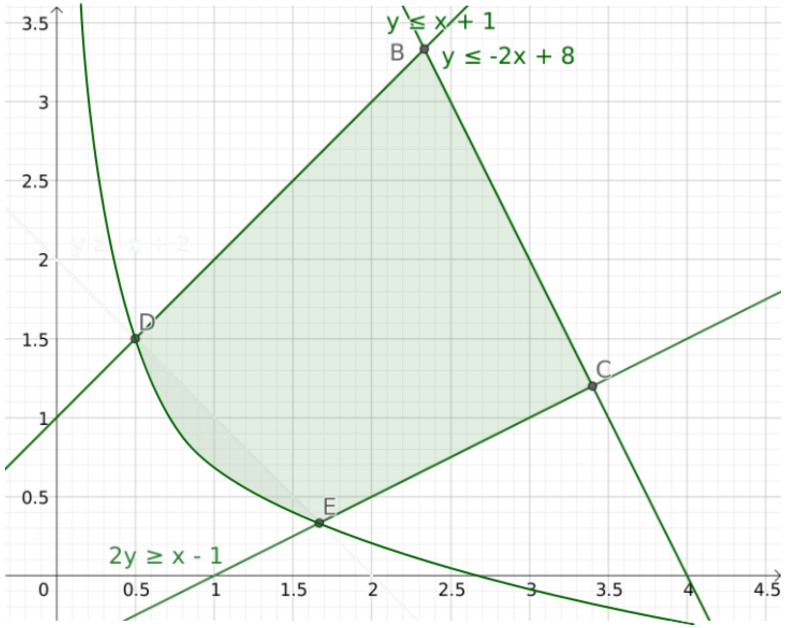
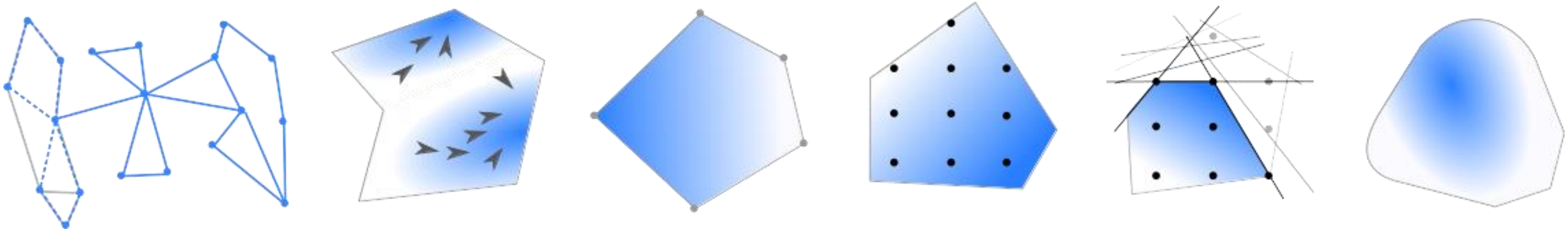
Meta-Heuristics 1

COMP4691 / 8691

R

A2





Previously on COMP4691(8691)

- Construct ← TUE
- Improve
  - $\boxed{\text{(Stochastic)}}$ Local Search
  - Simulated Annealing

← FRi

MH

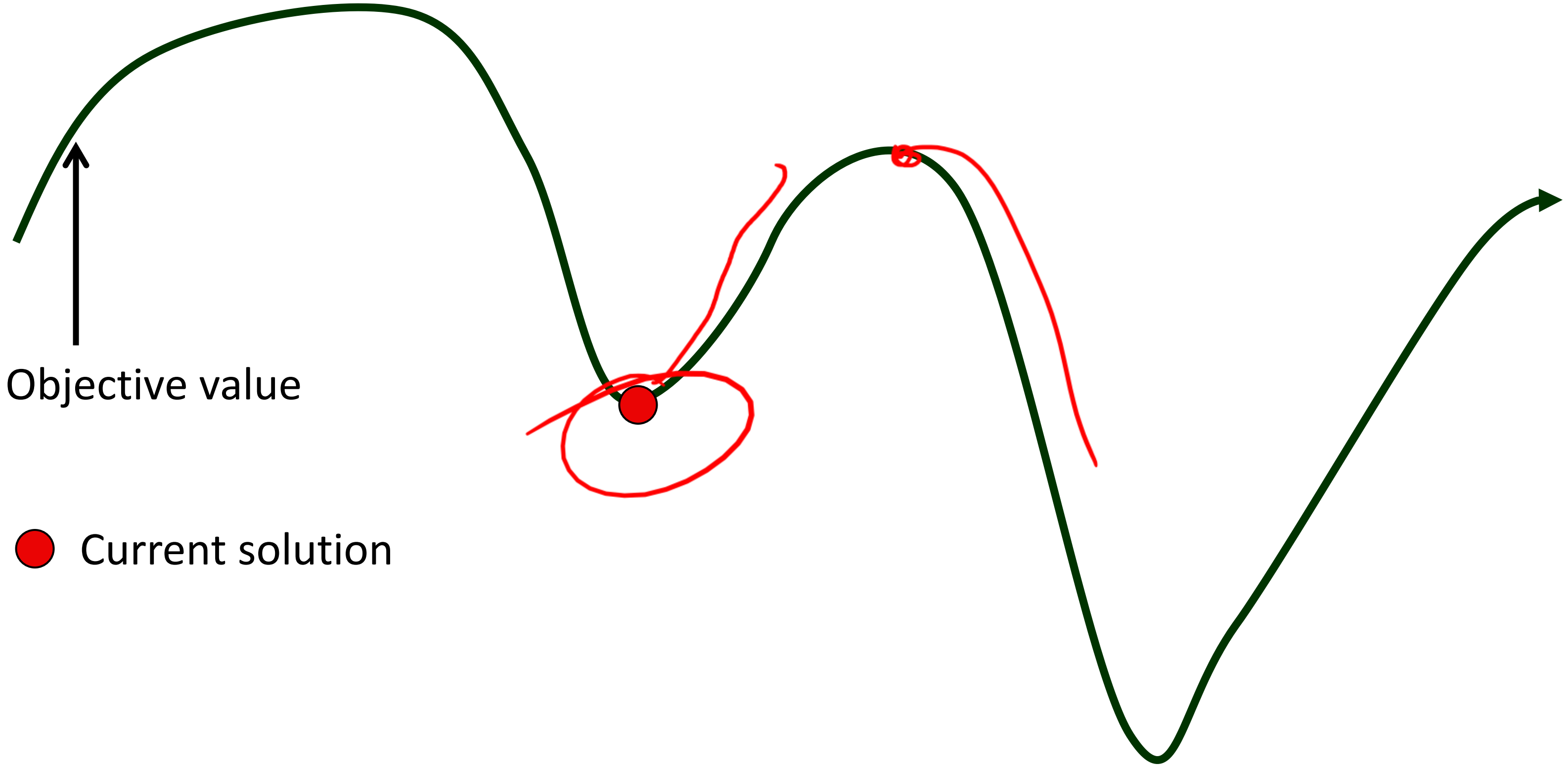
LOCAL SEARCH

Today:

- Other ways to escape local minima / search the solution space

Problems with Local Search I

Local minima



Meta-heuristics: Properties (1)

- can address both discrete- and continuous-domain optimisation problems
- are strategies that "guide" the search process
  LOCAL S.
- range from simple local search procedures to complex adaptive learning processes
- efficiently explore the search space to find good (near-optimal) feasible solutions
- provide no guarantee of global or local optimality
  B&B
- are agnostic to the unexplored feasible space (i.e., no "bound" information)
- lack a metric of "goodness" of solution (often stop due to an external time or iteration limit)

Meta-heuristics: Properties (2)

- are not based on some algebraic model (unlike exact methods)
- can be used in conjunction with an exact method
  - E.g., use meta-heuristic to provide upper bounds ← FEAS SOL
  - E.g., use restricted MILP as "local heuristic" (== matheuristic)
- are usually non-deterministic LNS
- are not problem specific (but their subordinate heuristics can be)
- may use some form of memory to better guide the search

VRP 3-OPTS

Meta-heuristics: Overview

EXPLOITATION VS EXPLORATION

Exhibit Intensification and Diversification

- Need to do both ←
- Can be explicitly controlled

Intensification (Exploitation):

- Concentrate search around already-found “good” solutions
- Look harder in a smaller area

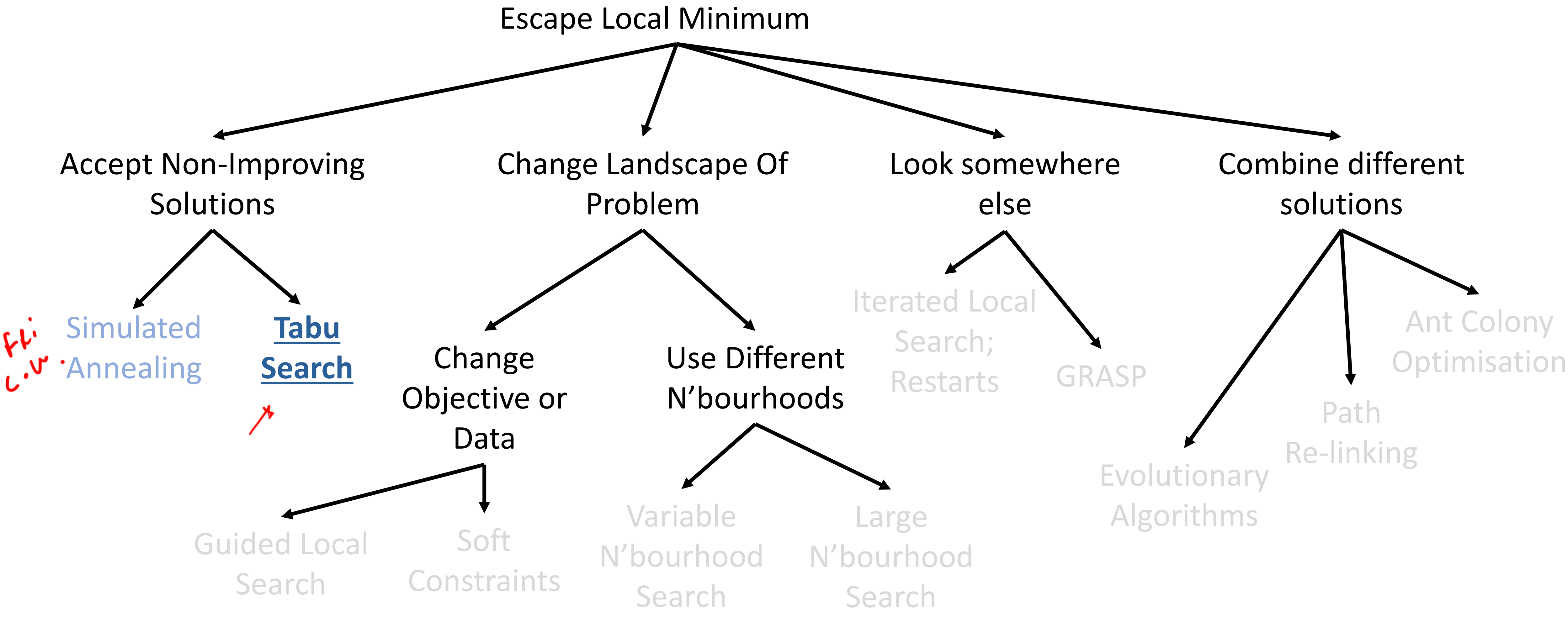
Diversification (Exploration)

- Expand the area being looked at
- Find new (promising?) areas to search
- Includes mechanisms to avoid getting trapped in confined areas of the search space

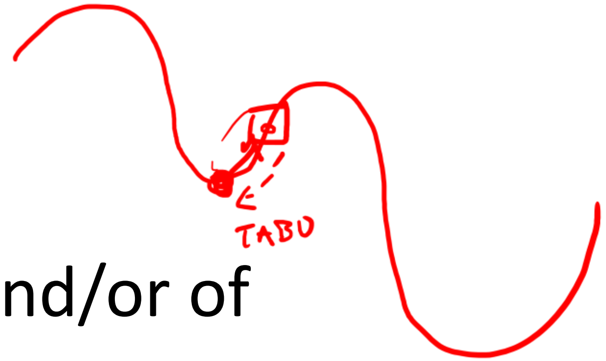
Meta-heuristics: An Incomplete Survey

- Escape Local Minimum
  - Accept Non-Improving Solutions
    - Simulated Annealing
    - Tabu Search
  - Change Landscape Of Problem
    - Change Objective or Data
      - Guided Local Search
      - Soft Constraints
    - Use Different N'bourhoods
      - Variable N'bourhood Search
      - Large N'bourhood Search
  - Look somewhere else
    - Iterated Local Search; Restarts
    - GRASP
  - Combine different solutions
    - Evolutionary Algorithms
    - Path Re-linking
    - Ant Colony Optimisation

