Greedy Randomized Adaptive Search Procedures

贪婪随机自适应搜索算法

吴婷钰

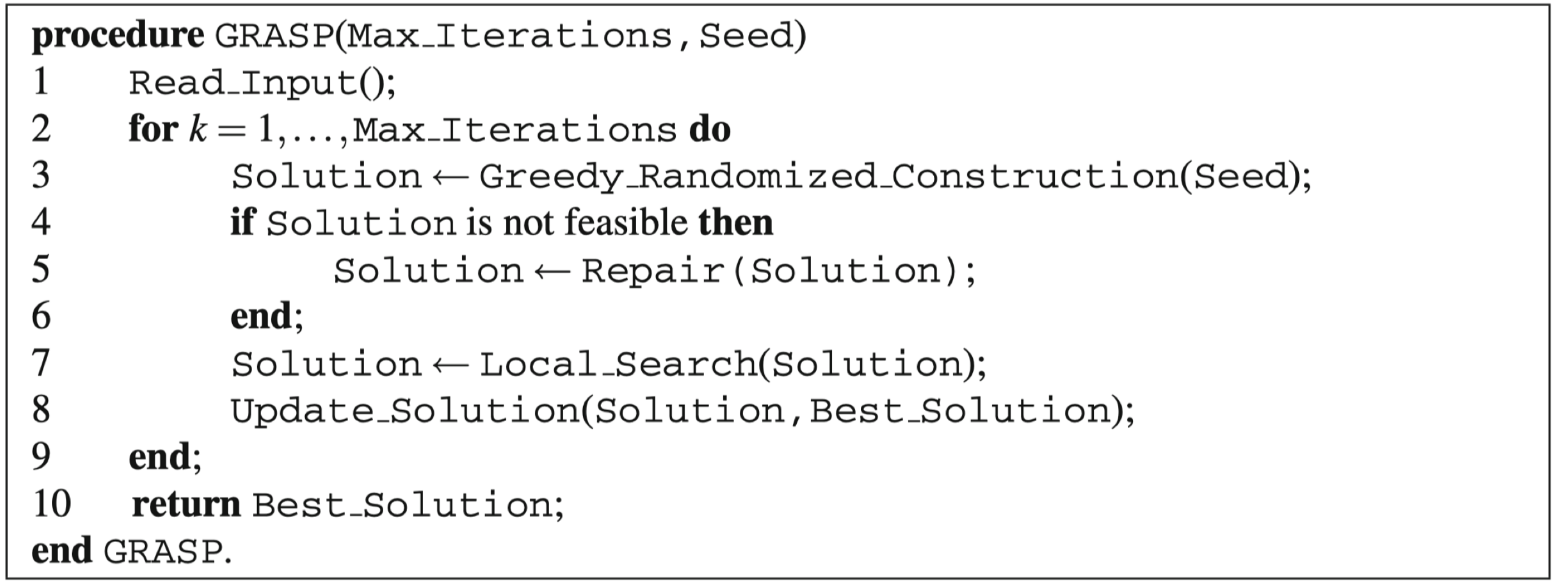
Outline

- ○ Introduction
- ○ GRASP
- ○ Alternative Construction Mechanisms

Introduction

- **GRASP** is a multi-start metaheuristic for combinatorial problems, in which each iteration consists basically of two phases: construction and local search.

