Simulated Annealing

Simulated annealing (SA)

- First approach in local search to escape local optima (Kirkpatrick et al. 1983)
- Inspired by the physical process of annealing to obtain a strong crystalline structure
  $\rightarrow$ **nature inspired method**
- Basic idea:
  Even worse solutions are accepted with a certain probability.
- Usually, SA is based on **random neighbor step function.**

procedure simulated annealing

```
begin
   t ← 0;
   T ← T_init;
   x ← initial solution;
   repeat:
      repeat:
         choose a x' ∈ N(x) randomly;
         if x' is better than x then
            x ← x';
         else
            if P < e^(−|f(x')−f(x)|/T) then
               x ← x';
         t ← t + 1;
      until equilibrium condition satisfied;
      T ← g(T, t);
   until stopping criteria satisfied;
end
```

Metropolis-criterion:

$P < e^{\frac{-|f(x')-f(x)|}{T}}$

$P \ldots$ random number $\in [0, 1)$

Annealing:

Temperature T

is slowly degraded

Parameters for the cooling schedule

- Initial temperature T_{init}
  - e.g., based on known bounds: $T_{\mathrm{init}} = f_{\mathsf{max}} - f_{\mathsf{min}}$
  - e.g., so that $\approx 3\%$ of the moves are rejected at the beginning

Parameters for the cooling schedule

- Initial temperature T_{init}
  - e.g., based on known bounds: $T_{\text{init}} = f_{\mathsf{max}} - f_{\mathsf{min}}$
  - e.g., so that $\approx 3\%$ of the moves are rejected at the beginning
- Cooling schedule $g(T, t)$
  - **Geometric cooling:**
    $g(T, t) = T \cdot \alpha$, $\alpha < 1$, (e.g., $\alpha = 0.95$)

Parameters for the cooling schedule

- Initial temperature T_{init}
  - e.g., based on known bounds: $T_{\mathrm{init}} = f_{\mathsf{max}} - f_{\mathsf{min}}$
  - e.g., so that $\approx$ 3% of the moves are rejected at the beginning
- Cooling schedule $g(T, t)$
  - **Geometric cooling:**
    $g(T, t) = T \cdot \alpha$, $\alpha < 1$, (e.g., $\alpha = 0.95$)
- Number of iterations to satisfy equilibrium condition
  - e.g., multiple of the neighborhood size

Parameters for the cooling schedule

- Initial temperature T_{init}
  - e.g., based on known bounds: $T_{\text{init}} = f_{\max} - f_{\min}$
  - e.g., so that $\approx 3\%$ of the moves are rejected at the beginning
- Cooling schedule $g(T, t)$
  - **Geometric cooling:**
    $g(T, t) = T \cdot \alpha$, $\alpha < 1$, (e.g., $\alpha = 0.95$)
- Number of iterations to satisfy equilibrium condition
  - e.g., multiple of the neighborhood size
- Stopping criteria
  - e.g., reaching a final temperature
  - or no improvement over τ levels of temperature

Example: simulated annealing for the TSP

- Simple implementation:

  (Johnson, McGeoch, 1997)

  - Start with a random tour
  - 2-exchange neighborhood
  - Cooling schedule:
    - T_{init} so that $\approx 3\%$ of moves are rejected
    - Geometric cooling, $\alpha = 0.95$
    - Iterations to satisfy equilibrium: $n \cdot (n-1)$
    - Stopping criteria: 5 temperature levels without improvement

Example: simulated annealing for the TSP

- Simple implementation:

  (Johnson, McGeoch, 1997)

  - Start with a random tour
  - 2-exchange neighborhood
  - Cooling schedule:
    - T_{init} so that $\approx 3\%$ of moves are rejected
    - Geometric cooling, $\alpha = 0.95$
    - Iterations to satisfy equilibrium: $n \cdot (n-1)$
    - Stopping criteria: 5 temperature levels without improvement

$\rightarrow$ relatively poor results

Example: simulated annealing for the TSP (cont.)

- Significant improvements
  - Use construction heuristic for initial solution
  - Restriction of the neighborhood to promising moves

Example: simulated annealing for the TSP (cont.)

- Significant improvements
  - Use construction heuristic for initial solution
  - Restriction of the neighborhood to promising moves
- Nevertheless
  Not competitive compared to leading TSP methods.

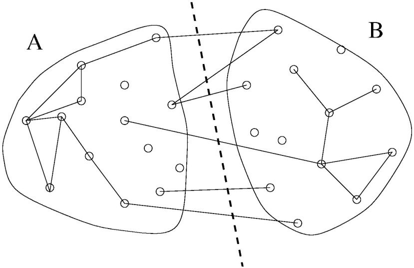
Example: SA for graph bipartitioning

Definition (graph bipartitioning)

Given: graph $G = (V, E)$

Wanted: partitioning of G into two vertex sets V_1, V_2, with $|V_1| = |V_2|$, $V_1 \cap V_2 = \emptyset$, and $V_1 \cup V_2 = V$,

minimize $|\{(u, v) \in E \mid u \in V_1 \wedge v \in V_2\}|$



Example: SA for graph bipartitioning (cont.)

- Solution representation:

  Characteristic vector $x = (x_1, \ldots, x_n)$, $n = |V|$
  $\rightarrow$ node i is assigned to set V_{x_i}

Example: SA for graph bipartitioning (cont.)

- Solution representation:

  Characteristic vector $x = (x_1, \ldots, x_n)$, $n = |V|$
  $\rightarrow$ node i is assigned to set V_{x_i}

- Simulated annealing (Johnson et al., 1989)
  - 2-node-exchange neighborhood
  - Random initial solution
  - Similar cooling schedule as for the TSP
  - One of the first applications of SA

Example: SA for graph partitioning (cont.)

- Improvements:
  - **Restriction of the neighborhood and permission of infeasible solutions**
  - *Modified objective function:*
    $f(V_1, V_2) \;=\; |\{(u, v) \in E \mid u \in V_1 \wedge v \in V_2\}| + \gamma(|V_1| - |V_2|)^2$
    γ: imbalance factor
  - *New neighborhood:*

    Move single node into the other set
    - $|N(x)| = n$ instead of $n^2/4$

Example: SA for graph partitioning (cont.)

- Improvements:
  - **Restriction of the neighborhood and permission of infeasible solutions**
  - *Modified objective function:*
    $f(V_1, V_2) = |\{(u, v) \in E \mid u \in V_1 \wedge v \in V_2\}| + \gamma(|V_1| - |V_2|)^2$
    γ: imbalance factor
  - *New neighborhood:*

    Move single node into the other set
    - $|N(x)| = n$ instead of $n^2/4$
- Comparison with Kernighan-Lin heuristic for this problem:
  - SA better for random graphs
  - Kernighan-Lin better for Euclidean graphs

Enhancements of SA

- Nonmonotonic cooling schedule
  - *"Reheating"*
- Dynamic vs. static cooling schedule
  - e.g., equilibrium state satisfied if no improvement was found after a certain number of iterations
- Deterministic investigation of the neighborhood
- Combination with other methods
- Parallelization

Conclusions for SA

- SA has been applied to several hundred applications.
- SA is one of the most commonly used and also theoretically most studied metaheuristics.
- Often it is simple to implement.
- SA delivers good results for many problems but usually no excellent results compared to other leading methods.
- SA is relatively time consuming.