Meta-heuristics: An Incomplete Survey



Tabu Search



- **Taboo** (*English*): prohibited, disallowed, forbidden
- **Tabu** (*Fijian*): forbidden to use due to being sacred and/or of supernatural powers

- Starts with classic Local Search until a local minimum is found
- Choose an objective-increasing move
- Make undoing that move "tabu"
  - Place it on a "tabu list"
- Repeat

$s \leftarrow$ GenerateInitialSolution()
$TabuList \leftarrow \emptyset$
while termination conditions not met **do**
    $s \leftarrow$ ChooseBestOf($\mathcal{N}(s) \setminus TabuList$)
    Update($TabuList$)
endwhile

Tabu Search

- Simple version: Tabu list has fixed length
  - Moves "fall off" the list after fixed number of iterations

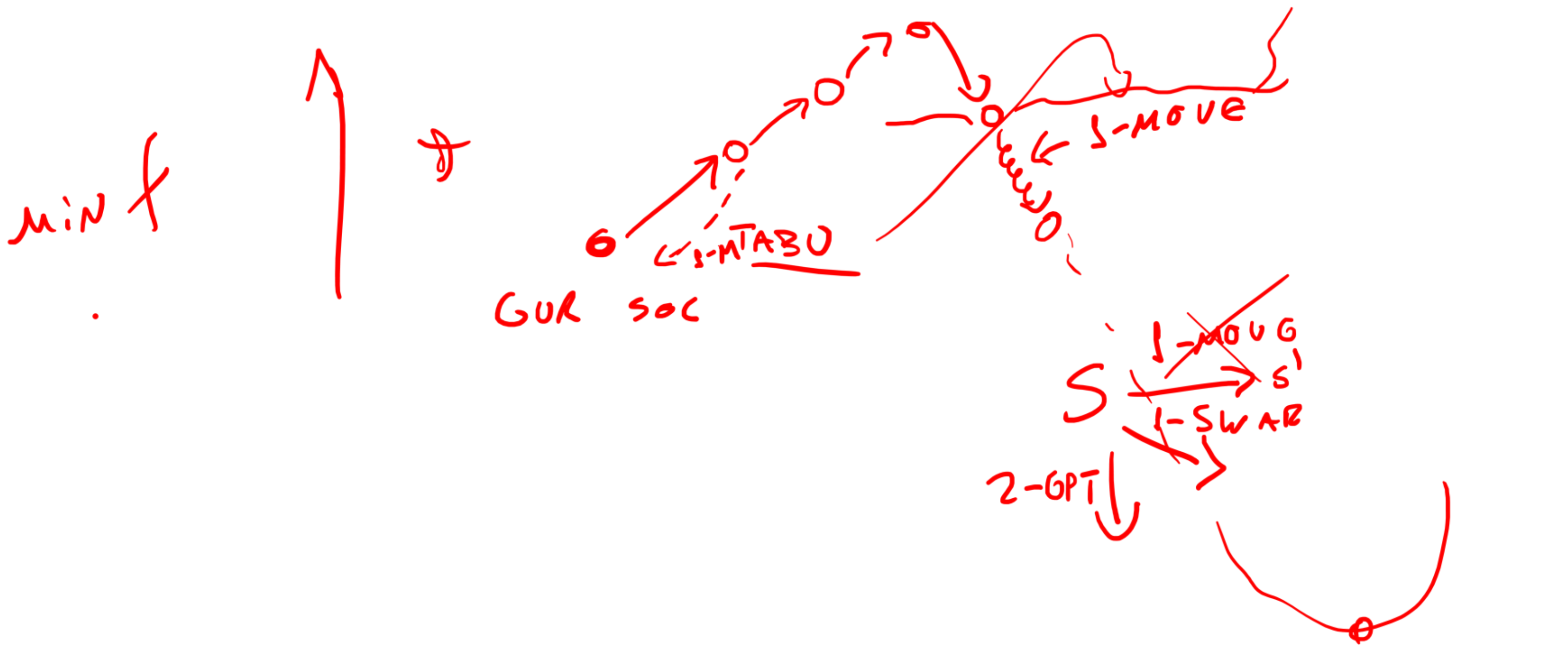
- Length of the list is a critical parameter
  - Too small → Keep falling back into same local minimum
  - Too big → Not allowed to explore new space properly

- Dynamic list length
  - If you see the same solution, increase list length
  - Decrease when new incumbent found

Tabu Search

"Aspiration Criteria"

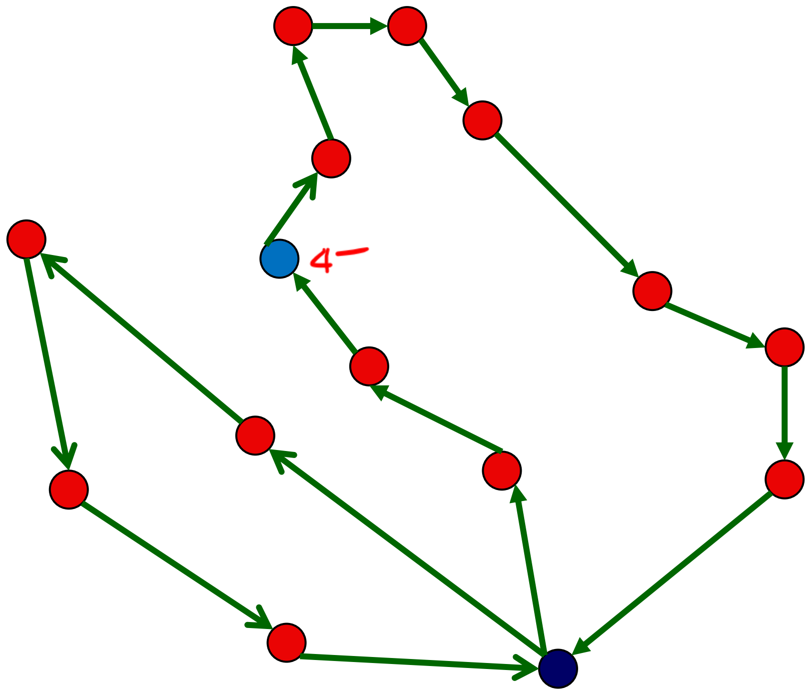
- Allowed to keep a solution if it meets certain criteria
- Most common: a new incumbent / best solution is kept