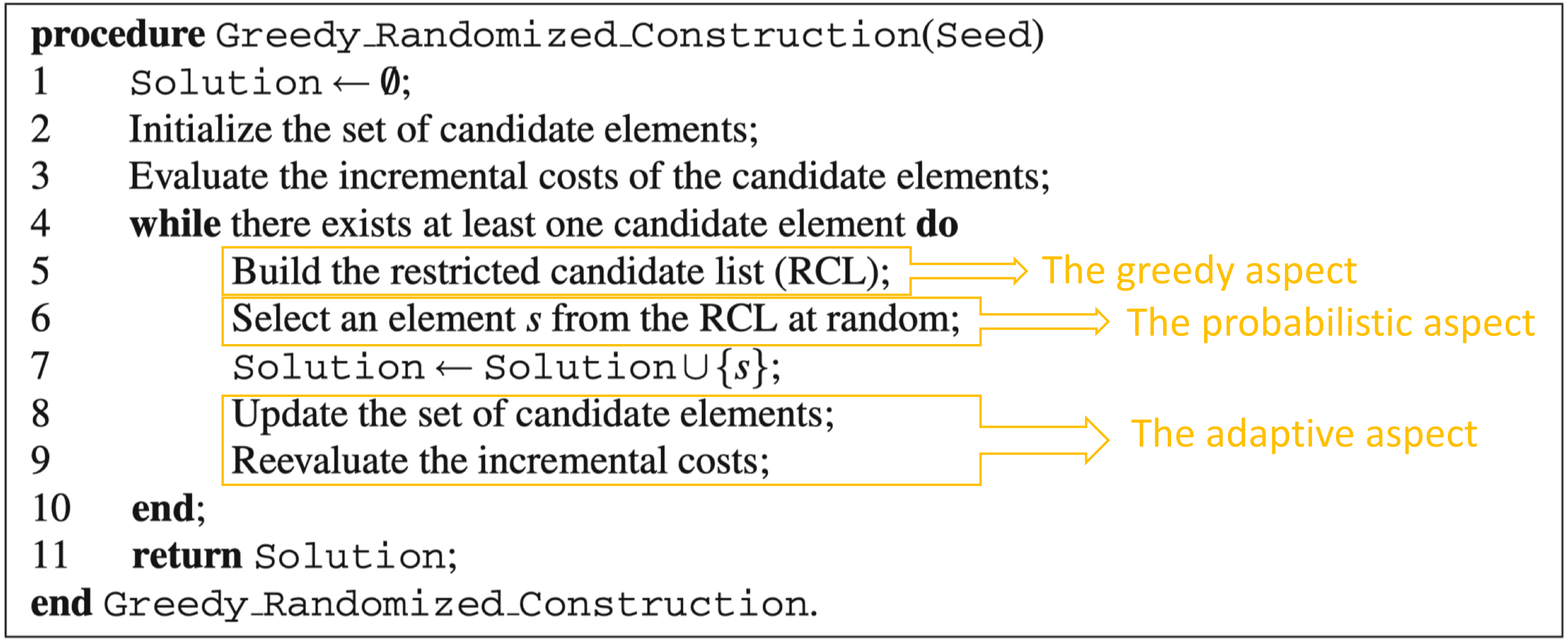
  - **Greedy_Randomized_Construction:** Build a feasible solution. If this solution is not feasible, then it is necessary to apply a repair procedure to achieve feasibility.
  - **Local Search:** Search the neighborhood until a local minimum is found.



```
procedure GRASP(Max_Iterations,Seed)
    Read_Input();
    for $k = 1,\ldots,$Max_Iterations do
        Solution ← Greedy_Randomized_Construction(Seed);
        if Solution is not feasible then
            Solution ← Repair(Solution);
        end;
        Solution ← Local_Search(Solution);
        Update_Solution(Solution,Best_Solution);
    end;
    return Best_Solution;
end GRASP.
```

GRASP

- **Greedy_Randomized_Construction:**
  - **The selection of the next element:** Evaluation of all candidate elements according to a greedy evaluation function.
  - **Restricted candidate list (RCL):** those whose incorporation to the current partial solution results in the smallest incremental costs.



```
procedure Greedy_Randomized_Construction(Seed)
    Solution ← ∅;
    Initialize the set of candidate elements;
    Evaluate the incremental costs of the candidate elements;
    while there exists at least one candidate element do
        Build the restricted candidate list (RCL);
        Select an element $s$ from the RCL at random;
        Solution ← Solution ∪ {$s$};
        Update the set of candidate elements;
        Reevaluate the incremental costs;
    end;
    return Solution;
end Greedy_Randomized_Construction.
```

