GRASP

GRASP ≡ Greedy Randomized Adaptive Search Procedure.

Heuristic for finding good local minimums.

One method to find a good local minimum is to ~~generate~~

- generate a random solution
- use local search to move from this random solution ~~unit~~ to a local minimum
- repeat

GRASP modifies this by trying to make the generated random solution reasonably good:

GRASP:

- Use a slightly randomized modified greedy algorithm to generate an initial solution
- Use local search to move from this initial solution to a local minimum
- Repeat

Eg: Set cover:

$$\min \quad x_1 + x_2 + x_3 + x_4$$

$$\text{s.t.} \quad x_1 + x_2 \geq 1$$

$$x_1 + x_3 \geq 1$$

$$x_1 + x_4 \geq 1$$

$$x_2 \geq 1$$

$$x_3 \geq 1$$

$$x_4 \geq 1$$

$$x_i \text{ binary}$$

Greedy approach: Satisfy as many constraints as possible.
So set $x_1 = 1$ initially.
Then need $x_2 = x_3 = x_4 = 1$ also.

GRASP:

$x_1 = 1$ satisfies 3 constraints

$x_2 = 1$ or $x_3 = 1$ or $x_4 = 1$ each satisfy 2 constraints.

If 2 is close enough to 3 in our selection procedure, pick any of the four variables on the first step.
May then be able to generate $x_1 = 0$, $x_2 = x_3 = x_4 = 1$ in initial phase.

So:

Don't necessarily pick the ~~best~~ ~~go~~ most greedy move.
Instead: pick a move that ~~looks~~ is reasonably close to the most greedy move.

Advantages:

(1) Compared with greedy, GRASP visits more of the feasible region.
(2) Compared with random initialization:
  (i) Random initialization generates some very bad solutions. It takes a long time to make one of these solutions look reasonable. So GRASP can look at more solutions than random initialization in the same amount of time.
  (ii) GRASP is like intensification in tabu search: We concentrate on the part of the feasible region which is likely to contain the optimal solution, and many good solutions.

Disadvantages:

May not visit the part of the feasible region containing the optimal solution. Can use diversification to try to overcome this.

Applications include:

Set covering. Node packing. Quadratic Assignment. MAXSAT.
Facility location problems. Production planning & scheduling.

(Reeves: "Genetic Algos for the Operations Research", INFORMS Journal on Computing 9(3), 1997, pp 231–250)

GENETIC ALGORITHMS

Another heuristic for finding good local minima.

Eg: TSP:
Have a collection of tours.
→ Pick two of the tours.
Combine them somehow to give a new tour. ← NOVEL ASPECT OF GENETIC ALGOS.
~~[illegible]~~.
Perhaps mutate this tour
Add this tour to the collection
Repeat. (loop back to "Pick two of the tours")

Eg: MAXCUT.
Combination:
If vertex [illegible] in both parents, keep it there. Else make a 50/50.
Need to break the symmetry.
So, eg. fix [illegible].

How do we perform these steps?

① **Choosing the parents** ("survival of the fittest")

Parents are ~~[illegible]~~ evaluated. Good tours are more likely to be chosen as parents.

(Eg. stochastic universal selection: circle divided into segments, one for each chromosome (ie, parent), have multi-armed spinner that it spins once to get to all parents.)

Use a fitness function, not necessarily the objective function.

② **Combining the parents**

Many possibilities.

Eg: Two routes: 1→2→3 | 4→5→6 | 7→8→1.
1→4→2→3→5→8→7→6→1

Randomly break in two places on Parent 1. Eg: after 3 and before 7.
Use Parent 2 route between these cities. Here get new route
1→2→3→~~5~~8→7→8→1. Fix this by: Remove one of visits to 8.
Add a visit to other cities using [illegible] SHO[illegible]

~~(3) Mutation~~

So perhaps get the tour:
$1 \to 4 \to 2 \to 3 \to 5 \to 6 \to 8 \to 7 \to 1$

(3) Mutation

Again many possibilities.

One choice with low probability, move one city to a different place on the tour.

Can also delete bad tours to keep size of population reasonable.

Widely used.

Eg: TSP
Vehicle routing
Quadratic assignment problems
VLSI Component placement
Job Shop Scheduling.
Database query optimization.
Satisfiability.