Tabu Search – VRP

E.g. VRP

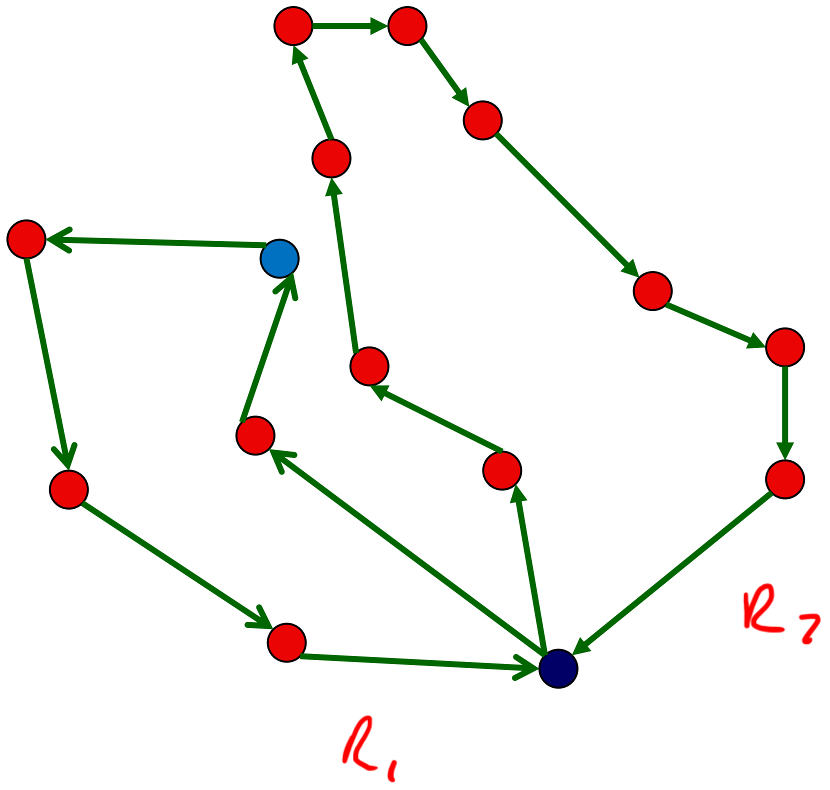
- 1-move



Tabu Search – VRP

E.g. VRP

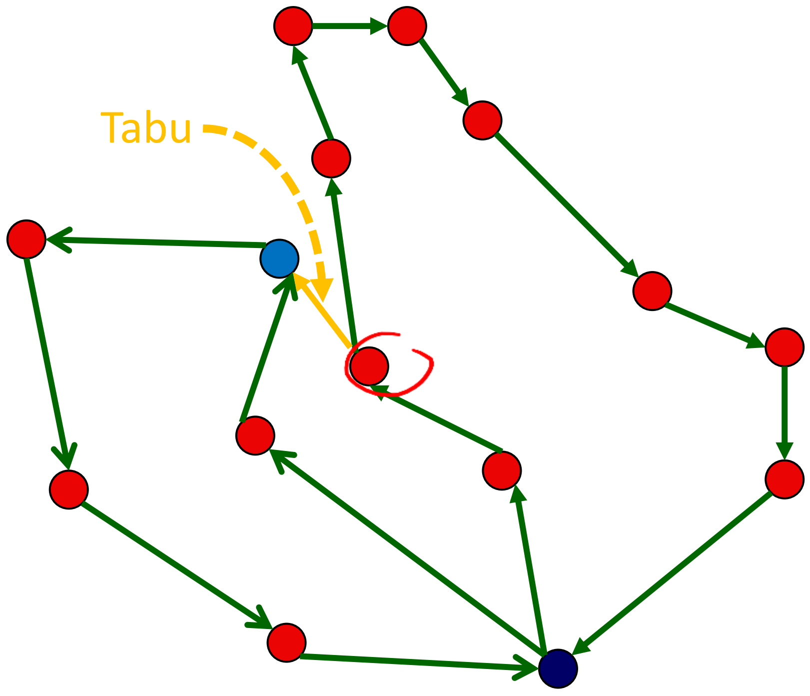
- 1-move



Tabu Search – VRP

E.g. VRP

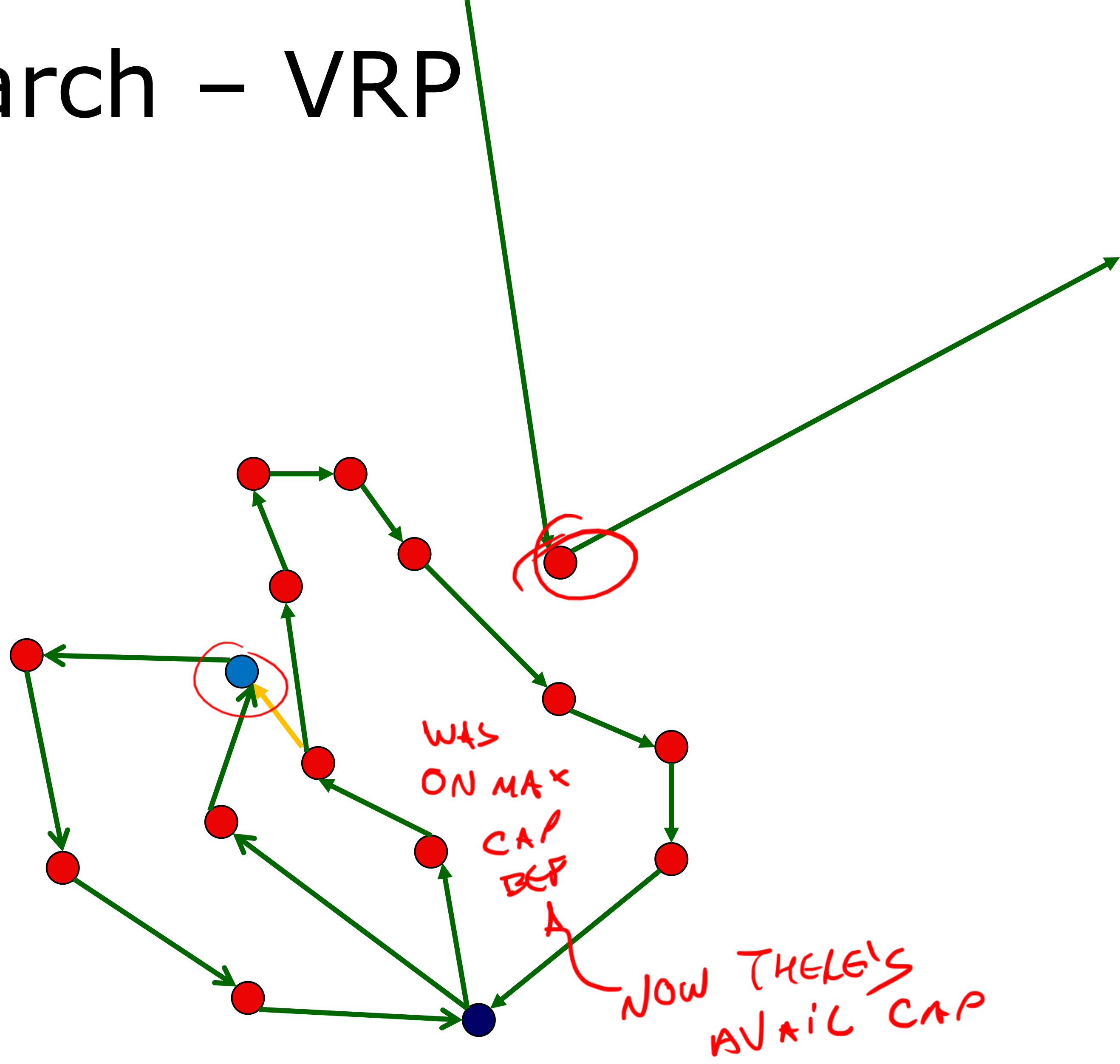
- 1-move



Tabu Search – VRP

E.g. VRP

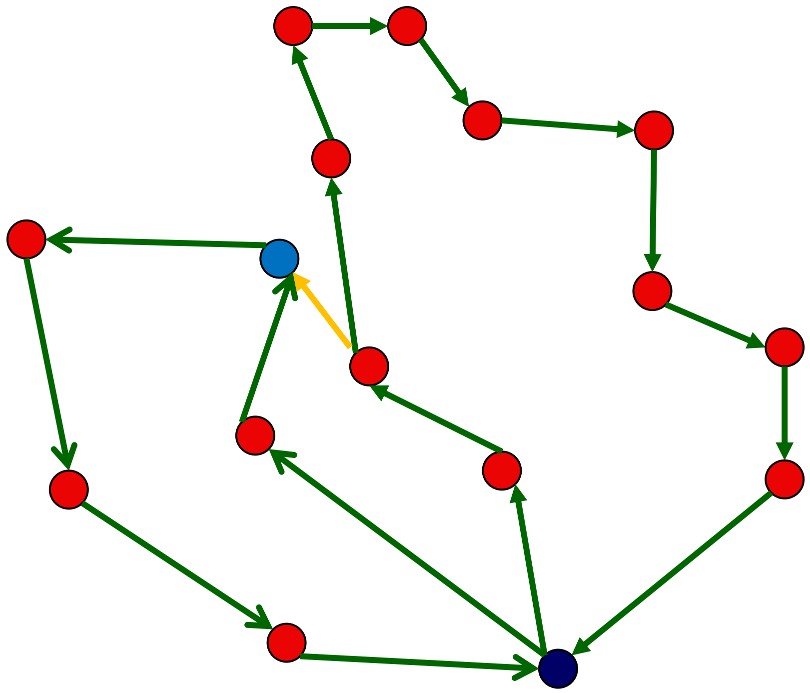
- 1-move



Tabu Search – VRP

E.g. VRP

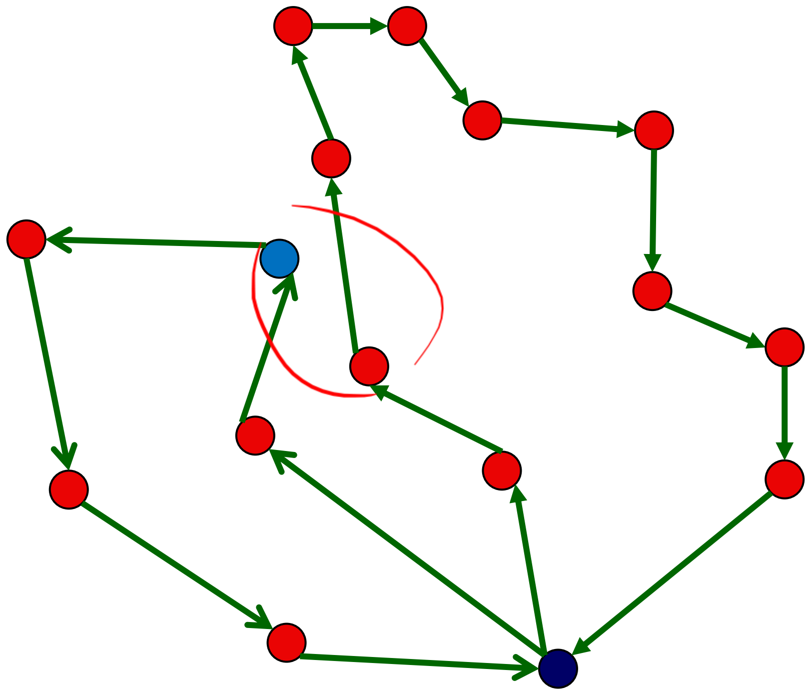
- 1-move



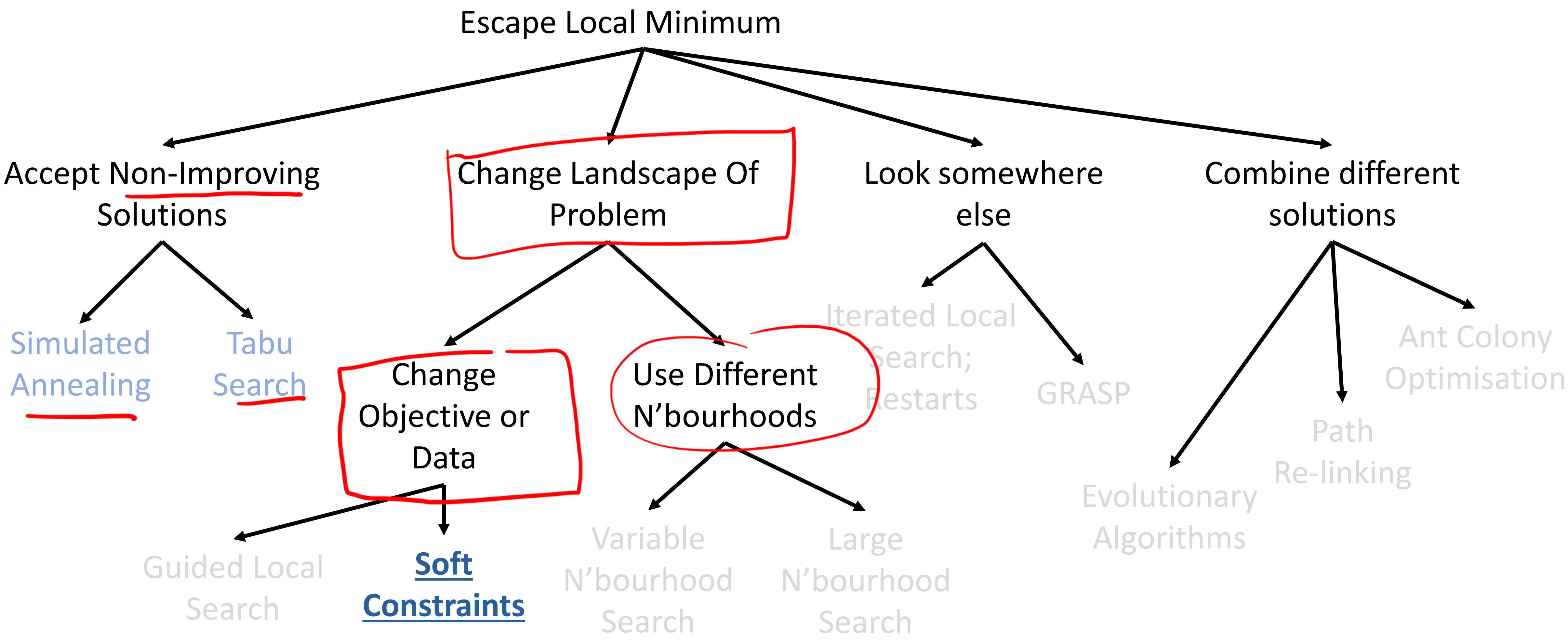
Tabu Search – VRP

E.g. VRP

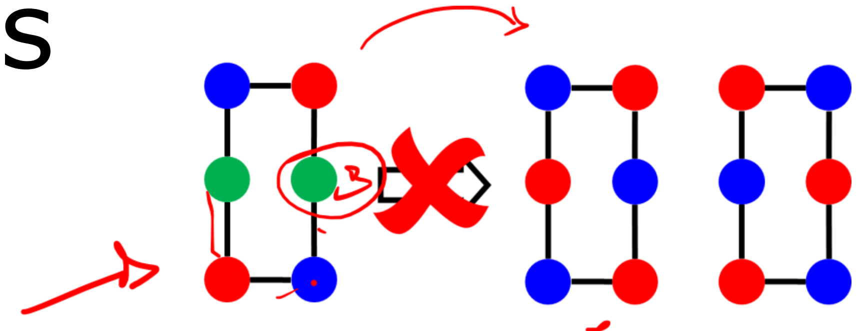
- 1-move



Meta-heuristics: An Incomplete Survey



Soft Constraints



General idea: Move constraints into the objective

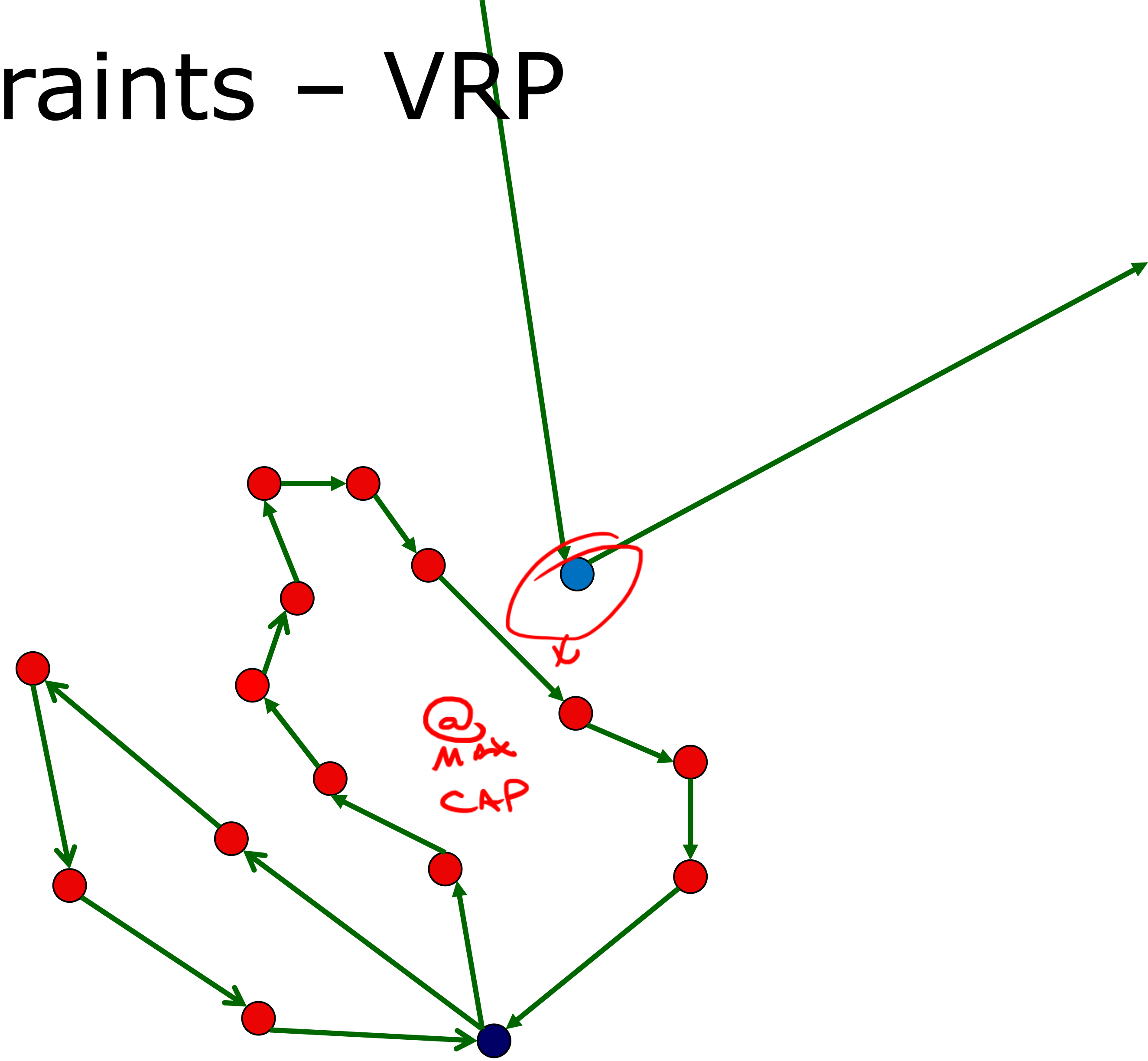
- Relax Constraints
  - Penalise degree of violation
    - Allows Local Search to move through "infeasibility barrier"
  - Opens new areas of search space
  - Maintain both best feasible solution, "best" infeasible solution
  - Increase penalty over time to force incumbent back to feasibility
  - Often used with Simulated Annealing or Tabu Search
  - Extensively used in practice, e.g., VRP