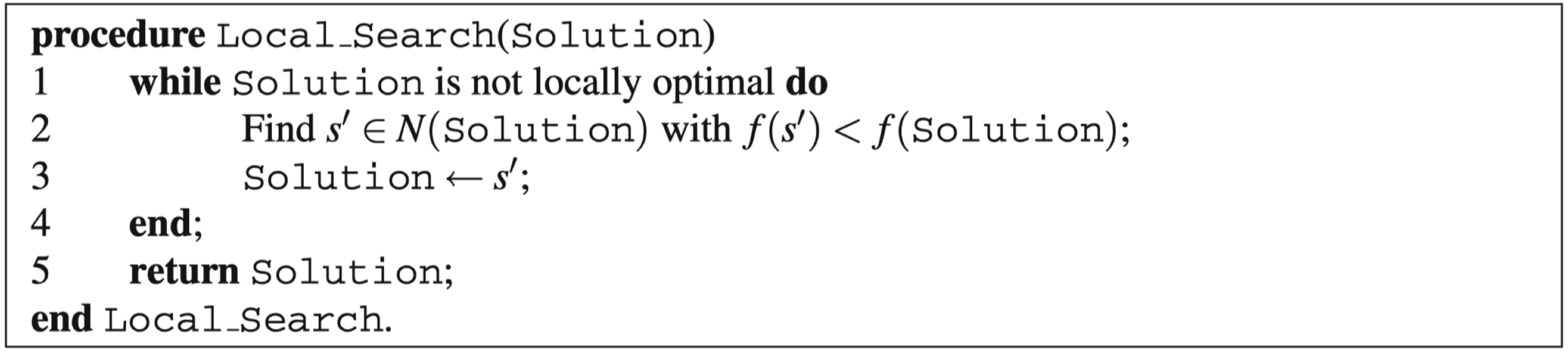
The greedy aspect

The probabilistic aspect

The adaptive aspect

GRASP

- **Local Search:**
  - Starting from the solution Solution constructed and using a neighborhood N.
  - The effectiveness of a LS procedure depends on:
    - the neighborhood structure
    - the neighborhood search technique
    - the fast evaluation of the cost function of the neighbors
    - **the starting solution:** The construction phase plays a very important role on building high-quality starting solutions for the local search



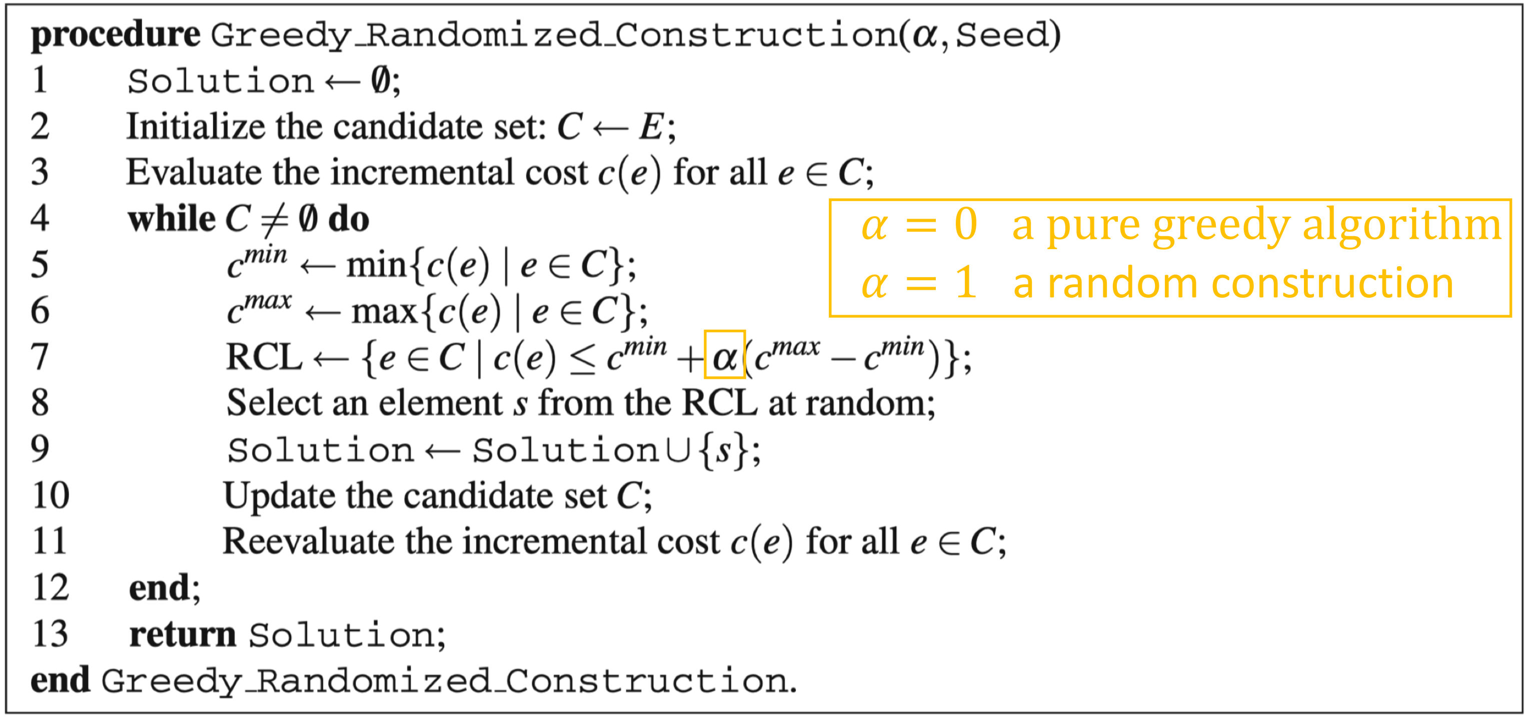
```
procedure Local_Search(Solution)
    while Solution is not locally optimal do
        Find $s' \in N(\text{Solution})$ with $f(s') < f(\text{Solution})$;
        Solution $\leftarrow s'$;
    end;
    return Solution;
end Local_Search.
```

