Examples of Local Search (meta)heuristics

- *Simulated Annealing* [Kirkpatrick, Gelatt, Vecchi, 1982]: stochastic acceptance criterion
- *Steepest descent*: search the entire neighborhood and stop when there is no improvement
- *First improvement*: accept the first generated solution that improves the current one
- *K-opt local search* [Lin, 1965; Johnson and McGeoch, 1997]: based on the *k-change neighborhood* that has proved to be very successful for TSPs and similar problems. $M(s)$ prescribes that k edges are removed from the tour s and then replaced with other k edges. The best results are usually obtained for $k = 2$ and $k = 3$.
- *Iterated local search* [Lourenco, Martin and Stüzle, 2002]: iteratively, after hitting a local optimum the search is restarted and the new starting point is derived from some randomization of the current the local optimum
- *Genetic algorithms* [Holland, 1975; Goldberg, 1989]: the "current solution" is a point in S^n, n = population dimension

Simulated Annealing

procedure Simulated_Annealing()
  define_neighborhood_structure();
  $s \leftarrow$ get_initial_solution(S);
  $s_{best} \leftarrow s$;
  $T \leftarrow$ initial_temperature();
  while ($\neg\ stopping\_criterion$)
    $search\_neighborhood \leftarrow true$;
    while ($search\_neighborhood$)
      $s' \leftarrow$ sample_solution_from_neighborhood($\mathcal{N}(s)$);
      if ($f(s') > f(s)$)

$$p_{accept} \leftarrow exp\left(\frac{f(s) - f(s')}{T}\right);$$

      else
        $p_{accept} \leftarrow 1$;
      end if
      if (random() $< p_{accept}$)
        $s \leftarrow s'$;
        $search\_neighborhood \leftarrow false$;
        else
        $search\_neighborhood \leftarrow$ keep_searching_current_neighborhood();
      end if
      if ($s < s_{best}$)
        $s_{best} \leftarrow s$;
      end if
    end while
    $T \leftarrow$ update_temperature(T);
  end while
return s_{best};

Simulated Annealing

- Applied to both combinatorial and continuous problems with mixed success
- Proofs of asymptotic convergence available [Geman and Geman, 1984, Lundy and Mees, 1986]
- Temperature's schedule is critical to obtain good results in finite time; at the beginning temperature T is expected to be high in order to favor exploration, but it has to be gradually (e.g., logarithmically) and possibly monotonically decreased over the iterations

Tabu Search

procedure Tabu_Search()
  define_neighborhood_structure();
  $s \leftarrow$ get_initial_solution(S);
  $s_{best} \leftarrow s$;
  initialize_tabu_list();
  while ($\neg$ *stopping_criterion*)
    $\mathcal{N}_t(s) \leftarrow$ neighbor_solutions_that_do_not_violate_tabu_condition($\mathcal{N}(s)$);
    $\mathcal{N}_a(s) \leftarrow$ neighbor_solutions_that_meet_aspiration($\mathcal{N}(s)$);
    $s' \leftarrow$ get_best_solution($\mathcal{N}_t(s) \cup \mathcal{N}_a(s)$);
    update_tabu_list();
    if ($s < s_{best}$)
      $s_{best} \leftarrow s$;
    end if
  end while
return s_{best};

- Tabu criteria: recency, frequency, quality, influence
- Tabu criteria are applied to either complete solutions or solution components
- Applied to several problems with good success

Rollout: a construction metaheuristic

procedure Rollout_algorithm()
  $t \leftarrow 0$;
  $x_t \leftarrow \emptyset$;
  while $(x_t \notin S \vee \neg stopping\_criterion))$
    $c_t \leftarrow \arg\min_{c_k \in \mathcal{C}(x_t)} J\big(\mathcal{H}(x_t \oplus c_k)\big)$;
    $x_{t+1} \leftarrow x_t \oplus c_t$;
    $t \leftarrow t + 1$;
  end while
return x_t;

