Metaheuristic Optimization Methods: Guided Local Search

Guided Local Search (GLS)

- Chapter 7 from Handbook on Metaheuristics
- Based on GENET: Davenport A., Tsang E., Wang, C., Zhu K. (1994) GENET: A connectionist architecture for solving constraint satisfaction problems by iterative improvement, *Proceedings 12th National Conference for Artificial Intelligence*, p.325-330
- GLS is often said to be closely related to Tabu Search
- Tries to overcome local optima by removing them:
    - Changes the topography of the search space
    - Uses an extended move evaluation function
- Focuses on promising parts of the search space

Features of a Solution

- GLS assumes that we can find *features* of a solution that we can penalize
- What is a "feature"?
  - A characteristic that can help describe a set of solutions
- Examples:
  - **TSP**: all solutions in which city A follows immediately after city B in the tour
  - **Knapsack**: all solutions that include item #5

Features & GLS

- The modification of the move evaluation in GLS is based on features
- Each features has a cost associated with it
  - Represents (directly or indirectly) the influence of a solution on the (extended) move evaluation function
  - Constant or variable (dependent on other features)
  - GLS tries to avoid costly features
- We use an indicator function as follows:

$$I_i(s) = \begin{cases} 1, & \text{if solution } s \text{ has feature } i \\ 0, & \text{if solution } s \text{ does not have feature } i \end{cases}$$

Extended Move Evaluation

- Let the set of features be denoted by: $F = \{1, \dots, G\}$
- We have our indicator function:

$$I_i(s) = \begin{cases} 1, & \text{if solution } s \text{ has feature } i \\ 0, & \text{if solution } s \text{ does not have feature } i \end{cases}$$

- We create a penalty vector:

$$\mathbf{p} = [p_i],\ \forall i = 1, \dots, G$$

p_i is the number of times feature i has been penalized until now

- The extended move evaluation function becomes

$$f^*(s) = f(s) + \lambda \sum_{i=1}^{G} I_i(s) p_i$$

Penalties

$$f^*(s) = f(s) + \lambda \sum_{i=1}^{G} I_i(s) p_i$$

- The penalties are initially equal to 0
- When the search has reached a local optimum (with respect to the extended move evaluation function) the penalty is increased for some of the features of the current (locally optimal) solution

Penalties

- How to select which feature to penalize?
- Define the *utility* of a feature *i* in solution *s* as follows:

$$u_i(s) = I_i(s)\frac{c_i}{1+p_i}$$

Here, c_i is the cost of the feature (in objective function) and p_i is the current penalty value of the feature

- In a local optimum, s, increase the penalty for the feature that has the highest utility value, $u_i(s)$
- Note: Penalties are only adjusted when the search has reached a local optimum, and only for features included in the local optimum

Guided Local Search

input: starting solution, s_0
input: neighborhood operator, N
input: evaluation function, f
input: a set of features, F
input: a penalty factor, λ
$current \Leftarrow s_0$
$best \Leftarrow s_0$
$p_i \Leftarrow 0$ (for all $i \in F$)

initialization

while stopping criterion not met **do**
    Define $f^*(s) = f(s) + \lambda \sum_{i \in F} I_i(s) p_i$
    $s^* \Leftarrow$ the best solution in $N(current)$, according to f^*
    if $f^*(s^*) < f^*(current)$ **then**
        $current \Leftarrow s^*$
        if $f(current) < f(best)$ **then**
            $best \Leftarrow current$
        end if
    else
        Define the utility, $u_i(current) = I_i(current) \frac{c_i}{1+p_i}$, for all $i \in F$
        $p_i \Leftarrow p_i + 1$ for each feature $i \in F$ having the maximum utility in solution $current$
    end if
end while

main loop

Guided Local Search

input: starting solution, s_0
input: neighborhood operator, N
input: evaluation function, f
input: a set of features, F
input: a penalty factor, λ
$current \Leftarrow s_0$
$best \Leftarrow s_0$
$p_i \Leftarrow 0$ (for all $i \in F$)

initialization

while stopping criterion not met **do**
  Define $f^*(s) = f(s) + \lambda \sum_{i \in F} I_i(s) p_i$
  $s^* \Leftarrow$ the best solution in $N(current)$, according to f^*
  if $f^*(s^*) < f^*(current)$ **then**
    $current \Leftarrow s^*$
    if $f(current) < f(best)$ **then**
      $best \Leftarrow current$

$best \Leftarrow s_0$

$p_i \Leftarrow 0$ (for all $i \in F$)

while stopping criterion not met **do**

loop

  Define $f^*(s) = f(s) + \lambda \sum_{i \in F} I_i(s) p_i$

  $s^* \Leftarrow$ the best solution in $N(current)$, according to f^*

  if $f^*(s^*) < f^*(current)$ **then**

    $current \Leftarrow s^*$

    if $f(current) < f(best)$ **then**

      $best \Leftarrow current$

    end if

  else

    Define the utility, $u_i(current) = I_i(current) \frac{c_i}{1+p_i}$, for all $i \in F$

