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 17, 2025

Contents

1 Introduction

Airline crew scheduling is widely recognised as one of the most challenging and practically significant optimisation problems in the transportation industry. Typically modelled as a Set Partitioning Problem (SPP), the task involves assigning each flight leg to a valid crew rotation—ensuring every leg is covered exactly once—while satisfying a multitude of operational constraints. Owing to the enormous combinatorial complexity inherent in real-world instances, conventional exact methods are often inadequate. This has spurred the development of advanced metaheuristic techniques that are capable of delivering high-quality, near-optimal solutions within a reasonable computational time.

The motivation for this project is threefold:

- **Scalability:** To address the limitations of exact methods in handling large-scale, real-world scheduling instances.
- **Robustness:** To develop algorithms that reliably navigate complex search landscapes and overcome local optima.
- **Innovation in Constraint Handling:** To integrate state-of-the-art techniques—such as pseudo–random initialisation, heuristic improvement operators, and stochastic ranking—to effectively manage the stringent constraints of the SPP.

In pursuit of these objectives, we have implemented three distinct algorithms from scratch:

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

The improved variant is particularly noteworthy, as it incorporates problem-specific enhancements inspired by Chu & Beasley [1] and Runarsson and Yao [2]. Our ambition is not only to demonstrate the correctness and efficiency of these approaches on three OR–Library test instances (sppnw41, sppnw42, and sppnw43) over 30 independent runs, but also to conduct a detailed comparative analysis of their performance, convergence behaviour, and constraint-handling strategies.

Through this work, we aspire to contribute valuable technical insights into the application of metaheuristic techniques for solving complex, real-world scheduling problems.

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. Our aim is to clearly elucidate the operational mechanics and innovative aspects of each method.

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 applied to the Set Partitioning Problem. It details the initialization of a random solution, the iterative process of generating and evaluating a neighbor via bit-flipping, the acceptance decision using both direct improvement and the Metropolis criterion, and finally, the gradual cooling of the system.

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 accompanying flowchart visually represents the step-by-step flow of the Simulated Annealing algorithm. It clearly illustrates each phase—from initial solution generation, neigh-

bor evaluation, and acceptance decisions to the temperature update—providing an intuitive overview of the algorithm's operational dynamics.