- Parameters:
  - Which constraints to relax
  - Penalty schedule

Soft Constraints – VRP

E.g. VRP

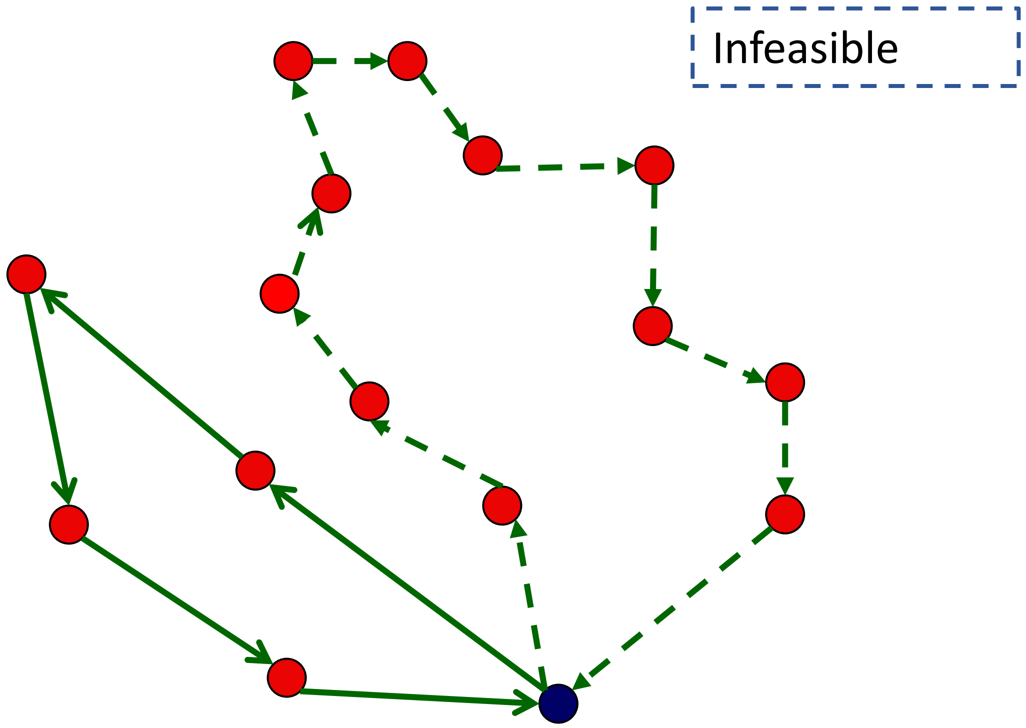
- 1-move



Soft Constraints + Tabu

E.g. VRP

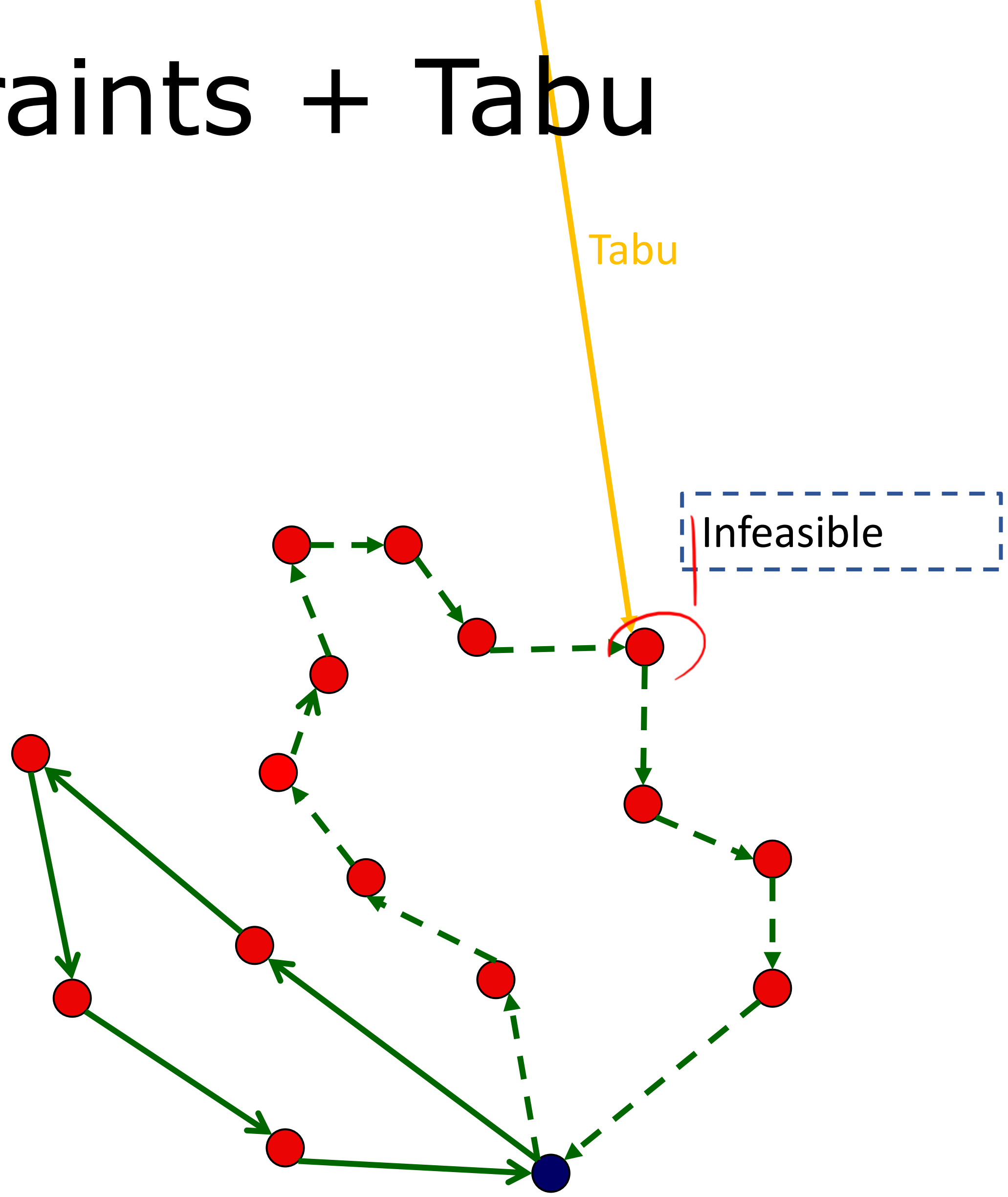
- 1-move



Soft Constraints + Tabu

E.g. VRP

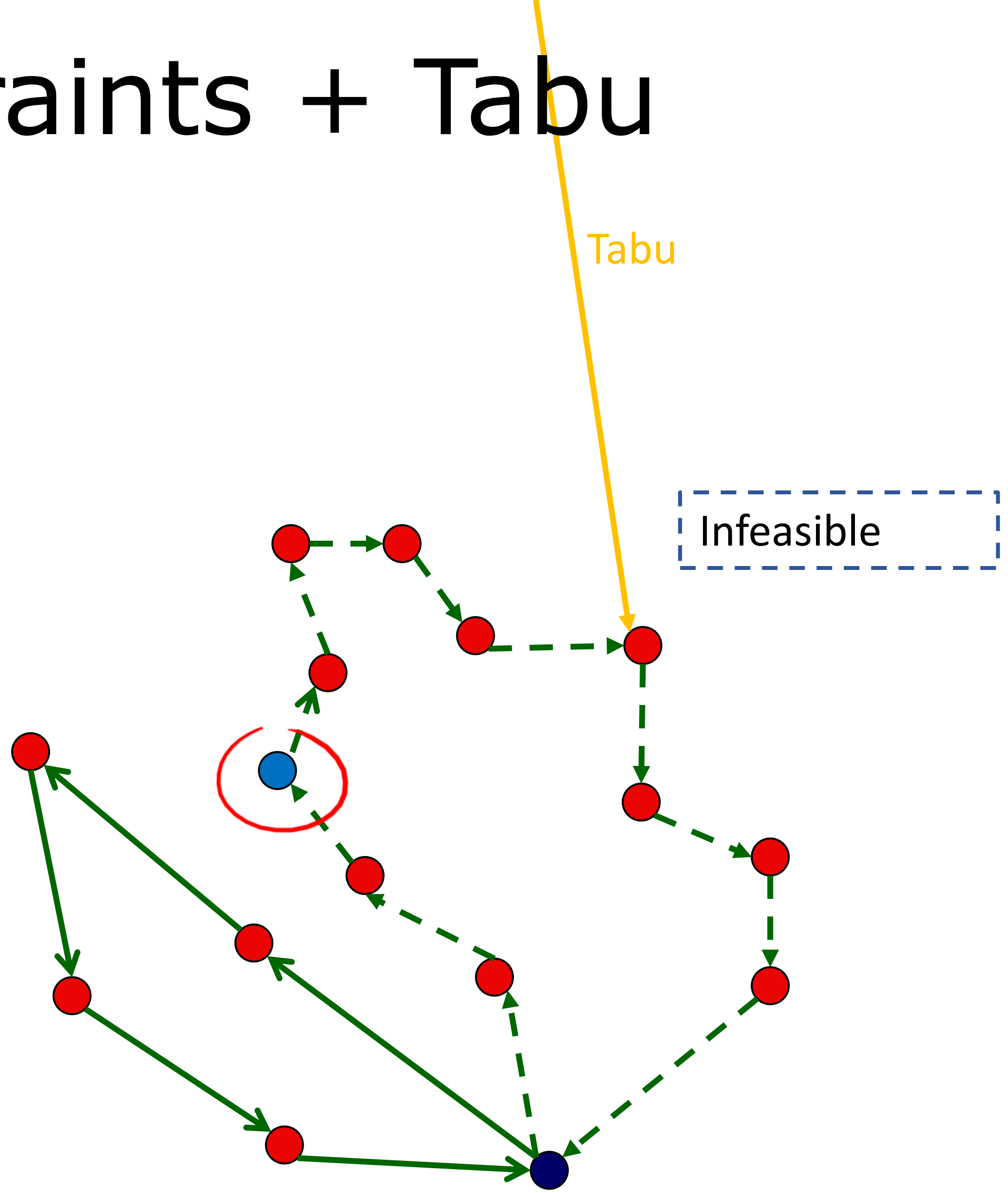
- 1-move



Soft Constraints + Tabu

E.g. VRP

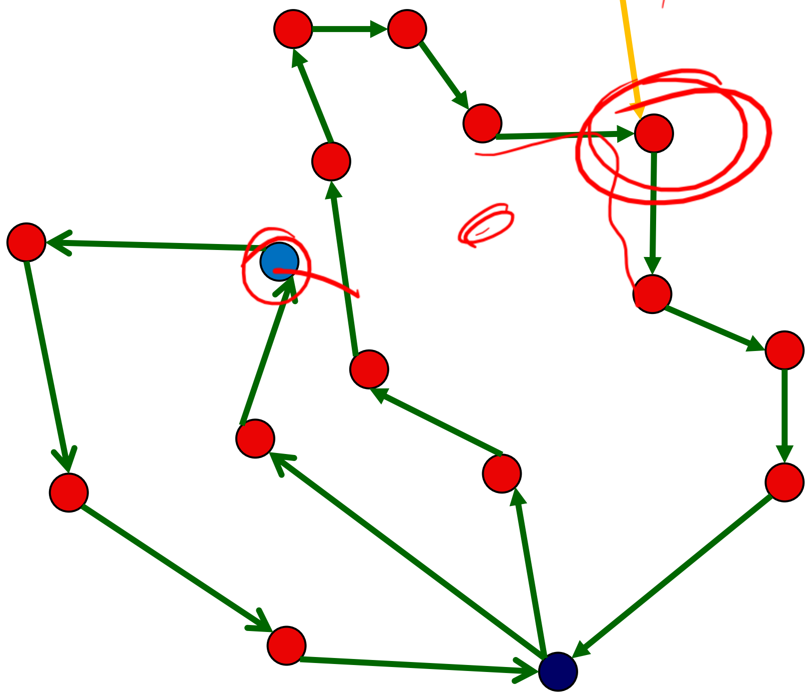
- 1-move



Soft Constraints + Tabu

E.g. VRP

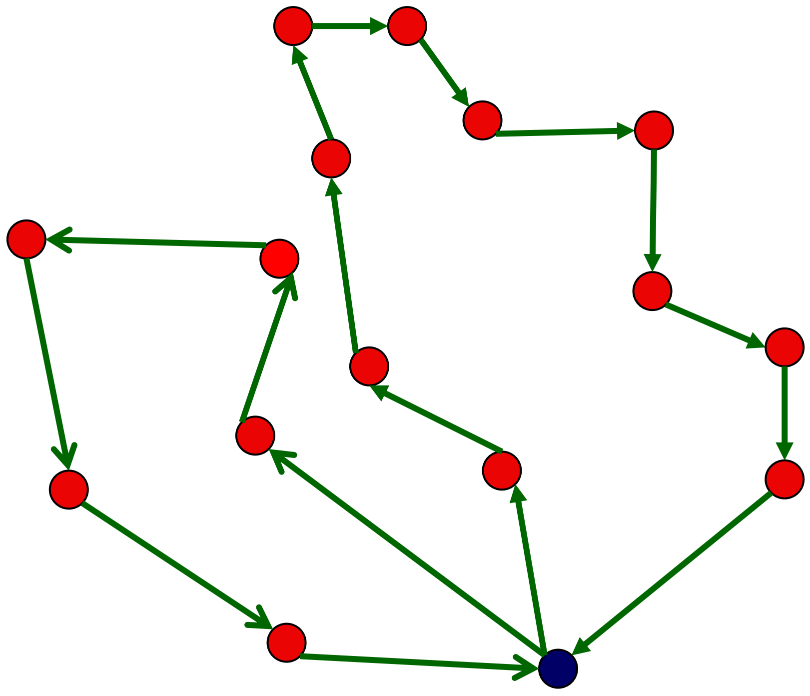
- 1-move



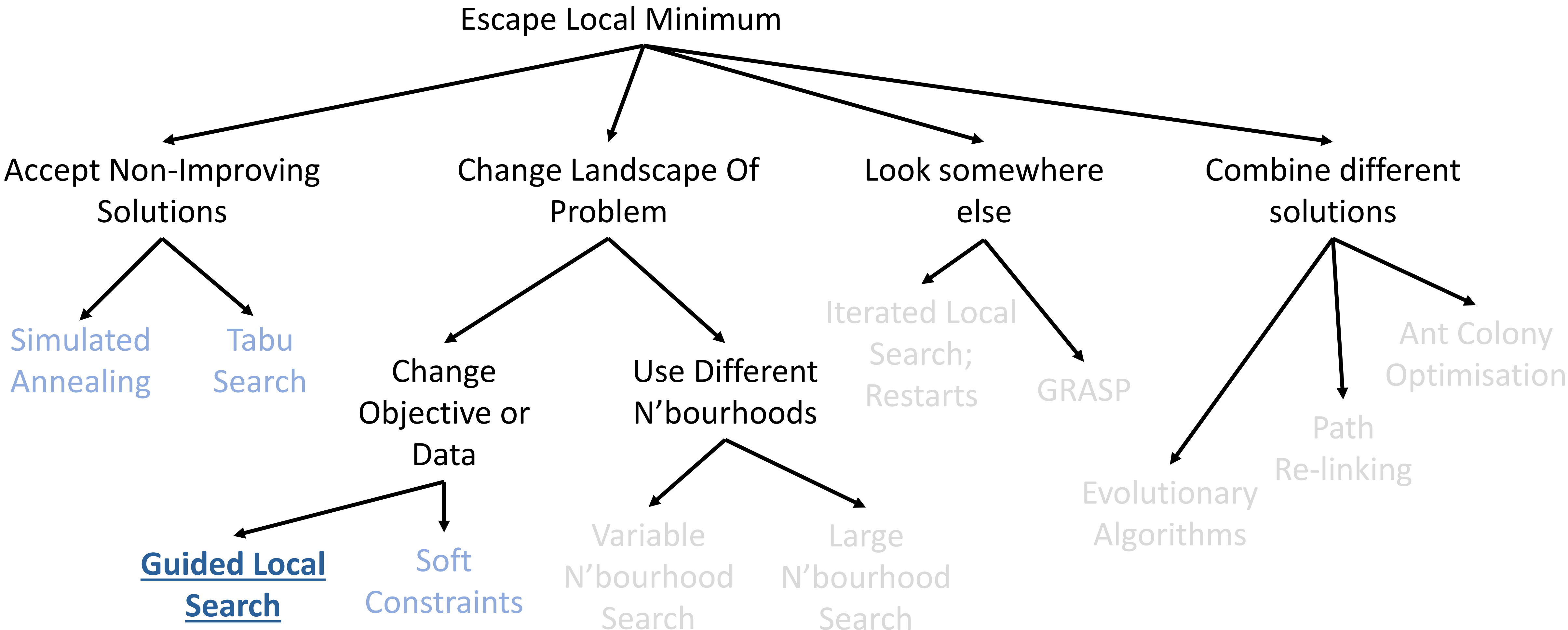
Soft Constraints + Tabu

E.g. VRP

- 1-move



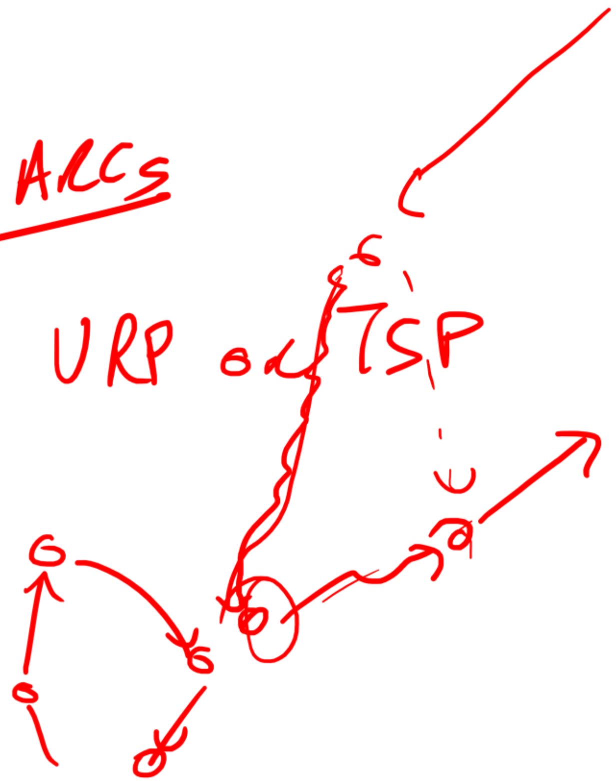
Meta-heuristics: An Incomplete Survey



Guided Local Search

LONG ARCS

FOR VRP or TSP



Basic idea:

- Select a *feature* that indicates a poor solution
- Penalise a solution that exhibits that feature

- Start with zero penalty
- Repeat
  - Perform Local Search to minimise original objective **+ penalties**
  - Select elements to penalise
  - Increase penalty on selected elements

Guided Local Search

Local Search

- Do local search with an *updated objective* $h(s) = \overbrace{g(s)}^{\$\text{ MILLIONS}} + \lambda \cdot \sum_{i=1}^{M} p_i \cdot I_i(s)$

10 ~ 100

- Updated objective
  - h(s): Augmented objective
  - g(s): Original objective
  - λ: "normalisation" $\boxed{\text{parameter}}$
  - p_i: **Count** of times feature *i* has been penalised
  - $I_i(s)$: Indicator function: 1 if feature *i* in solution *s;* 0 otherwise

Guided Local Search

GLS

Select elements to penalise:

- Update penalty of features that maximise Utility
- c_i: Original cost of feature
- $I_i(s)$: Does solution s exhibit feature *i*

MUST BE IN CUR SOL

ARC LENGTH

$$util(s_*, f_i) = \overbrace{I_i(s_*)} \cdot \frac{c_i}{1 + p_i}$$

CUR SOL

- At each iteration