    $p_i \Leftarrow p_i + 1$ for each feature $i \in F$ having the maximum utility solution $current$

  end if

end while

Lambda

$$f^*(s) = f(s) + \lambda \sum_{i=1}^{G} I_i(s) p_i$$

- The control parameter λ dictates the influence of the penalty on the extended move evaluation function
  - Low value: intensification
  - High value: diversification
- Recommended: fraction of the objective function value at a local minimum $\hat{s}$

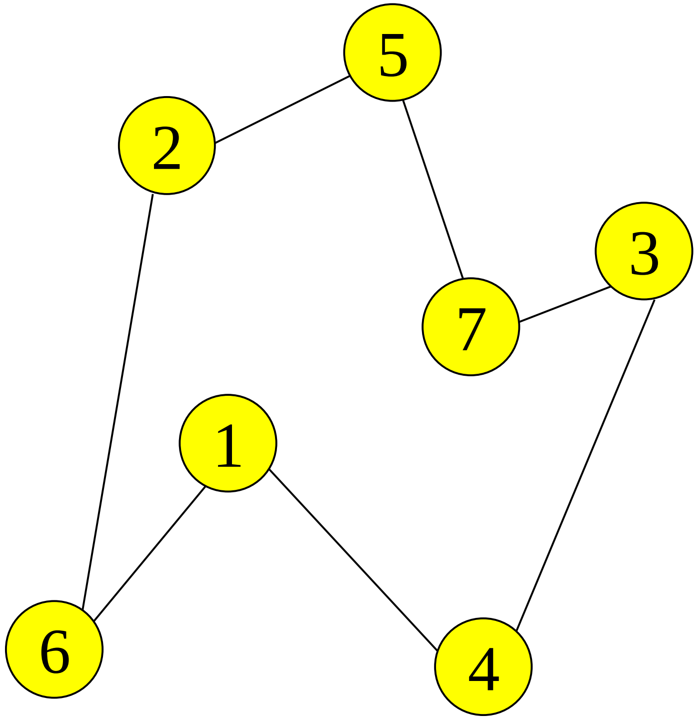
$$\lambda = \alpha \frac{f(\hat{s})}{\text{no. of features in } \hat{s}}$$

GLS - Example : TSP

- Features: edges included
- Cost of the features: edge length
- Functions

$$f^*(s) = f(s) + \lambda \sum_{i=1}^{G} I_i(s) p_i \qquad u_i(s) = I_i(s) \frac{c_i}{1 + p_i}$$

- The feature associated with e_{26} will be penalized in the solution on the right:
  - In next round, move evaluation function is same as before, $f(s)$, except if e_{26} is in the solution, when the value will be $f(s) + \lambda$

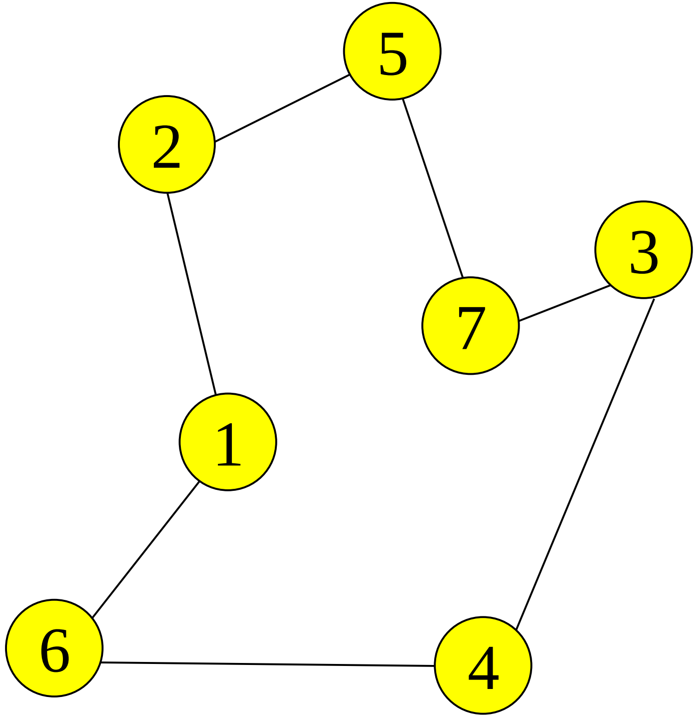


	1	2	3	4	5	6	7
1		0	0	0	0	0	0
2			0	0	0	1	0
3				0	0	0	0
4					0	0	0
5						0	0
6							0

