

## Iterated greedy (IG) algorithms + GRASP

Christian Blum

Artificial Intelligence Research Institute (IIIA-CSIC)  
Bellaterra, Spain

March, 2017

## Outline

### Simple constructive metaheuristics

- ▶ Iterated greedy (IG) methods
- ▶ Greedy randomized adaptive search procedures (GRASP)

## Historical note

- ▶ Technique that iteratively employs the partial destruction and subsequent reconstruction of a solution
- ▶ **Introduced** in the context of the set covering problem (SCP) [Jacobs and Brusco, 1995], [Marchiori and Steenbeek, 2000]
- ▶ Currently **most-cited paper** [Ruiz and Stützle, 2007]

### Literature:

1. [Jacobs and Brusco, 1995] A local search heuristic for large set-covering problems. Naval Research Logistics Quarterly 1, 61-68, 1995.
2. [Marchiori and Steenbeek, 2000] An evolutionary algorithm for large set covering problems with applications to airline crew scheduling. In: EvoWorkshops 2000, LNCS 1803, Springer Verlag Berlin, 367-381, 2000.
3. [Ruiz and Stützle, 2007] A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 177(3): 2033-2049 (2007)

## Basic IG

### Ideas:

- ▶ **Exploit** the power of the **constructive heuristic** by starting from many different partial solutions
- ▶ **Improve constructive heuristics** by some simple mechanism

## Basic IG

### Pseudo code

```

s ← ConstructGreedySolution()
while termination conditions not met do
    sp ← DestroyPartially(s)
    s' ← Rebuild(sp)
    ApplyLocalSearch(s') {optional}
    AcceptanceCriterion(s', s)
end while
output: best solution found
    
```

## IG Algorithms

### Examples:

- ▶ An IG algorithm for the permutation flow shop scheduling (PFSS) problem

### On the board!!

## Simple IG Extensions: PBIG

### Population-based Iterated Greedy

```

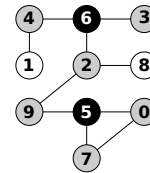
input: pop_size
Generate initial population P with pop_size solutions
while termination conditions not met do
  P' ← P
  for all s ∈ P do
    sP ← DestroyPartially(s)
    s' ← Rebuild(sP)
    ApplyLocalSearch(s') {optional}
    P' ← P' ∪ {s'}
  end for
  P ← Choose the best pop_size solutions from P'
end while
output: best solution from P'
  
```

## Recent Example of PBIG (1)

### Minimum Weight Dominating Set (MWDS)

#### Node weights

$w(0) = 69$   
 $w(1) = 91$   
 $w(2) = 84$   
 $w(3) = 113$   
 $w(4) = 118$   
 $w(5) = 81$   
 $w(6) = 103$   
 $w(7) = 96$   
 $w(8) = 83$   
 $w(9) = 99$



#### Definitions

Black nodes = partial solution  
 Grey nodes = neighbors of black nodes  
 White nodes = rest of the nodes

#### Greedy functions

- 1)  $gfv1(v) := w(v)/\text{current\_degree}(v)$
- 2)  $gfv2(v) := w(v)/\text{weight of white neighbors}$

## Recent Example of PBIG (2)

### Characteristics of the MWDS Application

- ▶ Apply probabilistic solution (re-)construction
- ▶ Choose probabilistically between two greedy functions
- ▶ Only consider the best two options at each step

### Paper

S. Bouamama and C. Blum. A hybrid algorithmic model for the minimum weight dominating set problem. *Simulation Modelling Practice and Theory* 64:57–68 (2016).

## Historical note

- ▶ Randomized constructive technique that uses local search for improving the constructed solutions
- ▶ Introduced by [Feo et al., 1994], [Feo and Resende, 1995]
- ▶ Currently most-cited application paper [Feo et al., 1994]

### Literature:

1. [Feo and Resende, 1995] Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133 (1995)
2. [Feo et al., 1994] A greedy randomized adaptive search procedure for maximum independent set. *Operations Research* 42(5):860-878 (1994)

## Basic GRASP

### Ideas:

- ▶ No use of memory (aim: saving computation time)
- ▶ At each iteration a randomized Greedy heuristic is used for constructing a starting point for local search
- ▶ At each construction step rank the possible extensions and choose some of them to form the restricted candidate list
- ▶ Use local search to improve each constructed solutions

## Basic GRASP

### Pseudo code

```

while termination conditions not met do
  s ← ConstructGreedyRandomizedSolution()
  ApplyLocalSearch(s)
end while
output: best solution found
  
```

## Basic GRASP

### ConstructGreedyRandomizedSolution()

```

 $s^p = \langle \rangle$ 
 $\alpha \leftarrow \text{DetermineRestrictedCandidateListParameter}()$ 
while  $N(s^p) \neq \emptyset$  do
   $RCL \leftarrow \text{GenerateRestrictedCandidateList}(\eta, N(s^p), \alpha)$ 
   $c \leftarrow \text{PickAtRandom}(RCL)$ 
   $s^p \leftarrow \text{extend } s^p \text{ by adding solution component } c$ 
end while

