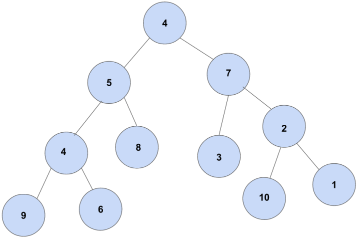
Iterated Greedy (IG) Algorithms + GRASP (Vienna 2022)

Christian Blum

Artificial Intelligence Research Institute (IIIA-CSIC)



Iterated Greedy (IG) algorithm

Main facts

- **In a nutshell:** Technique that iteratively employs the partial destruction and subsequent reconstruction of a solution
- **Introduced** for the *set covering* (SCP) problem [Jacobs and Brusco, 1995][a],[Marchiori and Steenbeek, 2000][b]
- The currently most-cited paper is about the *permutation flow shop scheduling* problem[c]

---

[a] L. W. Jacobs and M. J. Brusco. A local search heuristic for large set-covering problems. Naval Research Logistics Quartely 1, 61-68, 1995.

[b] E. Marchiori and A. Steenbeek. An evolutionary algorithm for large set covering problems with applications to airline crew scheduling. EvoStar 2000.

[c] R. Ruiz & T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 177(3): 2033-2049 (2007)

Basic Iterated Greedy (IG)

Ideas

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

Pseudo code

$s \leftarrow$ ConstructGreedySolution()
while termination conditions not met **do**
  $s^p \leftarrow$ DestroyPartially(s)
  $s' \leftarrow$ Rebuild(s^p)
  ApplyLocalSearch(s'){optional}
  AcceptanceCriterion(s',s)
end while
output: best solution found

Ex.: permutation flow shop scheduling

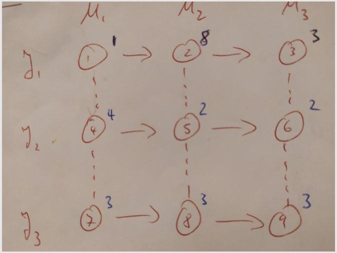
Definition: permutation flow shop scheduling (PFSS)

The PFSS problem is a special case of the GSS problem, with the following characteristics:

- The machine sequence is the same in each job
- All the machines must process the jobs in the same order

Observation: Any permutation of the jobs represents a feasible solution

Example instance



Example: IG for the PFSS problem

Greedy heuristic: Heuristic by Nawatz

1. Compute the sum of the processing times of all n jobs:

$$p_k = \sum_{o_i \in J_k} t(o_i), \quad \forall J_k \in \{J_1, \ldots, J_n\}$$

2. Order the jobs with respect to the p_k-values (descending)
   This provides a first permutation π of all jobs
3. Choose the first two jobs of π: $\pi(1)$ y $\pi(2)$
4. Evaluate the two possible orderings:
   1. $\pi(1)\pi(2)$
   2. $\pi(2)\pi(1)$

   Chose the ordering in which the processing of the operations finishes earlier: π'
5. For $i = 3, \ldots, n$: insert $\pi(i)$ into π' at the best possible place (that least augments the objective function value)

Example: IG for the PFSS problem

Specification of the IG

- **Initial solution π:** make use of the heuristic by Nawatz
- **Partial destruction:** remove d jobs from π (randomly chosen)
- **Reconstruction:** re-insert the removed jobs at the best possible places (in the ordern in which they were rmoved). This generates a solution π'
- **Acceptance criterion:** only accept π' as new incumbent solution if it is better than π

A simple extension of IG: PBIG

Population-based Iterated Greedy (PBIG)

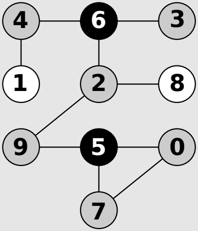
input: *pop_size*
Generate initial population P of *pop_size* solutions
while termination conditions not met **do**
  $P' \leftarrow P$
  for all $s \in P$ **do**
    $s^p \leftarrow \text{DestroyPartially}(s)$
    $s' \leftarrow \text{Rebuild}(s^p)$
    $\text{ApplyLocalSearch}(s')\{\text{optional}\}$
    $P' \leftarrow P' \cup \{s'\}$
  end for
  $P \leftarrow$ select the *pop_size* best solutions from P'
end while
output: the best solution of P'

PBIG: application example (1)

Problem: 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) := (white_degree(v)+eps)/w(v)
2) gfv2(v) := (weight of white neigh+eps)/w(v)

Publication

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).