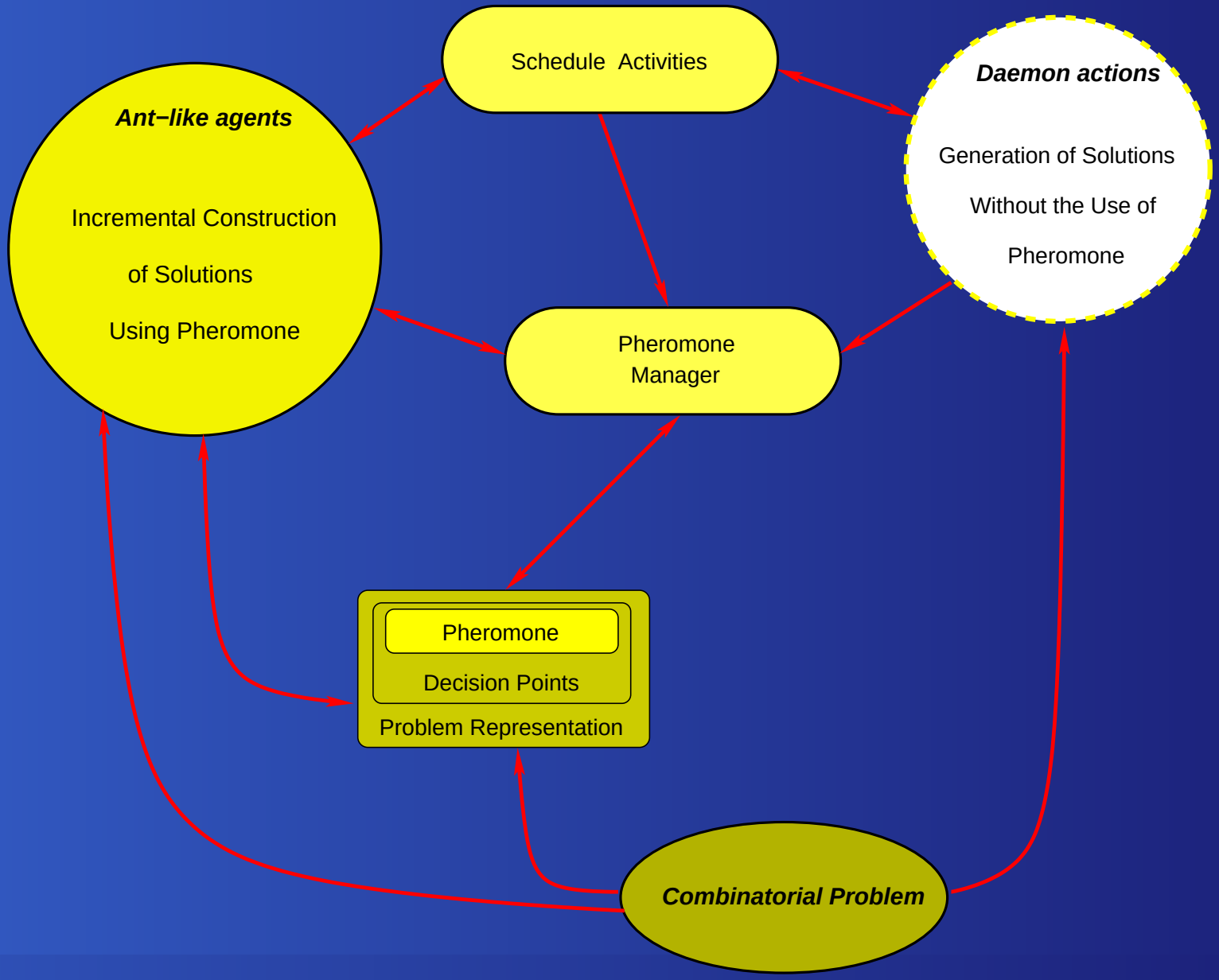
- The heuristic $\mathcal{H}$ takes a partial solution x_k and completes it into a feasible solution
- At each decision step t each feasible choice $c_t \in \mathcal{C}(x_t)$ is scored according to an estimate of the cost associated to the feasible solution that would result from the completion of the partial solution $(x_t \oplus c_t)$ by $\mathcal{H}$
- Instead of relying on systematic expansions of the partial solution (e.g., dynamic programming), the heuristic $\mathcal{H}$ is used to obtain an approximate (and possibly quick) evaluation of this cost. $\mathcal{H}$ can be any heuristic
- A pool of heuristics can be used and the heuristic providing the best expected cost can be used at each step
- Under mild mathematical conditions the rollout metaheuristic guarantees to improve the performance that could be obtained by using the base heuristic $\mathcal{H}$ [Bertsekas, Tsitsiklis and Wu, 1997]