GRASP

- **Construction of the RCL:**
  - An especially appealing characteristic of GRASP: Easy to implement.
  - Few parameters need to be set and tuned. Therefore, development can focus on implementing appropriate data structures for efficient construction and local search algorithms.
  - GRASP has two main parameters: one related to the stopping criterion and another to the quality of the elements in the restricted candidate list.
    - **Stopping criterion:** Max_Iterations of iterations.
    - **Construction of the RCL**

GRASP

- **Construction of the RCL:**
  - $c(e)$: **The incremental cost** associated with the incorporation of element $e \in E$ into the solution under construction.
  - At any GRASP iteration, c^{min} and c^{max} , the smallest and the largest incremental costs.



```
procedure Greedy_Randomized_Construction($\alpha$,Seed)
    Solution $\leftarrow \emptyset$;
    Initialize the candidate set: $C \leftarrow E$;
    Evaluate the incremental cost $c(e)$ for all $e \in C$;
    while $C \neq \emptyset$ do
        $c^{min} \leftarrow \min\{c(e) \mid e \in C\}$;
        $c^{max} \leftarrow \max\{c(e) \mid e \in C\}$;
        RCL $\leftarrow \{e \in C \mid c(e) \leq c^{min} + \boxed{\alpha}(c^{max} - c^{min})\}$;
        Select an element $s$ from the RCL at random;
        Solution $\leftarrow$ Solution $\cup \{s\}$;
        Update the candidate set $C$;
        Reevaluate the incremental cost $c(e)$ for all $e \in C$;
    end;
    return Solution;
end Greedy_Randomized_Construction.
```

$\alpha = 0$ a pure greedy algorithm

$\alpha = 1$ a random construction

Alternative Construction Mechanisms

- **Disadvantages**
  - **Independence of its iterations:** Can’t learn from the search history or from solutions found in previous iterations.
  - **Complexity of GRASP:** At each step of the construction, each yet unselected candidate element has to be evaluated by the greedy function.

Alternative Construction Mechanisms

- **Random Plus Greedy and Sampled Greedy Construction**
  - The **semi-greedy construction scheme** used to build randomized greedy solutions. Two other randomized greedy approaches were proposed, with smaller worst-case complexities than the semi-greedy algorithm.
  - Random Plus Greedy scheme:
    1. Applies randomness during **the first *p* construction steps** to produce a random partial solution.
    2. Next, completes the solution with pure greedy construction steps.
  - Sampled Greedy Construction:
    1. At each step of the construction process, **builds a RCL by sampling $min\{p, |C|\}$ elements** of the candidate set C.
    2. Each element of the RCL is evaluated by the greedy function. The element with the smallest greedy function value is added to the partial solution.

Alternative Construction Mechanisms

- **Reactive GRASP**
  - Incorporate **a learning mechanism** in the memoryless construction phase.
  - The value of the RCL parameter α is **not fixed**, but **randomly selected** at each iteration from a discrete set of possible values.
    - $\Psi = \{\alpha_1, \ldots, \alpha_m\}$: A set of possible values for α.
    - $p_i = 1/m$, for $i = 1, \ldots, m$: Initial the probabilities associated with the choice of each value.
    - z^*: The incumbent solution.
    - A_i: The average value of all solutions found using $\alpha = \alpha_i$, for $i = 1, \ldots, m$.
    - $p_i = q_i / \sum_{j=1}^{m} q_j$, with $q_i = z^* / A_i$: Reevaluated the selection probabilities.

Alternative Construction Mechanisms

- **Bias Functions**
- **In basic GRASP:** the next element is chosen at random from the candidates in the RCL. (equal probabilities of being chosen).
- **Bias the selection** toward some particular candidates. They are based on the rank r(σ) assigned to each candidate element σ , according to its greedy function value. Several bias functions were proposed, such as
  - random bias: $\texttt{bias}(r) = 1$;
  - linear bias: $\texttt{bias}(r) = 1/r$;
  - log bias: $\texttt{bias}(r) = \log^{-1}(r+1)$;
  - exponential bias: $\texttt{bias}(r) = e^{-r}$; and
  - polynomial bias of order n: $\texttt{bias}(r) = r^{-n}$.
- The probability π(σ) of selecting element σ is $\pi(\sigma) = \dfrac{\texttt{bias}(r(\sigma))}{\sum_{\sigma' \in C} \texttt{bias}(r(\sigma'))}$.

Thank You !