PBIG: application example (2)

Caracteristics of the PBIG aplication fo the MWDS

- The construction (resp. re-construction) of solutions is done in a probabilistic way
- **For each re-construction of a solution:** select randomly between two greedy functions
- **At each construction step:** select randomly between the best two options

Questions?

Main facts: Greedy Randomized Adaptive Search Procedure (GRASP)

- **In a nutshell:** GRASP is a randomized constructive technique that uses local search for improving the constructed solutions
- **Introduced** by [Feo and Resende, 1995][a] y [Pitsoulis and Resende, 2002][b]

---

[a] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995

[b] L. S. Pitsoulis and M. G. C. Resende. Greedy Randomized Adaptive Search procedure. In P.M. Pardalos and M.G.C. Resende, editors, *Handbook of Applied Optimization*, pages 168–183. Oxford University Press, 2002.

GRASP: ideas + pseudo code

Principles

- 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 the constructed solutions

Pseudo code

while termination conditions not met **do**
  $s \leftarrow$ ConstructGreedyRandomizedSolution()
  ApplyLocalSearch(s)
end while
output: best solution found

GRASP: construction of solutions

ConstructGreedyRandomizedSolution()

$s^p = \langle\rangle$
$\alpha \leftarrow$ DetermineRestrictedCandidateListParameter()
while $N(s^p) \neq \emptyset$ **do**
  $RCL \leftarrow$ GenerateRestrictedCandidateList(η,$N(s^p)$,α)
  $c \leftarrow$ PickAtRandom(RCL)
  $s^p \leftarrow$ extend s^p by appending solution component c
end while

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

GRASP: restrictred candidate list

Size of the candidate list

- A fixed integer: $\alpha > 0$
- Quality based: $\alpha \in [0, 1]$

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

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

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

Choice from the candidate list

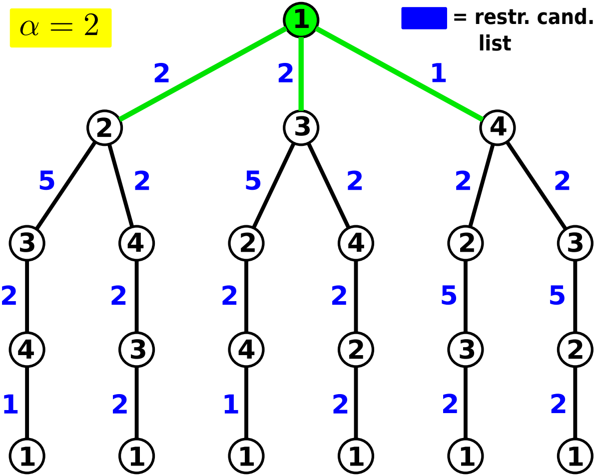
- Uniformly at random
- Rank the elements in *RCL*. Then assign probabilities:
  1. ... linear bias
  2. ... exponential bias
  3. ... logarithmic bias

GRASP example: application to TSP

Example: TSP

- **Solution construction mechanism**: *Nearest-neighbor* heuristic
- **Neighborhood for local search**: 2-opt