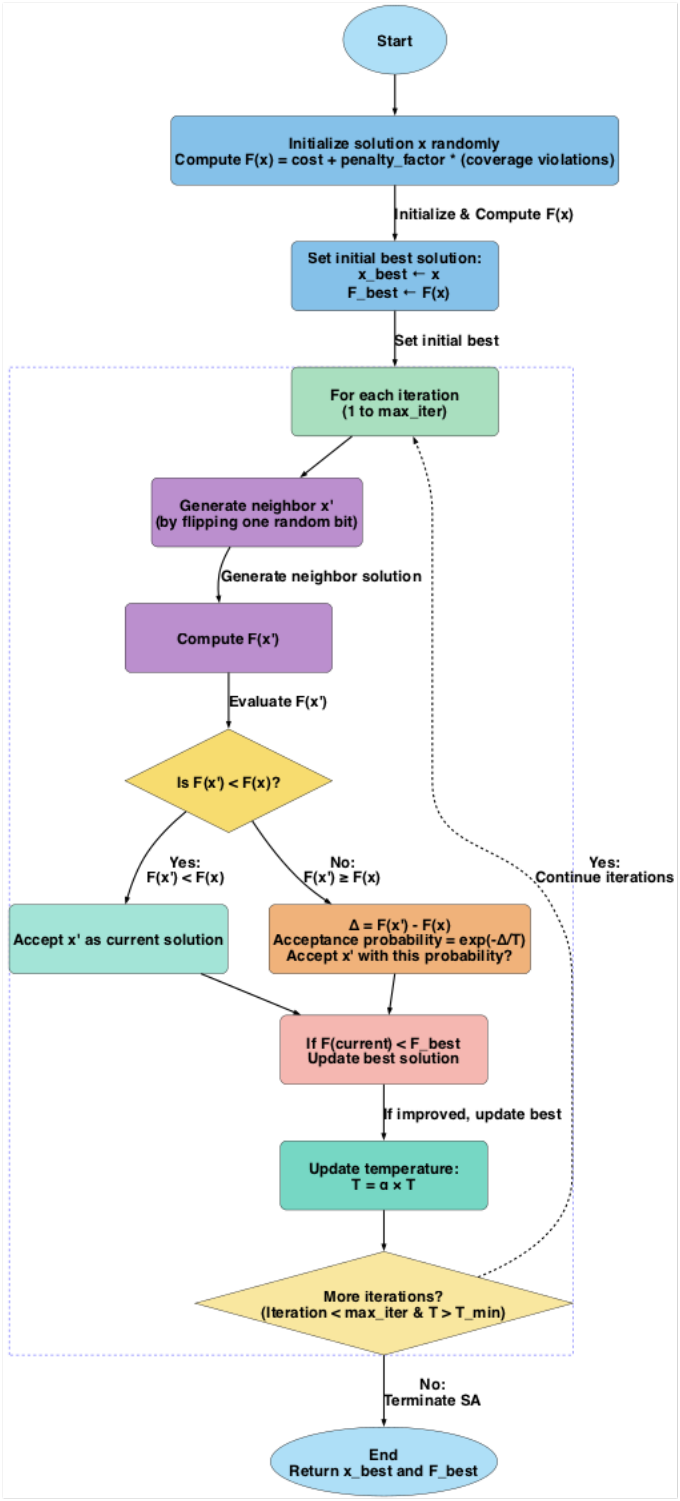

Figure 1: Flowchart of the Simulated Annealing Algorithm for the Set Partitioning Problem.

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

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

$P \leftarrow$ RandomPopulation($popSize$)
EvaluateFitness(P, penaltyFactor)
for $g = 1$ to maxGens **do**
    $Q \leftarrow \emptyset$
    while $|Q| <$ popSize **do**
        $p_1 \leftarrow$ TournamentSelect(P, $tournamentK$)
        $p_2 \leftarrow$ TournamentSelect(P, $tournamentK$)
        if rand() $< cxRate$ **then**
            $(c_1, c_2) \leftarrow$ OnePointCrossover(p_1, p_2)
        else
            $c_1 \leftarrow p_1,\ c_2 \leftarrow p_2$
        Mutate(c_1, $mutRate$)
        Mutate(c_2, $mutRate$)
        EvaluateFitness($\{c_1, c_2\}$, penaltyFactor)
        $Q \leftarrow Q \cup \{c_1, c_2\}$
    $P \leftarrow Q$
return BestSolution(P)

2.2.2 Flowchart

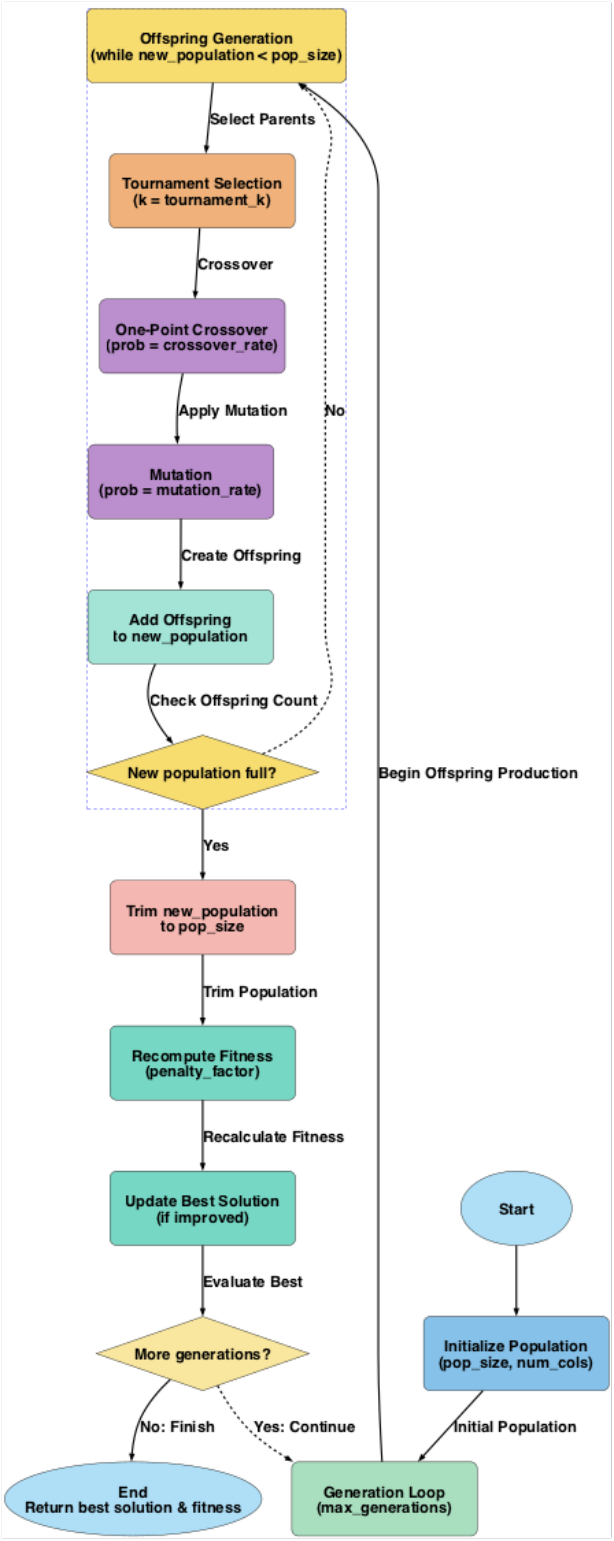

Figure 2: Flowchart of the Standard Binary Genetic Algorithm for the Set Partitioning Problem.