  - Local Search using augmented objective
  - Select maximum utility features
  - Set $p_i = p_i + 1$ for all selected features

$$h(s) = g(s) + \lambda \cdot \sum_{i=1}^{M} p_i \cdot I_i(s)$$

- Penalty increases each iteration
  - Local Search tries harder to eliminate feature
- Utility decreases the more often you penalise a feature
  - Eventually select other features

Guided Local Search – **TSP**

- Feature: an arc (i,j)
- $I_{ij}(s)$ = 1 if arc (i,j) is present; 0 otherwise

$$h(s) = g(s) + \lambda \cdot \sum_{i=1}^{M} p_i \cdot I_i(s)$$

- At each iteration
  - Penalise the longest arc.
  - Try harder to get rid of it

$$util(s_*, f_i) = I_i(s_*) \cdot \frac{c_i}{1 + p_i}$$

- As the search progresses
  - Utility of first arc decreases
  - Other arcs start being penalised

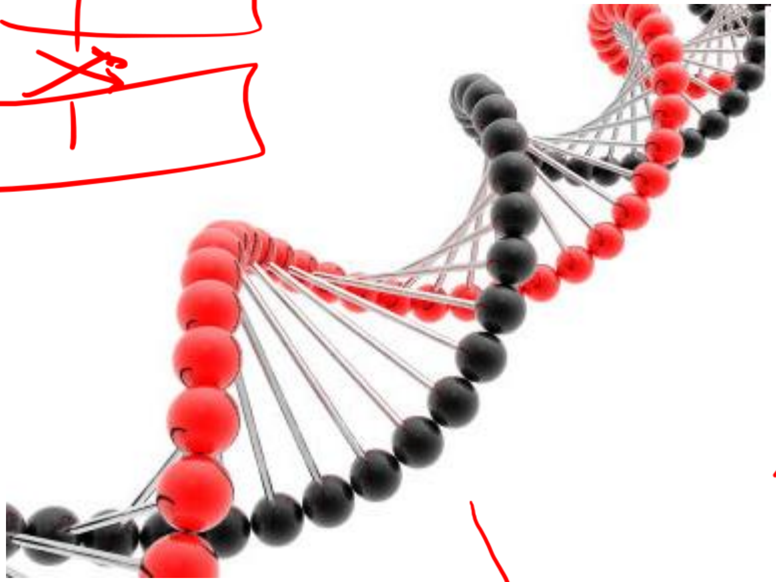
Meta-heuristics: An Incomplete Survey

- Escape Local Minimum
  - Accept Non-Improving Solutions
    - Simulated Annealing
    - Tabu Search
  - Change Landscape Of Problem
    - Change Objective or Data
      - Guided Local Search — NOT SO POPULAR ANYMORE
      - Soft Constraints — VERY POPULAR
    - Use Different N'bourhoods — FRI
      - Variable N'bourhood Search
      - Large N'bourhood Search
  - Look somewhere else — FRI
    - Iterated Local Search; Restarts
    - GRASP
  - Combine different solutions
    - **Evolutionary Algorithms**
    - Path Re-linking
    - Ant Colony Optimisation

Genetic Algorithms

- Generate a population of solutions (construct methods)
- Evaluate fitness (objective)
- Create next generation:
  - Choose two solutions from population
  - Combine the two (two ways)
  - [Mutate]
  - Produce offspring (calculate fitness)
  - [Improve] LOCAL SEARCH ON OFFS.
  - Repeat until population doubles
- Apply selection:
  - Bottom half "dies"
- Repeat

