaltyFitness(x, penaltyFactor)
$x_{\text{best}} \leftarrow x,\ F_{\text{best}} \leftarrow F(x)$
for $iter = 1$ to maxIter **do**
    $x' \leftarrow$ FlipOneRandomBit(x) ▷ Generate a neighbor solution by flipping one bit
    $F(x') \leftarrow$ PenaltyFitness(x', penaltyFactor)
    if $F(x') < F(x)$ **then**
        $x \leftarrow x'$ ▷ Accept the neighbor if it is better
    else
        $\Delta \leftarrow F(x') - F(x)$
        if rand() $< e^{-\Delta/T}$ **then**
            $x \leftarrow x'$ ▷ Accept with probability $e^{-\Delta/T}$ (Metropolis criterion)
    if $F(x) < F_{\text{best}}$ **then**
        $x_{\text{best}} \leftarrow x,\ F_{\text{best}} \leftarrow F(x)$ ▷ Update best solution if improved
    $T \leftarrow \alpha\, T$ ▷ Cool down the temperature
return $(x_{\text{best}}, F_{\text{best}})$

2.1.2 Flowchart

The flowchart visually represents the step-by-step process of the Simulated Annealing algorithm—from initialization and neighbor evaluation to acceptance decisions and temperature cooling.

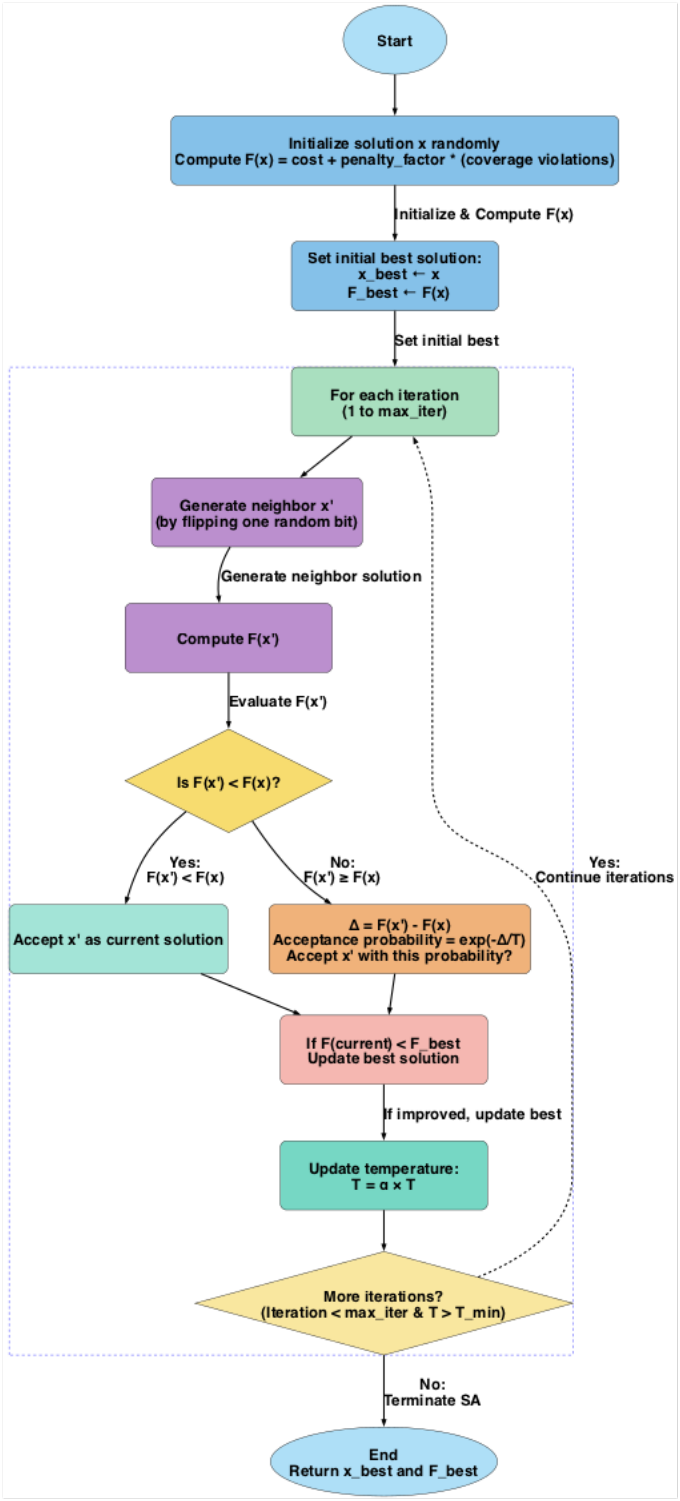

Figure 1: Flowchart of the Simulated Annealing Algorithm for the Set Partitioning Problem.