ACO, bio-inspired multi-agent metaheuristic

```
procedure ACO_metaheuristic()
  while (¬ stopping_criterion)
    schedule_activities
      ant_agents_construct_solutions_using_pheromone();
      pheromone_updating();
      daemon_actions();      /* OPTIONAL */
    end schedule_activities
  end while
return best_solution_generated;
```

- Ant agents ***construct*** solutions according to stochastic decisional processes depending on pheromone variables (***stigmergic communication***)
- Philosophical assumption: ***memory*** and ***learning*** can be useful to solve combinatorial optimization problems

ACO's logical structure



ACO in words (1)

- According to some chosen schedule (e.g., groups of 10 ants at-a-time), ant agents are repeatedly generated. The task of each ant is to ***construct***, in a relatively simple and computationally light way, a solution for the problem at hand
- Starting from an empty solution, each ant during its ***forward*** journey constructs a possibly feasible solution by adding step-by-step components to the partial solution
- At each *construction step* an ant applies a ***stochastic decision policy*** to decide the next action, that is, the new solution component to include into the current partial solution
- The decision policy depends on two sets of variables, in some sense ***local to the decision step***, the ***pheromone variables*** and the ***heuristic variables***. Both these two sets of variables encode the desirability of issuing a specific decision to extend the current partial solution conditionally to the fact of being in the current decision step

ACO in words (2)

- Pheromone variables, as in the case of the ants, encode the value of ***desirability of a local choice as collectively learned*** from the so far generated solutions
- Heuristic variables assign a value of desirability on the basis of either a priori knowledge about the problem or as the outcome of a process independent of the ants (e.g., the computation of a lower bound estimate)
- Pheromone variables which bias the probabilistic decisions of the ants, are in turn repeatedly updated during algorithm execution to reflect the incremental knowledge about the characteristics of the solution set that has been acquired through the same solution generation processes
- After building a solution, metaphorically (or in practice) the ant reports the solution to a sort of *pheromone manager*, which authorizes or not the ant to update the pheromone variables associated to the built solution according to its *evaluation*

ACO in words (3)

- In the positive case, the ant starts its *backward* journey, retracing its solution and updating pheromone values, usually of an amount proportional to the *evaluated quality* of the solution. In this way, decisions associated to solutions which are either of good quality or are chosen more often, will likely have associated higher levels of pheromone, that is, higher local desirability
- The process is iterated over time and can happen in a distributed or centralized way

Some facts about ACO

- It has been applied with success to a number of classical combinatorial problems: traveling salesman, quadratic assignment, graph coloring, sequential ordering, set covering, job scheduling, . . . and problems of adaptive routing in different types of networks
- Performance are very good (better or comparable to state-of-the-art) in the case of routing problems and in the case of several combinatorial problems (usually when a local search daemon procedure is also used)
- It is the most popular and effective framework inspired by an ant behavior and making use of stigmergy
- The workshop *ANTS* is held every two years and gathers between 50 and 100 participants (`http://iridia.ulb.ac.be/ ants/ants2004/`)
- For more information and extensive discussions and review of applications refer to [Dorigo and Di Caro, 1999; Dorigo, Di Caro and Gambardella, 1999; Dorigo and Stutzle, 2004; Dorigo, Di Caro and Sampels, 2002; Di Caro 2004]