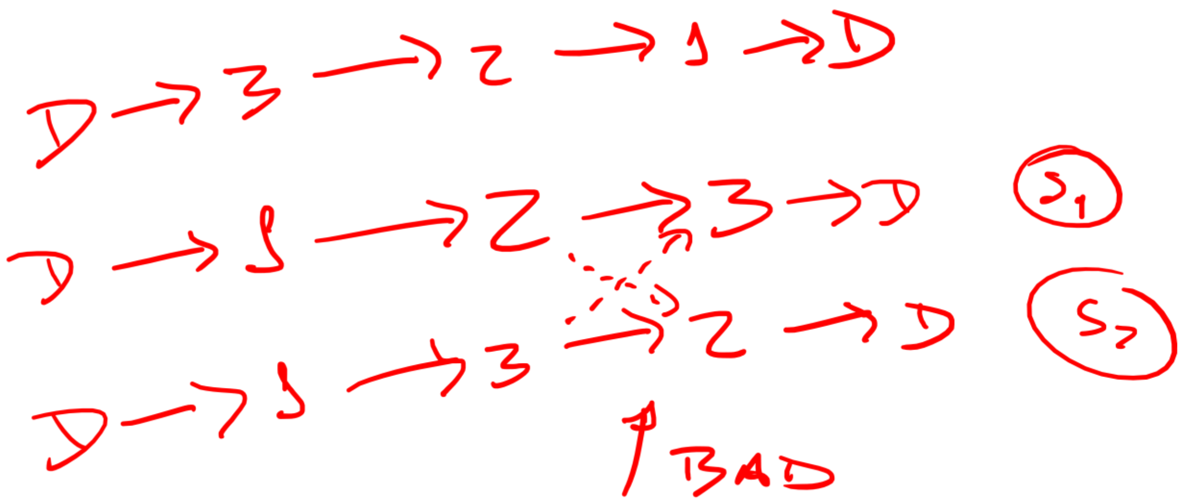
Turns it into a *Memetic Algorithm*



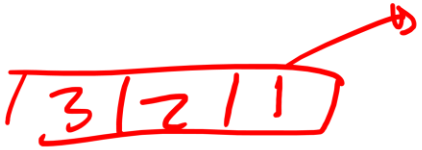
Genetic Algorithms

Solution Representation is key

- Needs to fulfil **multiple goals**
  - Easy to calculate fitness (objective)
  - Easy to perform crossover (merge)
  - Easy to manipulate (mutation)
  - Easy for local search



- E.g. VRP



  - First attempts used array for each route, or successor info
  - Very difficult for crossover
  - Better rep turns out to be a single array

Genetic Algorithms

E.g. VRP

- "Split" method
- Introduced by Prins (2004)

- Solution represented as a "Grand Tour" (ordering of all customers)
- Split algorithm divides the tour into feasible routes
  - Uses Dynamic Programming

3 TRUCKS

6	1	9	7	3	2	10	4	8	5

4 – GRAND TOUR

Genetic Algorithms

E.g. VRP

- "Split" method
- Introduced by Prins (2004)

- Solution represented as a "Grand Tour" (ordering of all customers)
- Split algorithm divides the tour into feasible routes
    - Uses Dynamic Programming

6	1	9	7	3	2	10	4	8	5

T_1 T_2 T_3

Genetic Algorithms

E.g. VRP

- "Split" method
- Introduced by Prins (2004)

- Solution represented as a "Grand Tour" (ordering of all customers)
- Split algorithm divides the tour into feasible routes
  - Uses Dynamic Programming

6	1	9	7	3	2	10	4	8	5

Genetic Algorithms

E.g. VRP

- “Split” method
- Introduced by Prins (2004)

- Solution represented as a “Grand Tour” (ordering of all customers)
- Split algorithm divides the tour into feasible routes
  - Uses Dynamic Programming

6	1	9	7	3	2	10	4	8	5

Genetic Algorithms

E.g. VRP

- "Split" method
- Introduced by Prins (2004)

- Solution represented as a "Grand Tour" (ordering of all customers)
- Split algorithm divides the tour into feasible routes
  - Uses Dynamic Programming

6	1	9	7	3	2	10	4	8	5

Genetic Algorithms

E.g. VRP

- "Split" method
- Introduced by Prins (2004)

- Solution represented as a "Grand Tour" (ordering of all customers)
- Split algorithm divides the tour into feasible routes
  - Uses Dynamic Programming

6	1	9	7	3	2	10	4	8	5

Genetic Algorithms

E.g. VRP

- "Split" method
- Introduced by Prins (2004)

- Solution represented as a "Grand Tour" (ordering of all customers)
- Split algorithm divides the tour into feasible routes
  - Uses Dynamic Programming

6	1	9	7	3	2	10	4	8	5

Genetic Algorithms

E.g. VRP

- “Split” method
- Introduced by Prins (2004)

- Solution represented as a “Grand Tour” (ordering of all customers)
- Split algorithm divides the tour into feasible routes
  - Uses Dynamic Programming

6	1	9	7	3	2	10	4	8	5

Genetic Algorithms

E.g. VRP

- "Split" method
- Introduced by Prins (2004)

- Solution represented as a "Grand Tour" (ordering of all customers)
- Split algorithm divides the tour into feasible routes
  - Uses Dynamic Programming

6	1	9	7	3	2	10	4	8	5

T_1 T_2 T_3

125

OBJ F.

Genetic Algorithms

2 TRUCKS

T1 T2

1	4	6	2	3	8	9	5	10	7	125
1	10	8	2	5	4	9	3	6	7	132
6	1	9	7	3	2	10	4	8	5	160
1	9	7	10	8	6	4	3	2	5	198

Genetic Algorithms

1	4	6	~~2~~	3	~~8~~	9	5	10	7	125
1	10	8	2	5	4	9	3	6	7	132
6	1	9	7	3	2	10	4	8	5	160
1	9	7	10	8	6	4	3	2	5	198
1	10	8	2	3	9	5	7	6	4	128

Genetic Algorithms

1	4	6	2	3	8	9	5	10	7	125	S_1
1	10	8	2	5	4	9	3	6	7	132	S_2
6	1	9	7	3	2	10	4	8	5	160	
1	9	7	10	8	6	4	3	2	5	198	
1	10	8	2	3	9	5	7	6	4	128	
1	4	6	5	9	3	7	10	8	2	206	

2 OFFS. FROM S_1 & S_2

Genetic Algorithms

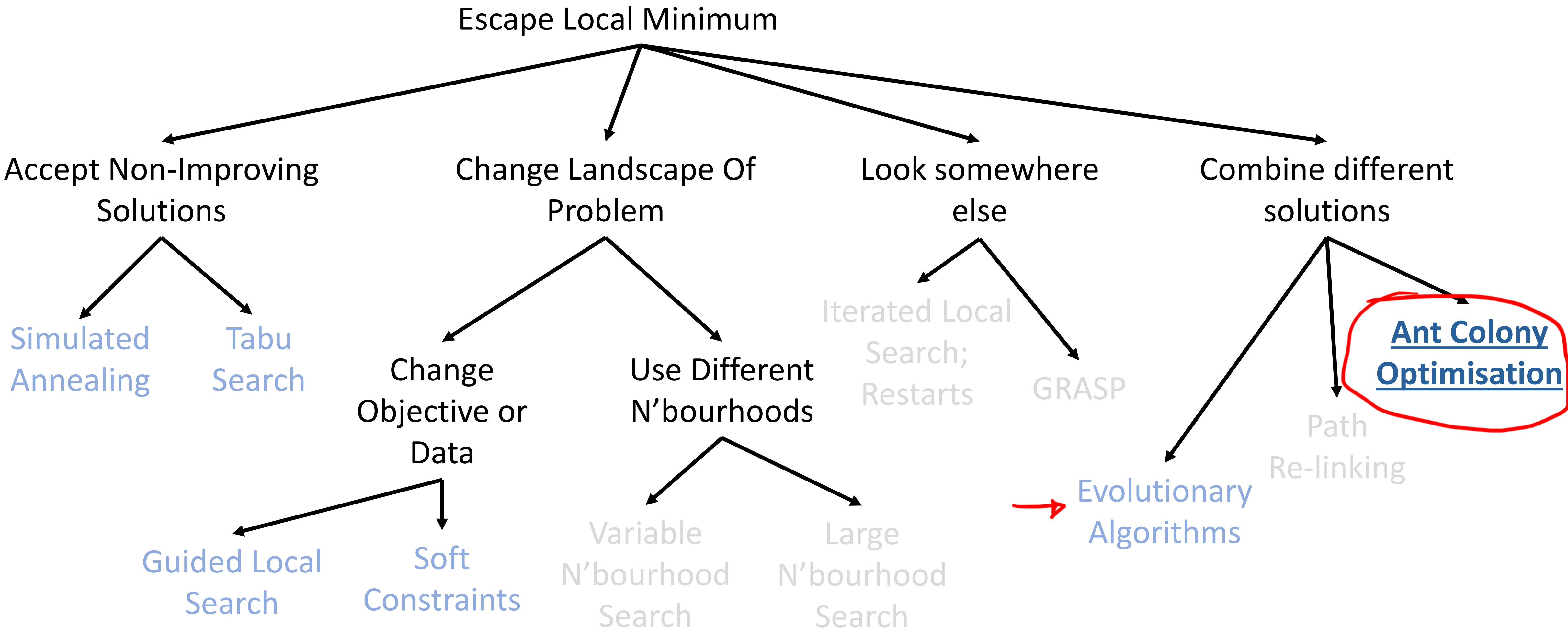
Diversification:

- Big problem is getting a homogenous population
- Too much intensification, not enough diversification

EXPLORATION

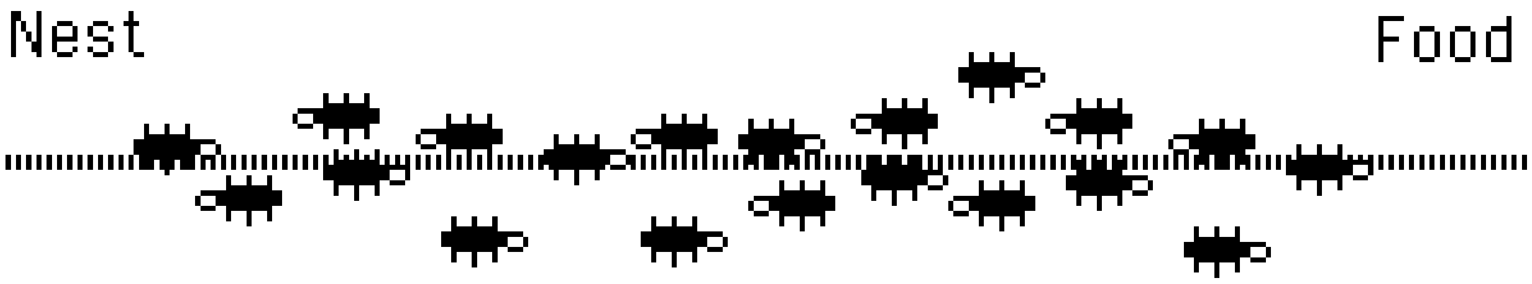
- Some algorithms explicitly measure diversity
  - keep lower-quality solutions that maintain diversity
- Meta-meta: $\boxed{\text{Soft constraints}}$
  - Maintain a separate population of infeasible solutions

Meta-heuristics: An Incomplete Survey



Ant Colony Optimisation (ACO)