GLS - Example : TSP

- After the next local optimum, e_{34} is penalized
- After this the move evaluation function is as before, $f(s)$, except if e_{26} or e_{34} is in the solution

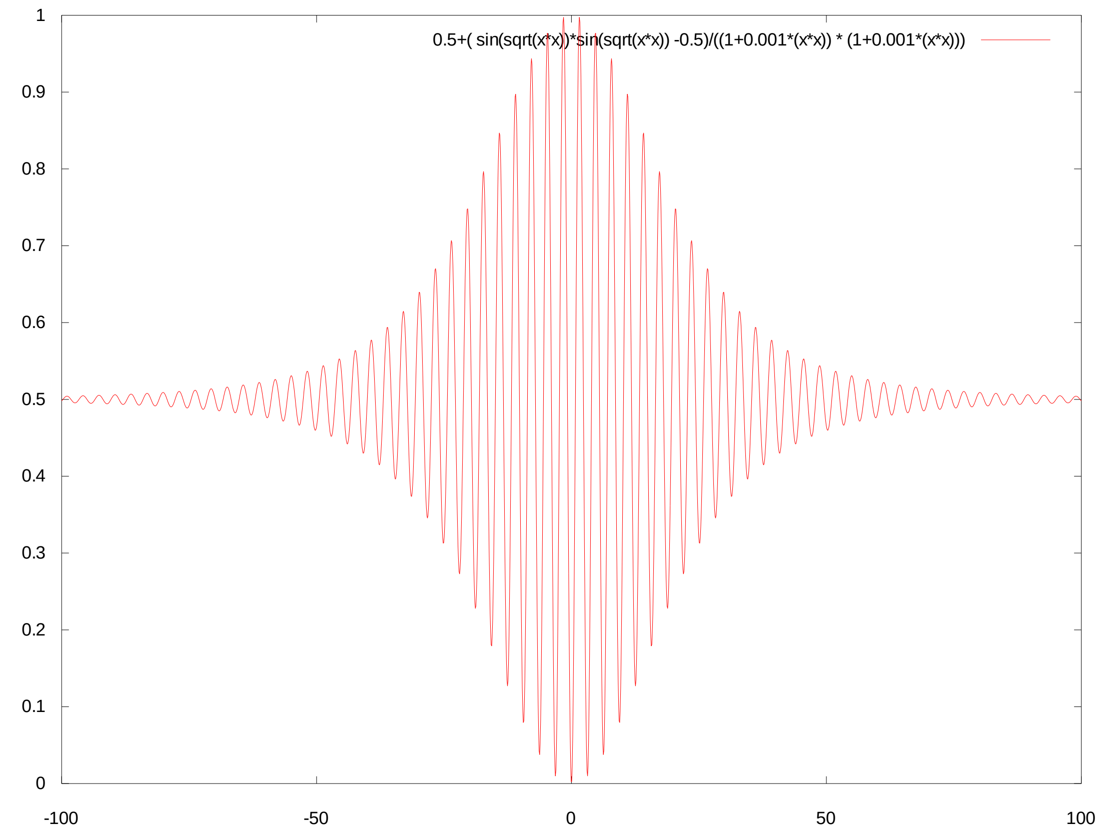
$$f^*(s) = f(s) + \begin{cases} 0 & e_{26} \text{ and } e_{34} \text{ not used in } s \\ \lambda & \text{one of } e_{26} \text{ and } e_{34} \text{ used in } s \\ 2\lambda & e_{26} \text{ and } e_{34} \text{ both used in } s \end{cases}$$



	1	**2**	**3**	**4**	**5**	**6**	**7**
1		0	0	0	0	0	0
2			0	0	0	1	0
3				1	0	0	0
4					0	0	0
5						0	0
6							0

Possibilities and Extensions

- Limited life time of penalties
- Diminishing penalties
- Awards in addition to penalties
- Automatic regulation of λ
- New utility-functions to find features to penalize



- Has been used for function optimization, good results on:

$$F6(x,y) = 0.5 + \frac{\sin^2\sqrt{x^2+y^2} - 0.5}{\left[1+0.001(x^2+y^2)\right]^2}$$

GLS versus SA

- It is difficult in SA to find the right cooling schedule (problem dependent)
  - High temperature gives bad solutions
  - Low temperature gives convergence to a local minimum
  - SA is non-deterministic
- GLS visits local minima, but can escape
  - Not random up-hill moves as in SA
  - GLS is deterministic
  - Does not converge to a local minimum; penalties are added until the search escapes

GLS vs. Tabu Search

- Both have mechanisms to guide the Local Search away from local optima
  - GLS penalizes features in the solutions
  - TS bans (makes taboo) features in the solutions
- Both incorporate memory structures
  - GLS has the accumulated penalties
  - TS has different memory structures
    - Short term, long term, frequency, recency, …