Ant System, the first ACO for TSP, 1991

procedure AS-ant-agent_life_cycle()
  $i \leftarrow 0$;
  $x_i \leftarrow$ get_starting_city();
  $c_i \leftarrow x_i$;
  $J(x_i) \leftarrow 0$;
  $\mathcal{H}(0) \leftarrow \{x_0, c_0, J(x_0)\}$;
  while $(|x_i| \neq N)$
    foreach $c_j \in \mathcal{N}_{x_i}(c_i)$ **do**
      $a_{ij} \leftarrow \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$;
    end foreach
    $c \leftarrow$ apply_AS_stochastic_decision_rule$(\mathcal{A}_{x_i}(c_i))$;
    $x_{i+1} \leftarrow (x_i, c)$;
    $c_{i+1} \leftarrow c$;
    $J(x_{i+1}) \leftarrow J(x_i) + \mathcal{J}(c_{i+1}|c_i)$;
    $\mathcal{H}(i+1) \leftarrow \{c_{i+1}, x_{i+1}, J(x_{i+1})\}$;
    $i \leftarrow i + 1$;
  end while
  $s \leftarrow x_i$;
  $J(s) \leftarrow J(x_i) + \mathcal{J}(c_0|c_i)$;
  foreach $c_i, c_j \in \mathcal{H},\ i = 0, 1, 2 \ldots, N-1,\ j = i+1$ **do**
    $\tau_{ij} \leftarrow \tau_{ij} + 1/J(s)$;
  end foreach
  removal_from_the_system();
end procedure

Ant System (2)

- Pheromone evaporation for exploration:

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t), \qquad \forall i,j \in \{1,\ldots,N\},\ \rho \in (0,1]$$

- Ant decision rule:

$$p_{ij}^{k}(t) = \frac{a_{ij}}{\sum_{c_n \in \mathcal{N}_{x^k}(c_i)} a_{in}^{k}(t)}, \qquad a_{ij} = \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$$

- However many other choices are possible . . . [Dorigo and Di Caro, 1999; Dorigo and Stutzle, 2004; Di Caro 2004]

Cultural algorithms [Reynolds, 1994]

procedure Cultural_Algorithm()
  $t \leftarrow 0$;
  $\mathcal{P}(t) \leftarrow$ initialize_population();
  $\mathcal{B}(t) \leftarrow$ initialize_belief_space();
  evaluate_population($\mathcal{P}(t)$);
  while $(\neg\, termination\_condition)$
    communicate($\mathcal{P}(t), \mathcal{B}(t)$);
    $\mathcal{B}(t) \leftarrow$ adjust_belief_space($\mathcal{B}(t)$);
    communicate($\mathcal{B}(t), \mathcal{P}(t)$);
    $\mathcal{P}(t+1) \leftarrow$ select($\mathcal{P}(t)$);
    $\mathcal{P}(t+1) \leftarrow$ evolve($\mathcal{P}(t+1)$);
    evaluate_population($\mathcal{P}(t+1)$);
    $t \leftarrow t+1$;
  end while
return $best\_solution\_generated$;

Cultural algorithms (2)

- Derived from cultural evolution process which support the basic mechanisms for cultural change
- Population-based: each individual has behavioral traits that can be modified and exchanged by socially motivated operators (*micro-evolutionary level*)
- At the *macro-evolutionary level*, individuals experiences (generated solutions) are evaluated and then collected, merged, generalized, and specialized in a shared *belief space*
- The two levels interact through a *communications protocol*

The End

Thanks for listening, I hope it was (it will be) useful. Good luck for your studies and career!