2.2 Standard Binary Genetic Algorithm (BGA)

The Standard Binary Genetic Algorithm is a population-based technique that evolves a set of candidate solutions using selection, crossover, and mutation operators. It utilises tournament selection to choose parents, applies one-point crossover to combine their genetic material, and uses bit-flip mutation to introduce variability. A penalty function is employed to combine the cost of a solution with constraint violations, ensuring that feasibility is encouraged over generations. This method is well-suited for problems with discrete binary representations and offers robustness through its stochastic operations.

2.2.1 Pseudocode

The pseudocode below outlines the Standard Binary Genetic Algorithm for the SPP, detailing population initialization, tournament selection, crossover, mutation, and population update.

Algorithm 2 StandardBGA(popSize, cxRate, mutRate, maxGens, penaltyFactor, tournK)

$P \leftarrow$ RandomPopulation($popSize$) ▷ Initialize population
EvaluateFitness(P, penaltyFactor) ▷ Evaluate fitness
for $g = 1$ to maxGens **do**
    $Q \leftarrow \emptyset$ ▷ New offspring
    while $|Q| <$ popSize **do**
        $p_1 \leftarrow$ TournamentSelect(P, $tournamentK$) ▷ Select parent 1
        $p_2 \leftarrow$ TournamentSelect(P, $tournamentK$) ▷ Select parent 2
        if rand() $< cxRate$ **then**
            $(c_1, c_2) \leftarrow$ OnePointCrossover(p_1, p_2) ▷ Crossover
        else
            $c_1 \leftarrow p_1,\ c_2 \leftarrow p_2$ ▷ Copy parents
        Mutate(c_1, $mutRate$) ▷ Mutate child 1
        Mutate(c_2, $mutRate$) ▷ Mutate child 2
        EvaluateFitness($\{c_1, c_2\}$, penaltyFactor) ▷ Evaluate offspring
        $Q \leftarrow Q \cup \{c_1, c_2\}$ ▷ Add offspring
    $P \leftarrow Q$ ▷ Update population
return BestSolution(P) ▷ Return best solution

2.2.2 Flowchart

The flowchart provides a visual overview of the Standard BGA's iterative process, from offspring generation to re-ranking and best solution update.