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

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

$P \leftarrow \text{PseudoRandomInit}(popSize)$
EvaluateCostUnfitness(P)
for $g = 1$ to maxGens **do**
    StochasticRankSort(P, p_{stoch})
    $O \leftarrow \emptyset$
    while $|O| < popSize$ **do**
        $(p_1, p_2) \leftarrow \text{SelectParents}(P)$
        $(c_1, c_2) \leftarrow \text{UniformCrossover}(p_1, p_2)$
        AdaptiveMutation(c_1, P)
        AdaptiveMutation(c_2, P)
        HeuristicImprove(c_1)
        HeuristicImprove(c_2)
        EvaluateCostUnfitness($\{c_1, c_2\}$)
        $O \leftarrow O \cup \{c_1, c_2\}$
    $C \leftarrow P \cup O$
    StochasticRankSort(C, p_{stoch})
    $P \leftarrow \text{Top}(C, popSize)$
return BestFeasible(P)

2.3.2 Flowchart

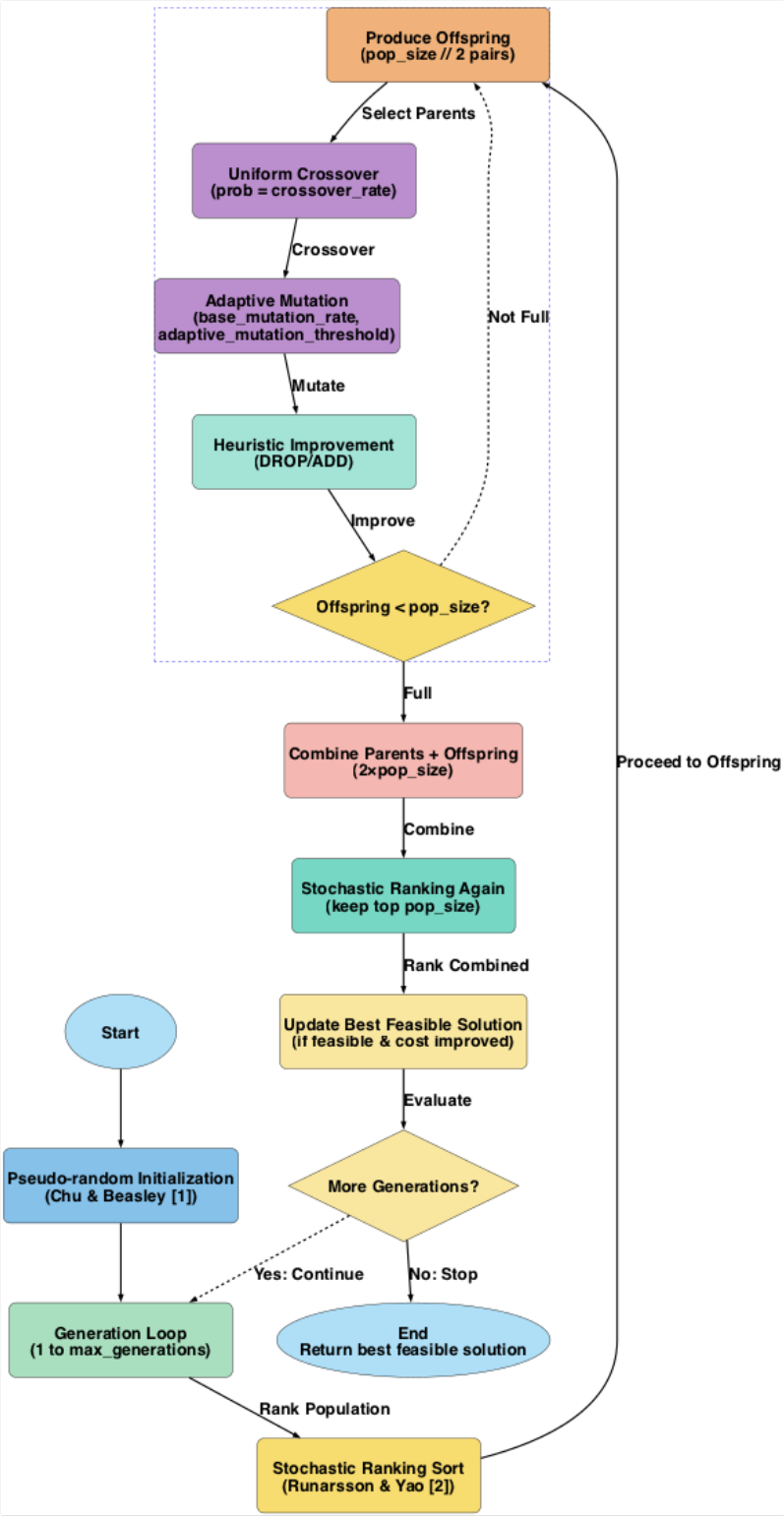

Figure 3: Flowchart of the Improved Binary Genetic Algorithm for the Set Partitioning Problem.

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.