The Basic GLSM Model

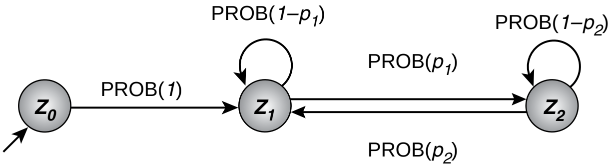
Many high-performance SLS methods are based on combinations of *simple (pure) search strategies* (*e.g.*, ILS, MA).

These hybrid SLS methods operate on two levels:

- **lower level:** execution of underlying simple search strategies
- **higher level:** activation of and transition between lower-level search strategies.

Key idea underlying Generalised Local Search Machines:
Explicitly represent higher-level search control mechanism in the form of a *finite state machine*.

Example: Simple 3-state GLSM



- States z_0, z_1, z_2 represent simple search strategies, such as Random Picking (for initialisation), Iterative Best Improvement and Uninformed Random Walk.
- PROB(p) refers to a probabilistic state transition with probability p after each search step.

Generalised Local Search Machines (GLSMs)

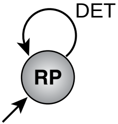
- States $\cong$ simple search strategies.
- State transitions $\cong$ search control.

- GLSM $\mathcal{M}$ starts in initial state.
- In each iteration:
  - $\mathcal{M}$ executes one search step associated with its current state z;
  - $\mathcal{M}$ selects a new state (which may be the same as z) in a nondeterministic manner.

- $\mathcal{M}$ terminates when a given termination criterion is satisfied.

Modelling SLS Methods Using GLSMs

Uninformed Picking and Uninformed Random Walk



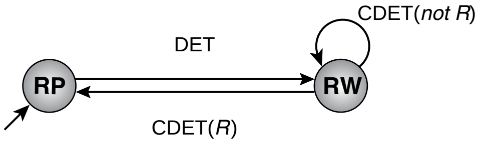
procedure *step-RP*(π, s)
input: *problem instance* $\pi \in \Pi$,
candidate solution $s \in S(\pi)$
output: *candidate solution* $s \in S(\pi)$

$s' := selectRandom(S)$;
return s'
end *step-RP*

procedure *step-RW*(π, s)
input: *problem instance* $\pi \in \Pi$,
candidate solution $s \in S(\pi)$
output: *candidate solution* $s \in S(\pi)$

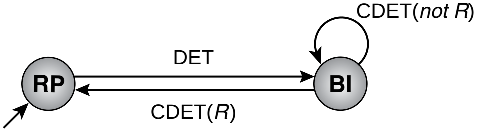
$s' := selectRandom(N(s))$;
return s'
end *step-RW*

Uninformed Random Walk with Random Restart



R = restart predicate, *e.g.*, countm(k)

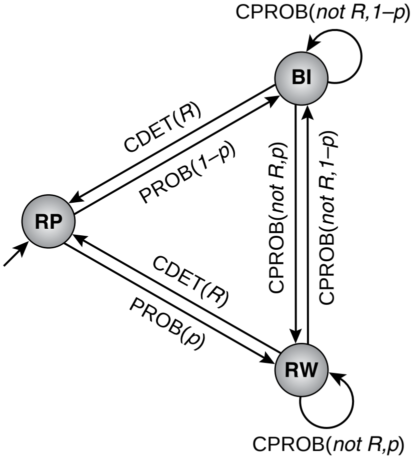
Iterative Best Improvement with Random Restart



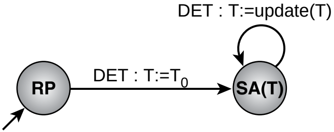
procedure *step-BI*(π, s)
input: *problem instance* $\pi \in \Pi$, *candidate solution* $s \in S(\pi)$
output: *candidate solution* $s \in S(\pi)$

$g^* := \min\{g(s') \mid s' \in N(s)\}$;
$s' := selectRandom(\{s' \in N(s) \mid g(s') = g^*\})$;
return s'
end *step-BI*

Randomised Iterative Best Improvement with Random Restart

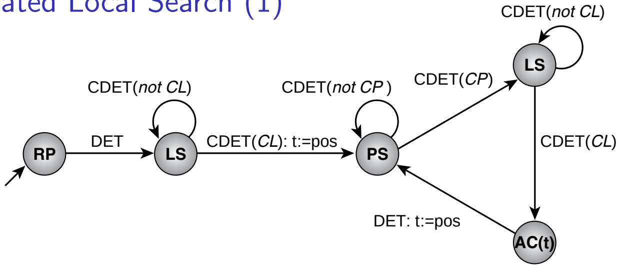


Simulated Annealing



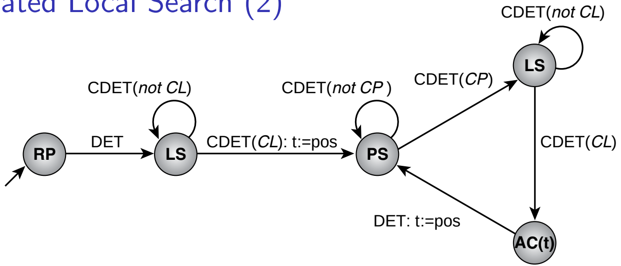
- Note the use of transition actions and memory for temperature T.
- The parametric state SA(T) implements probabilistic improvement steps for given temperature T.
- The initial temperature T_0 and function *update* implement the annealing schedule.

Iterated Local Search (1)



- The acceptance criterion is modelled as a state type, since it affects the search position.
- Note the use of transition actions for memorising the current candidate solution (pos) at the end of each local search phase.
- Condition predicates *CP* and *CL* determine the end of perturbation and local search phases, respectively; in many ILS algorithms, $CL :=$ lmin.

Iterated Local Search (2)



procedure *step-AC*(π, s, t)
    input: *problem instance* $\pi \in \Pi$,
        candidate solution $s \in S(\pi)$
    output: *candidate solution* $s \in S(\pi)$
    if $C(\pi, s, t)$ **then**
        return s
    else
        return t
    end
end *step-AC*

Ant Colony Optimisation (1)

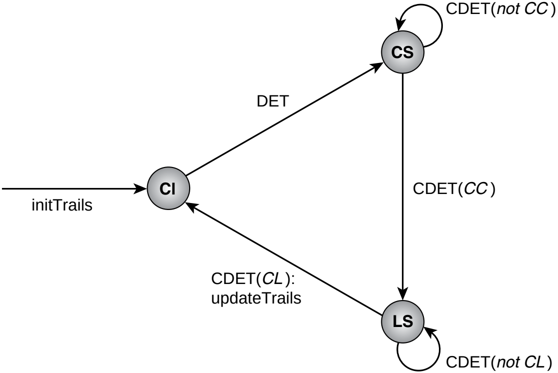
- General approach for modelling population-based SLS methods, such as ACO, as GLSMs:

  Define search positions as *sets of candidate solutions*; search steps manipulate some or all elements of these sets.

  Example: In this view, Iterative Improvement (II) applied to a population *sp* in each step performs one II step on each candidate solution from *sp* that is not already a local minimum.

  (Alternative approaches exist.)
- Pheromone levels are represented by memory states and are initialised and updated by means of transition actions.

Ant Colony Optimisation (2)



- The condition predicate *CC* determines the end of the construction phase.
- The condition predicate *CL* determines the end of the local search phase; in many ACO algorithms, $CL :=$ lmin.