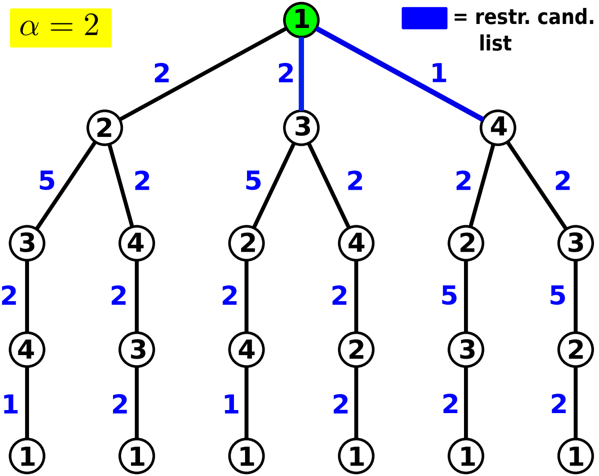
$\alpha = 2$



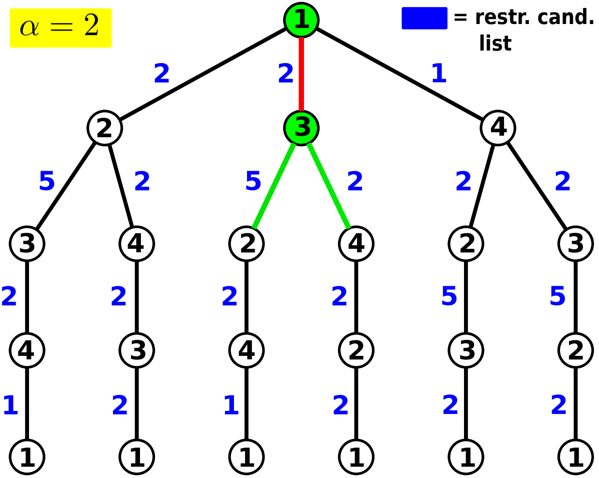
= restr. cand. list
1
2
2
1
2
3
4
5
2
5
2
2
2
3
4
2
4
2
3
2
2
2
2
5
5
4
3
4
2
3
2
1
2
1
2
2
2
1
1
1
1
1
1