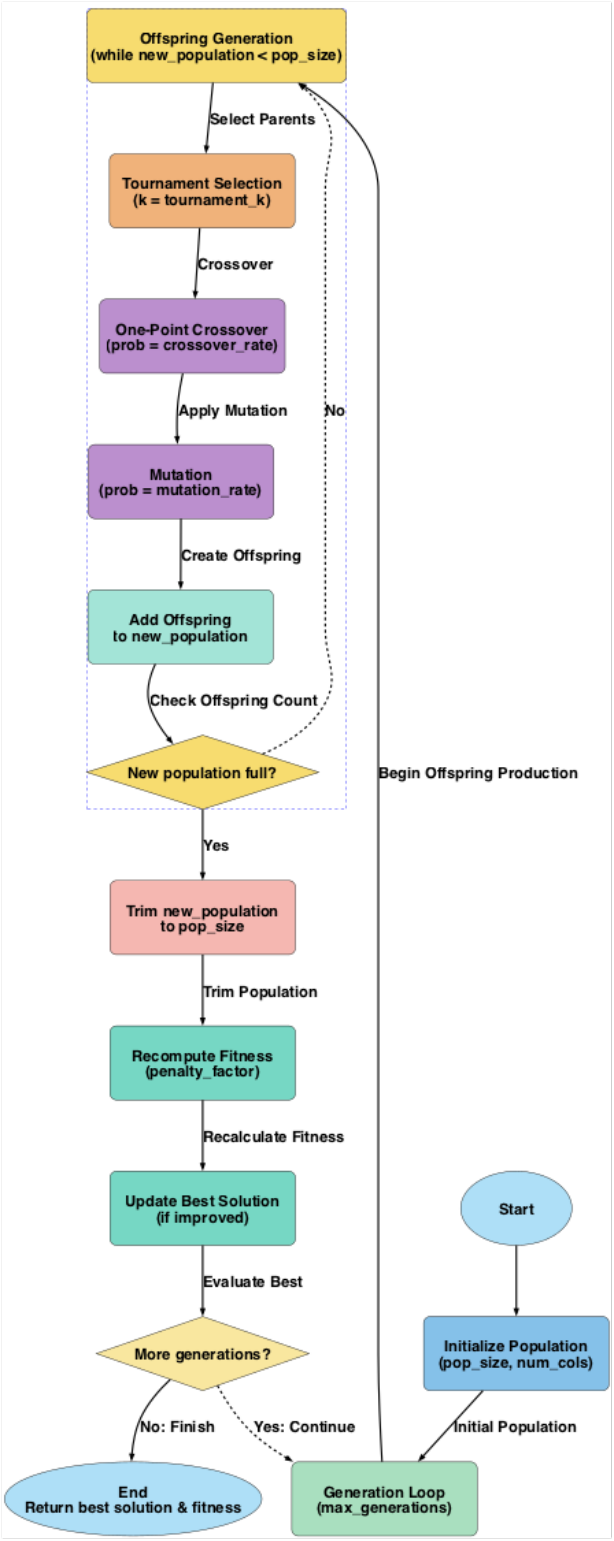

Figure 2: Flowchart of the Standard Binary Genetic Algorithm for the Set Partitioning Problem.

2.3 Improved Binary Genetic Algorithm (BGA)

The Improved Binary Genetic Algorithm extends the standard BGA by integrating several problem-specific enhancements. It begins with a pseudo–random initialisation that aims to reduce over–coverage in the initial population. The algorithm then incorporates a stochastic ranking procedure that probabilistically balances cost and constraint violation, allowing for a more flexible handling of infeasible solutions. Adaptive mutation is employed to reintroduce promising genetic material, and a DROP/ADD heuristic improvement operator repairs infeasible solutions. These enhancements, inspired by Chu & Beasley [1] and Runarsson & Yao [2], make the Improved BGA particularly robust for the SPP. Overall, this algorithm offers a more sophisticated search mechanism by blending global search operators with domain-specific repairs.

2.3.1 Pseudocode

The pseudocode below describes the Improved Binary Genetic Algorithm for the SPP, integrating pseudo–random initialization, stochastic ranking, adaptive mutation, and heuristic repair.

Algorithm 3 ImprovedBGA(popSize, maxGens, p_{stoch}, ...)

$P \leftarrow$ PseudoRandomInit($popSize$) ▷ Initialize population
EvaluateCostUnfitness(P) ▷ Evaluate individuals
for $g = 1$ to maxGens **do**
    StochasticRankSort(P, p_{stoch}) ▷ Rank population
    $O \leftarrow \emptyset$ ▷ Reset offspring
    while $|O| < popSize$ **do**
        $(p_1, p_2) \leftarrow$ SelectParents(P) ▷ Select parents
        $(c_1, c_2) \leftarrow$ UniformCrossover(p_1, p_2) ▷ Crossover
        AdaptiveMutation(c_1, P) ▷ Mutate child 1
        AdaptiveMutation(c_2, P) ▷ Mutate child 2
        HeuristicImprove(c_1) ▷ Repair child 1
        HeuristicImprove(c_2) ▷ Repair child 2
        EvaluateCostUnfitness($\{c_1, c_2\}$) ▷ Evaluate offspring
        $O \leftarrow O \cup \{c_1, c_2\}$ ▷ Add offspring
    $C \leftarrow P \cup O$ ▷ Combine populations
    StochasticRankSort(C, p_{stoch}) ▷ Rank combined pop
    $P \leftarrow$ Top($C, popSize$) ▷ Select top individuals
return BestFeasible(P) ▷ Return best feasible solution

2.3.2 Flowchart

The flowchart visually maps the Enhanced BGA's process, clearly highlighting its advanced offspring production and repair mechanisms.

