Pseudo Code of Genetic Algorithm and Multi-Start Strategy Based Simulated Annealing Algorithm for Large Scale Next Release Problem

Jifeng Xuan, He Jiang, Zhilei Ren
E-mail: xuan@mail.dlut.edu.cn

School of Software, Dalian University of Technology, China

1. Introduction
In this report, we will introduce the pseudo code of two algorithms for the large scale Next Release Problem (NRP). Both of these two algorithms are used for the experiments. We list the pseudo code to show the details for the readers, who are interested in the implementations.

The NRP is a combinatorial optimization problem in search based requirements engineering. The problem model can be found in [1].

2. Pseudo Code of Genetic Algorithm
The Genetic Algorithm (GA) is a classic algorithm, which is a bio-inspired and population-based technology for complex problems, also used for the NRP [2], [3]. We list a kind of implementation of GA for the large scale NRP.

In Algorithm 1, the GA mainly includes the phases of initialization, selection, crossover, and mutation.

There is a repair operator in Line 16, which can generate a feasible solution from an infeasible solution. Many implementations can be done for such repairing. A simple implementation is to randomly remove a selected customer from the solution until the solution is feasible.
### Algorithm 1. Genetic Algorithm for the NRP

**Input:**
- instance \( \Pi \),
- size \( \alpha \) of population,
- rate \( \beta \) of elitism,
- rate \( \gamma \) of mutation,
- number \( \delta \) of iterations

**Output:** solution \( X \)

```plaintext
// Initialization
1. generate \( \alpha \) feasible solutions randomly;
2. save them in the population \( Pop \);

// Loop until the terminal condition
3. for \( i = 1 \) to \( \delta \) do

// Elitism based selection
4. number of elitism \( ne = \alpha \cdot \beta \);
5. select the best \( ne \) solutions in \( Pop \) and save them in \( Pop_1 \);

// Crossover
6. number of crossover \( nc = (\alpha - ne)/2 \) ;
7. for \( j = 1 \) to \( nc \) do
8. randomly select two solutions \( X_A \) and \( X_B \) from \( Pop \);
9. generate \( X_C \) and \( X_D \) by one-point crossover to \( X_A \) and \( X_B \);
10. save \( X_C \) and \( X_D \) to \( Pop_2 \);
7. endfor

// Mutation
12. for \( j = 1 \) to \( nc \) do
13. select a solution \( X_j \) from \( Pop_2 \);
14. mutate each bit of \( X_j \) under the rate \( \gamma \) and generate a new solution \( X_j' \);
15. if \( X_j' \) is unfeasible
16. update \( X_j \) with a feasible solution by repairing \( X_j' \);
17. endif
18. update \( X_j \) with \( X_j' \) in \( Pop_2 \);
19. endfor

// Updating
20. update \( Pop = Pop_1 + Pop_2 \);
21. endfor

// Returning the best solution
22. return the best solution \( X \) in \( Pop \);
```

### 3. Pseudo Code of Multi-Start Strategy Based Simulated Annealing Algorithm

The Simulated Annealing Algorithm (SA) is a typical algorithm for the NRP [1], [4]. We add a multi-start framework [5] to SA for the NRP. Our algorithm is called Multi-Start Strategy Based Simulated Annealing Algorithm (MSSA).
Algorithm 2. MSSA for the NRP

**Input:** instance \( \Pi \),
- non-linear simulated annealing operator \( \mathcal{Z} \),
- number \( \delta \) of iterations

**Output:** solution \( X \)

```
// Initialization
1 generate a feasible solution \( X_{\text{max}} \) randomly;
// Loop until the terminal condition
2 for \( i = 1 \) to \( \delta \) do
3     generate a feasible solution \( X_1 \) randomly;
4     generate a new solution by \( X_1' = \mathcal{Z} \) for \( X_1 \);
5     if \( X_1' \) is better than \( X_{\text{max}} \)
6         update \( X_{\text{max}} \) with \( X_1' \);
7     endif
8 endfor
// Returning the best solution
9 return the best solution \( X_{\text{max}} \);```

References