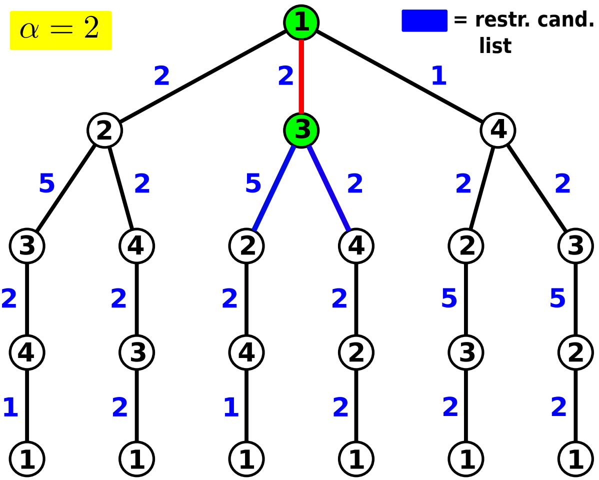
GRASP example: application to TSP

$\alpha = 2$



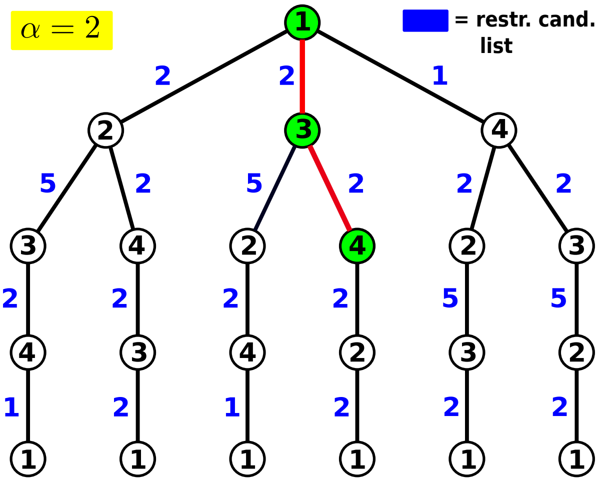
GRASP example: application to TSP



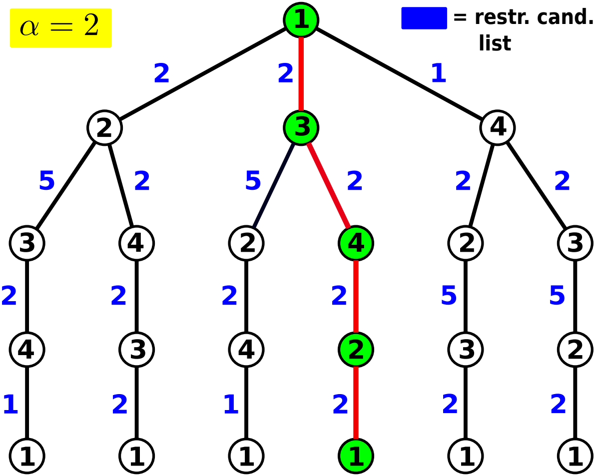


$\alpha = 2$
= restr. cand. list
1
2
2
1
2
3
4
5
2
5
2
2
2
3
4
2
4
2
3
2
2
2
2
5
5
4
3
4
2
3
2
1
2
1
2
2
2
1
1
1
1
1
1

GRASP example: application to TSP



$\alpha = 2$



= restr. cand. list
1
2
2
1
2
3
4
5
2
5
2
2
2
3
4
2
4
2
3
2
2
2
2
5
5
4
3
4
2
3
2
1
2
1
2
2
2
1
1
1
1
1
1

Sophisticated TSP example

Greedy algorithm

1. Each node v_i, $i = 1, \ldots, n$ of the TSP graph G is considered a *sub-tour*.
2. Let $\hat{E} \subseteq E$ be the set of all edges of G such that: $\forall\ e_{i,j} \in \hat{E}$, nodes v_i and v_j form part of different *sub-tours*, e.g., subtours S and S'
3. At each step of the heuristic:
   1. Choose $e_{i,j} \in \hat{E}$ such that $d_{ij} = \min\{d_{k,l} \mid e_{k,l} \in \hat{E}\}$
   2. Merge the two *sub-tours* of v_i and v_j

At the white board!

There are three different cases for merging *sub-tours*

Sophisticated TSP example

Way of working of the GRASP application

1. At each step of a solution construction: order $\hat{E}$ (ascending with respect to the distances)
2. Select the first $\alpha > 0$ edges of the ordered $\hat{E}$ as restricted candidate list (RCL)
3. Pick an option randomly from RCL (uniformly at random)
4. After the construction of a solution apply a 3-opt local search for improving the solution (strategy: first-improvement).

Example: GRASP for the GSS problem

Way of working of the GRASP

- **Solution construction mechanism:** *list scheduler* algorithm
- **Greedy function:** earliest starting time
- **Restricted candidate list (RCL):**
  - Select the best $X\%$ percent of all options (parameter α indicates this percentage)
  - Probabilities are assigned with a linear bias to all options in RCL
- **Local search:** based on the neighborhood of inverting the directions of the first and last arcs of each *group block*, resp. *machine block* on a critical path (strategy: **best-improvement**)
- **Additional algorithmic component:** an archive M able to store maximally q solutions.

Example: GRASP for the GSS problem

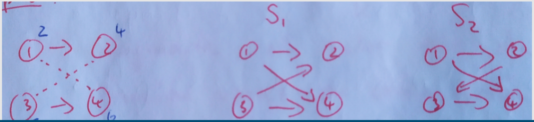
How is the archive M managed?

- **At the start:** M is empty. The first q generated solutions are added to M
- **When M is full:** a new solution s produced by GRASP is added to M iff
  1. s is better than the currently best solution of M. In this case, s substitutes de worst solution of M.
  2. s is **sufficiently different** to the solutions of M

Aim

The solutions in M must be of high quality, but also diverse.

Measures of differences between solutions



How to make use of M? Example

- For each $o_i \in RCL$ (with respect to a partial solution s) compute the following:

$$H_i := \{s' \in M \mid t(s', o_i) = t(s^p, o_i)\}$$

  where $t(s', o_i)$ is the starting time of o_i in solution s'.
- Replace the greedy function $\eta()$ by the following one:

$$\eta_M(o_i) := \frac{\eta(o_i)}{|H_i| + 1}$$

- **Effect:** operations with the same processing time as solutions in M are preferred.

GRASP: an additional feature

On the white board!

- *Path relinking*

Questions?