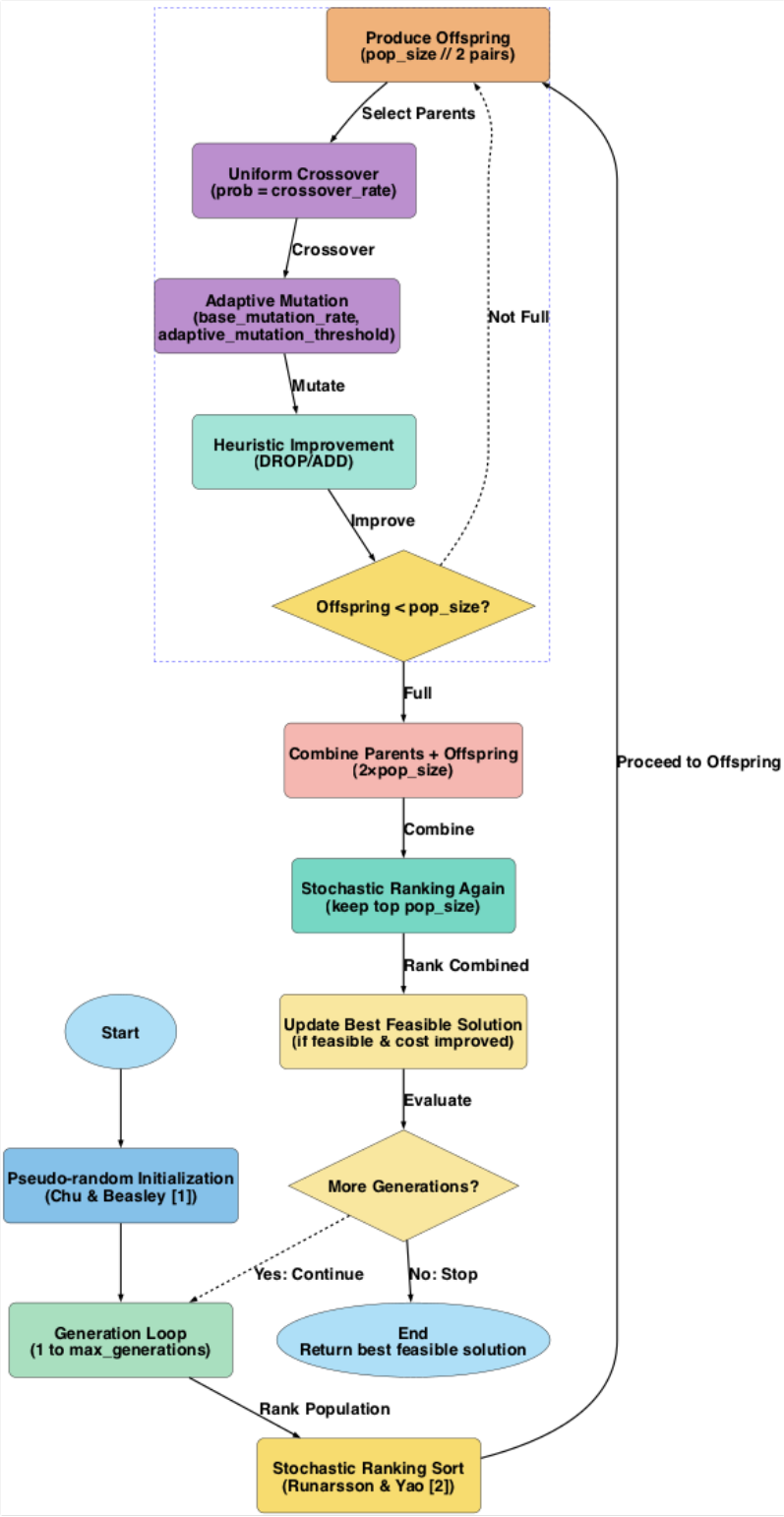

Figure 3: Flowchart of the Improved Binary Genetic Algorithm for the Set Partitioning Problem.

3 Benchmark Results

4 Discussion and Comparison

5 Ranking Replacement vs. Stochastic Ranking

Both methods aim to balance cost and constraint satisfaction in the SPP, yet they differ in approach. The **Ranking Replacement** method (Chu & Beasley, 1998) partitions the population deterministically into four subgroups based on fitness (cost) and unfitness (constraint violation). A new solution replaces an individual from the first non-empty subgroup (starting with those worst in both criteria), thereby steadily improving the overall population. This approach offers a clear structure but can be rigid as the balance between cost and constraint violation may evolve during the search.

In contrast, the **Stochastic Ranking** method (Runarsson & Yao, 2000) uses a bubble-sort-like procedure where adjacent solutions are compared probabilistically—if at least one solution is infeasible, they are compared by cost with a set probability P (typically less than 0.5) and by unfitness otherwise. This allows for a more adaptive balance, enabling the search to explore infeasible regions as bridges between isolated feasible areas without rigid subgroup thresholds.

In summary, while both methods share the goal of guiding the search toward feasible, high-quality solutions, Ranking Replacement enforces a strict hierarchical structure, whereas Stochastic Ranking offers a flexible, adaptive mechanism that reduces the need for extensive parameter tuning.

6 Conclusion

This report has presented three advanced algorithms for solving airline crew scheduling problems: Simulated Annealing, a Standard Binary Genetic Algorithm, and an Improved Binary Genetic Algorithm incorporating problem-specific enhancements. Detailed, moderately sized flowcharts and refined pseudocode have been provided to elucidate each method's internal workings. Benchmark results and an in-depth discussion will be appended once experiments are complete. Furthermore, a comparative discussion on constraint-handling—contrasting ranking replacement and stochastic ranking—has been included, underscoring the adaptive advantages of the latter.

References

[1] P. C. Chu and J. E. Beasley, *Constraint Handling in Genetic Algorithms: The Set Partitioning Problem*, Journal of Heuristics, 11:323–357, 1998.

[2] T. P. Runarsson and X. Yao, *Stochastic Ranking for Constrained Evolutionary Optimisation*, IEEE Transactions on Evolutionary Computation, 4(3):284–294, 2000.