```

## GRASP

### Design guidelines

- ▶ The solution construction mechanism should **sample the most promising regions** of the search space
- ▶ The solutions constructed by the constructive heuristic should belong to basins of attraction of different local minima

## Restriced candidate list

### Size of the candidate list

- ▶ **Cardinality based:**  $\alpha$  is set to a fixed integer
- ▶ **Quality based:**  $\alpha \in [0, 1]$ , Then

$$\bar{\eta} = \max\{\eta(c) \mid c \in N(s^p)\} \quad (1)$$

$$\underline{\eta} = \min\{\eta(c) \mid c \in N(s^p)\} \quad (2)$$

$$RCL = \{c \in N(s^p) \mid \bar{\eta} \geq \eta(c) \geq \bar{\eta} - \alpha(\bar{\eta} - \underline{\eta})\} \quad (3)$$

## Restriced candidate list

### Choice from the candidate list

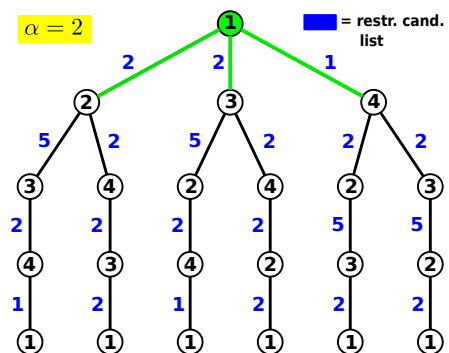
- ▶ Uniformly at random
- ▶ Rank the elements of  $RCL$ , then choose probabilistically according to ...
  1. ... linear bias
  2. ... exponential bias
  3. ... logarithmic bias

## Basic GRASP

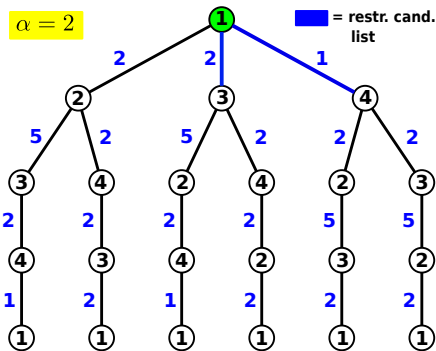
### Example: TSP

- ▶ Construction mechanism: Nearest-neighbor heuristic as explained before.
- ▶ Neighborhood for local search: 2-opt

## GRASP example: TSP



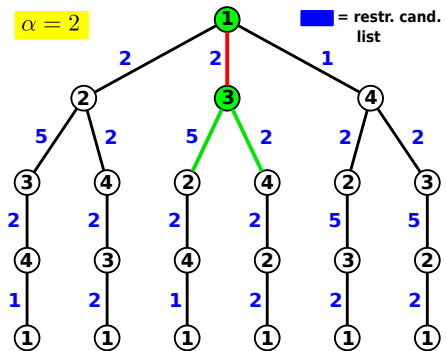
## GRASP example: TSP



© Christian Blum

19

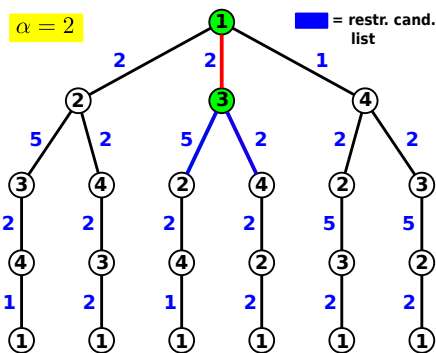
## GRASP example: TSP



© Christian Blum

20

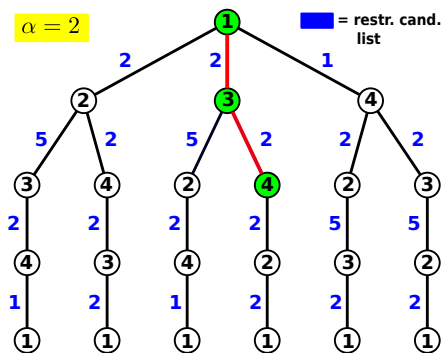
## GRASP example: TSP



© Christian Blum

21

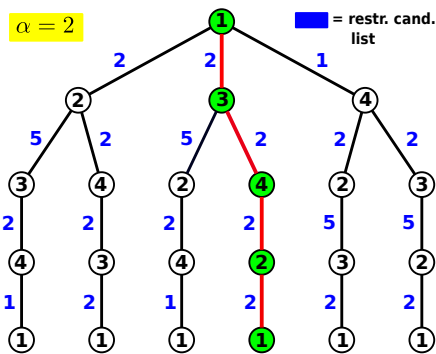
## GRASP example: TSP



© Christian Blum

22

## GRASP example: TSP



© Christian Blum

23

## GRASP examples

### Further examples:

- ▶ A more sophisticated GRASP for the TSP
- ▶ A GRASP for the job shop scheduling (JSS) problem

### Additional features:

- ▶ Added memory
- ▶ Path relinking

On the board!!

© Christian Blum

24

## Questions?