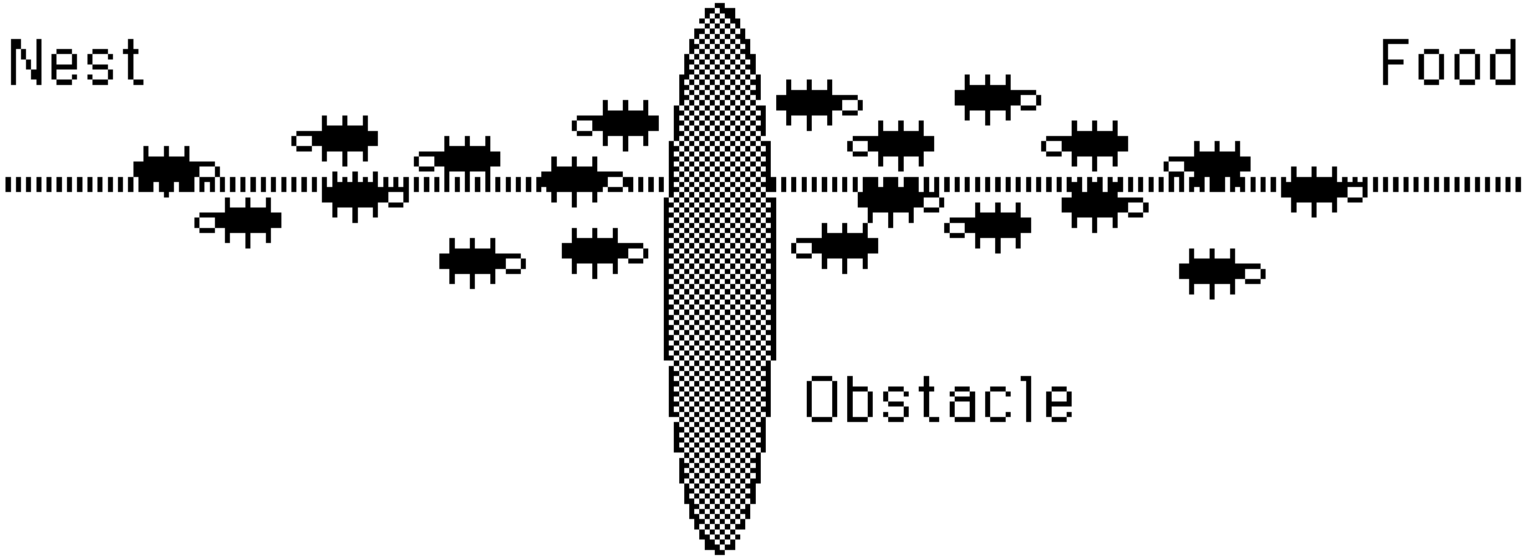
- By analogy to foraging behaviour of Ants



Pictures from in Marco Dorigo's Ph.D. thesis

Ant Colony Optimisation (ACO)

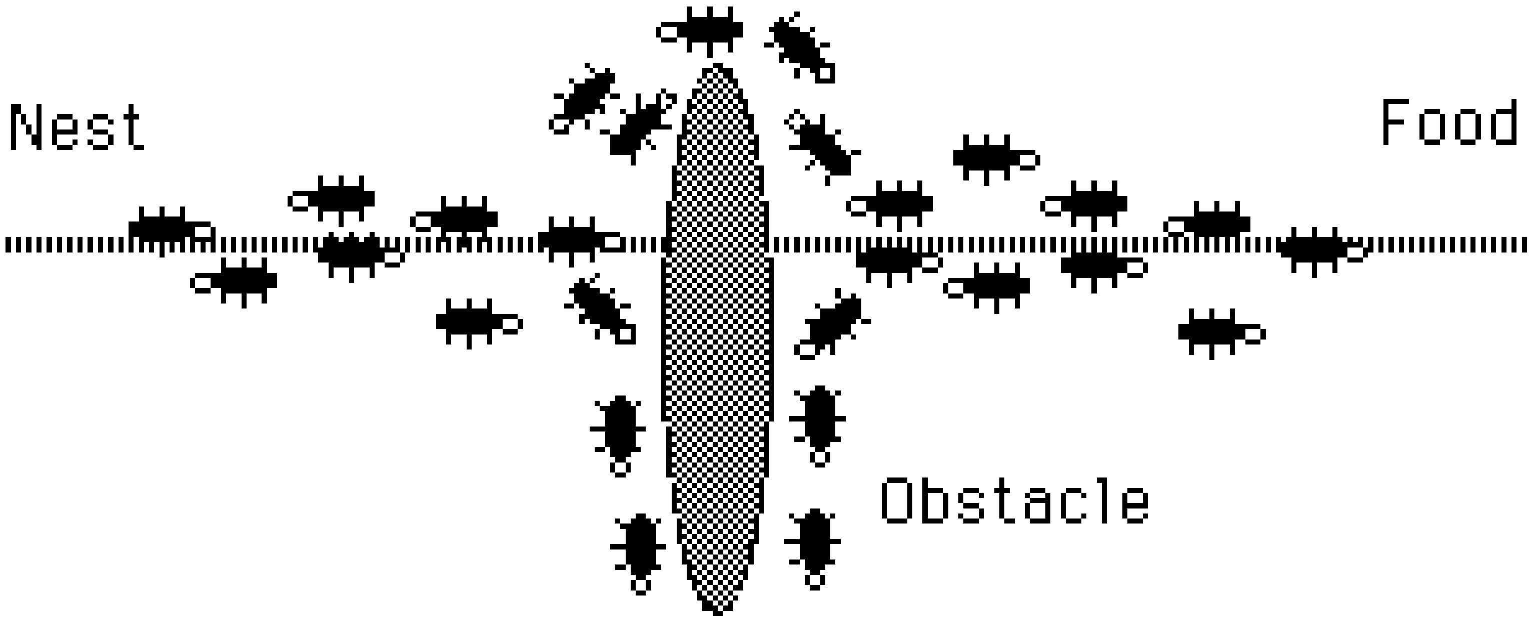
- An obstacle appears!



Pictures from in Marco Dorigo's Ph.D. thesis

Ant Colony Optimisation (ACO)

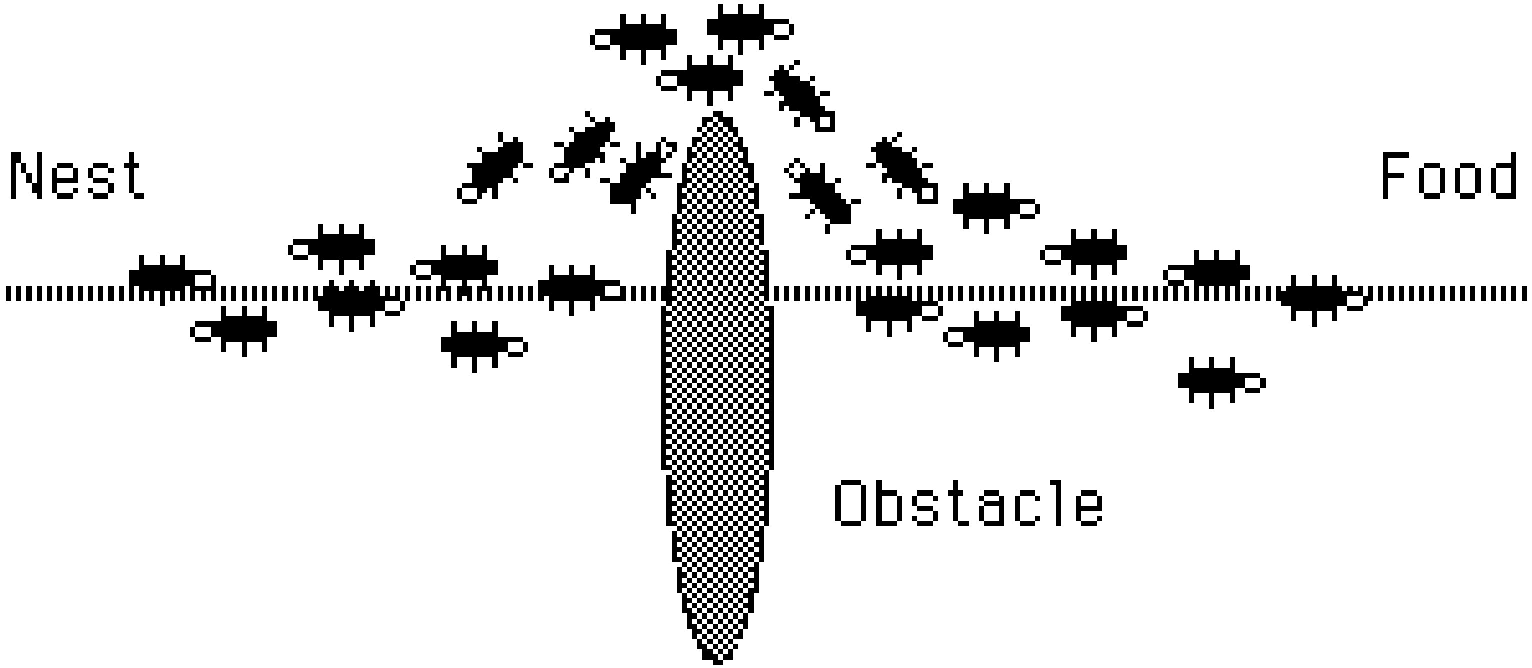
- Everybody flip a coin



Pictures from in Marco Dorigo's Ph.D. thesis

Ant Colony Optimisation (ACO)

- Shortest path is reinforced



Pictures from in Marco Dorigo's Ph.D. thesis

Ant Colony Optimisation (ACO)

TCP PACKETS IN NET
CELL COM.

E.g.: Want to find shortest paths in a communications network

PACKETS

- Send out ants that choose a path
  - Partly at random (our naïve heuristic)
  - Partly influenced by previous ants: **Pheromone trail** – our meta-heuristic
- The first ant to get to a destination increases the Pheromone on its path
- Pheromone levels decrease over time
  - More ants select the best path
  - The best path gets reinforced

Ant Colony Optimisation (ACO)

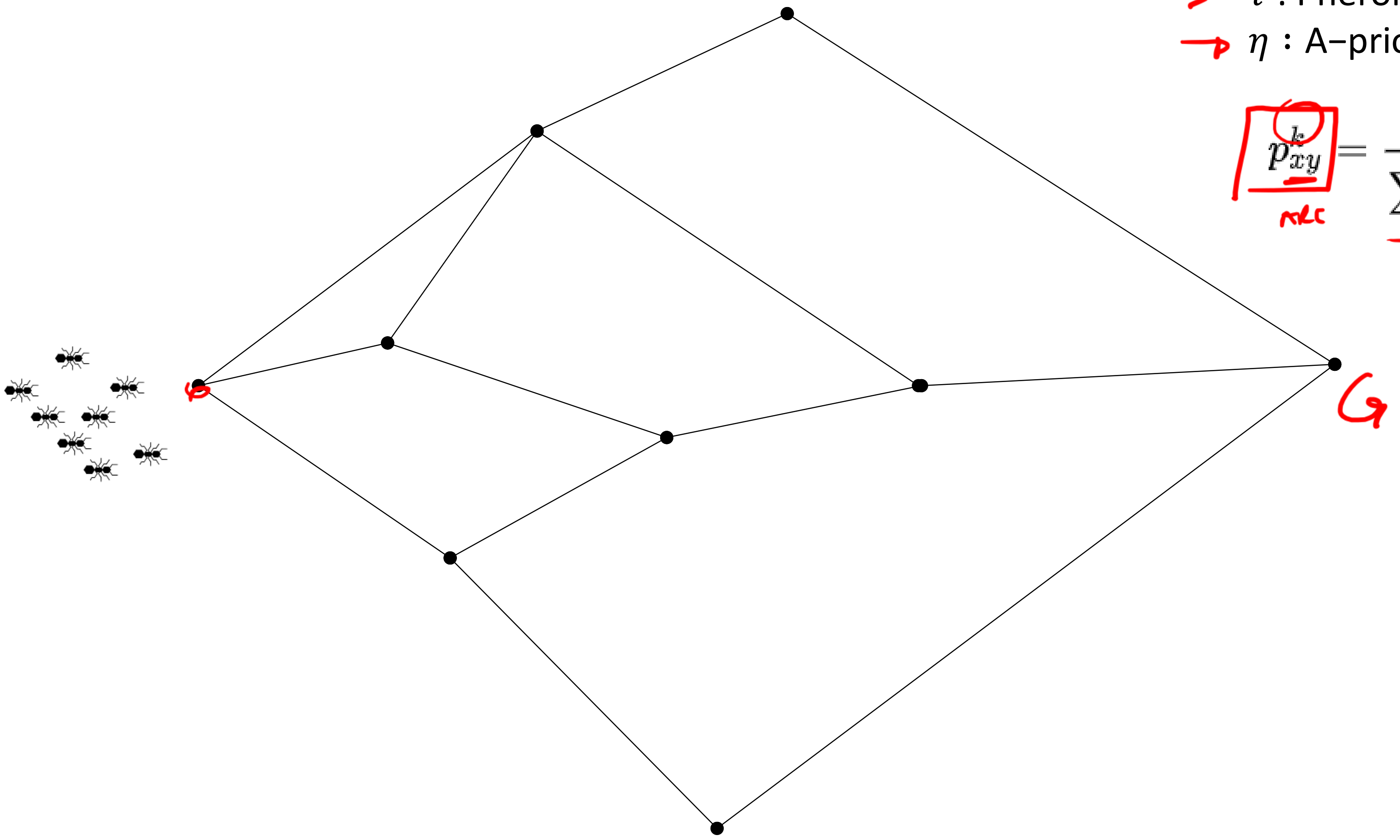
Advantages:

- (Relatively) Simple
- Distributed
- Easy to parallelise
- Robust:
  - If the network changes, new pheromones will be deposited and a new path found

Disadvantages:

- Sensitive to parameter settings
  - Lay down too much pheromone → Lock in poor solution early (poor diversity)
  - Lay down too little → Slow to converge (poor intensification)
  - Decay pheromone too slowly → Bad paths persist
  - Decay too quickly → Best path is forgotten

Ant Colony Optimisation (ACO) – TSP



Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A–priori "attractiveness"

$$\boxed{p^{k}_{xy}} = \frac{(\tau^{\alpha}_{xy})(\eta^{\beta}_{xy})}{\sum_{z \in \mathrm{allowed}_x} (\tau^{\alpha}_{xz})(\eta^{\beta}_{xz})}$$

arc

Ant Colony Optimisation (ACO) – TSP

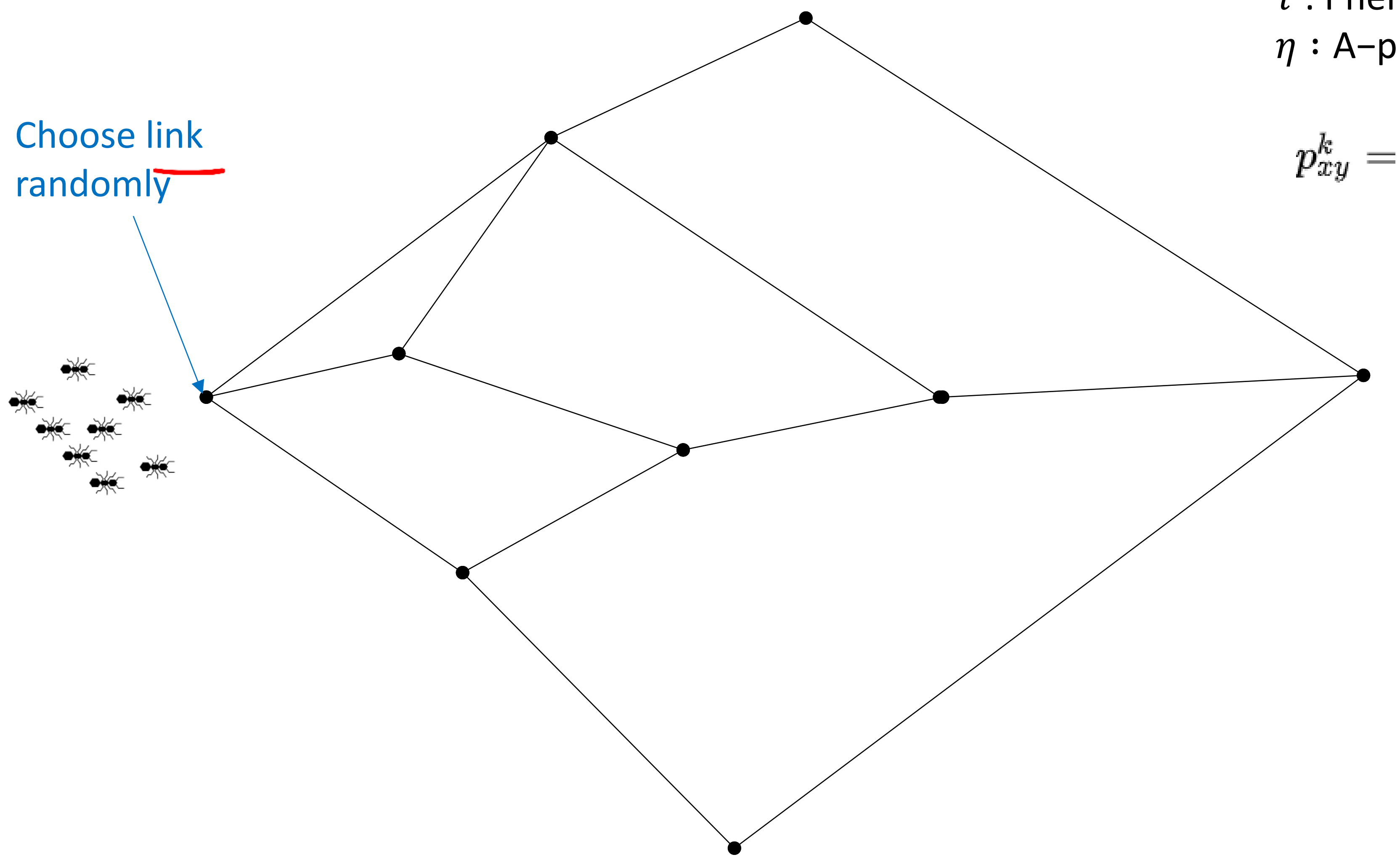
Probability that ant chooses arc *xy*

τ : Pheromone strength

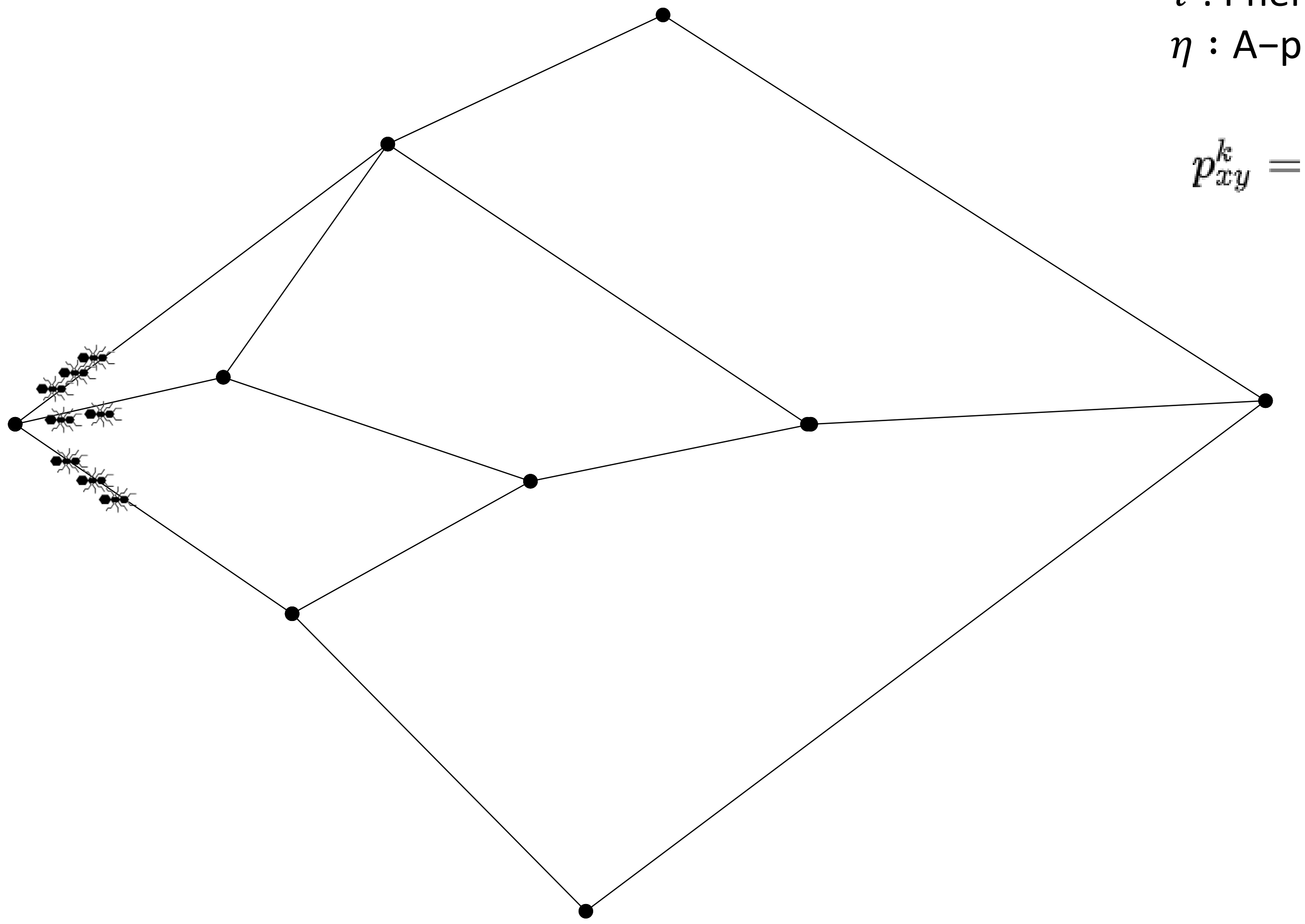
η : A–priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Choose link randomly



Ant Colony Optimisation (ACO) – TSP



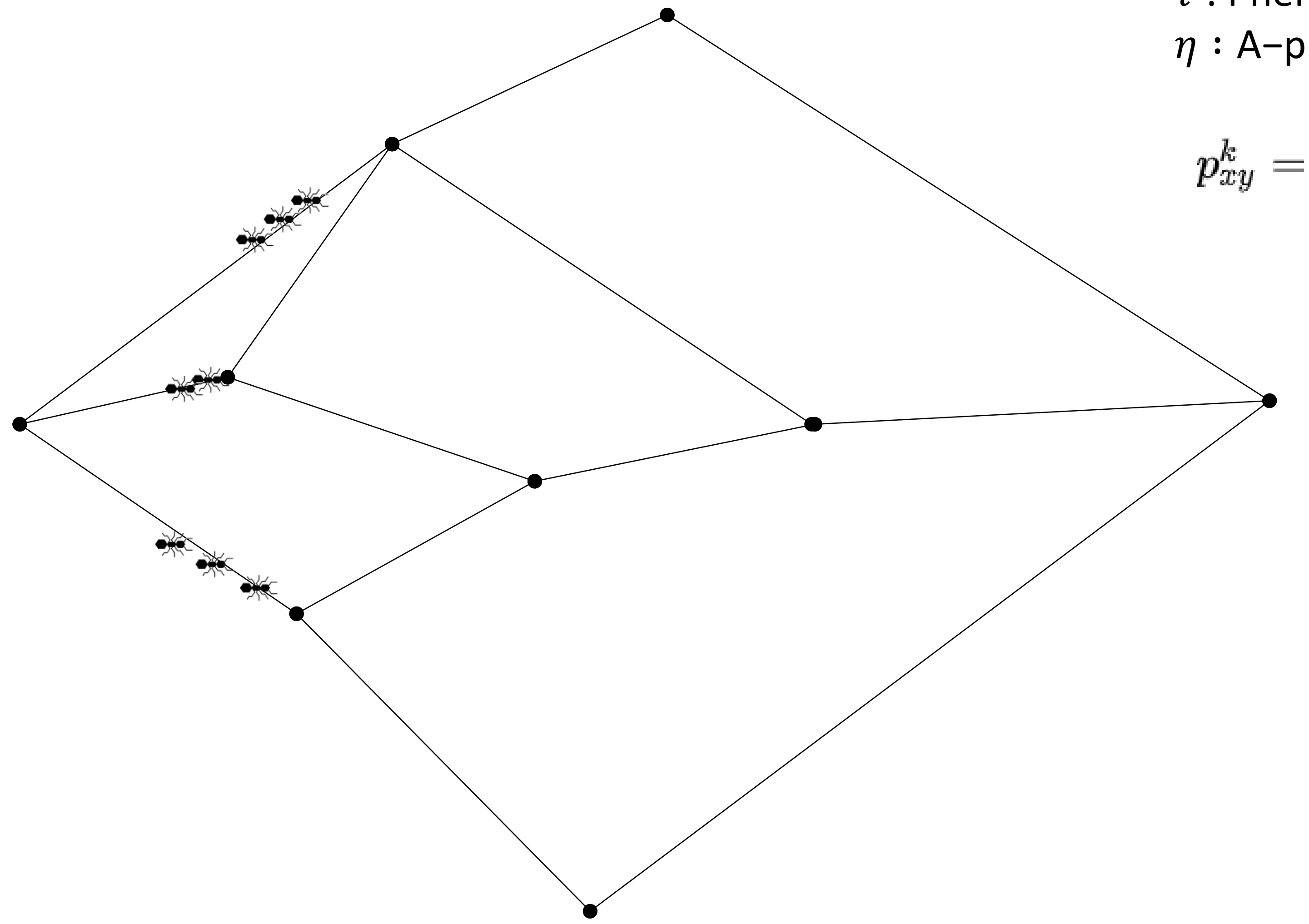
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A−priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \mathrm{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Ant Colony Optimisation (ACO) – TSP



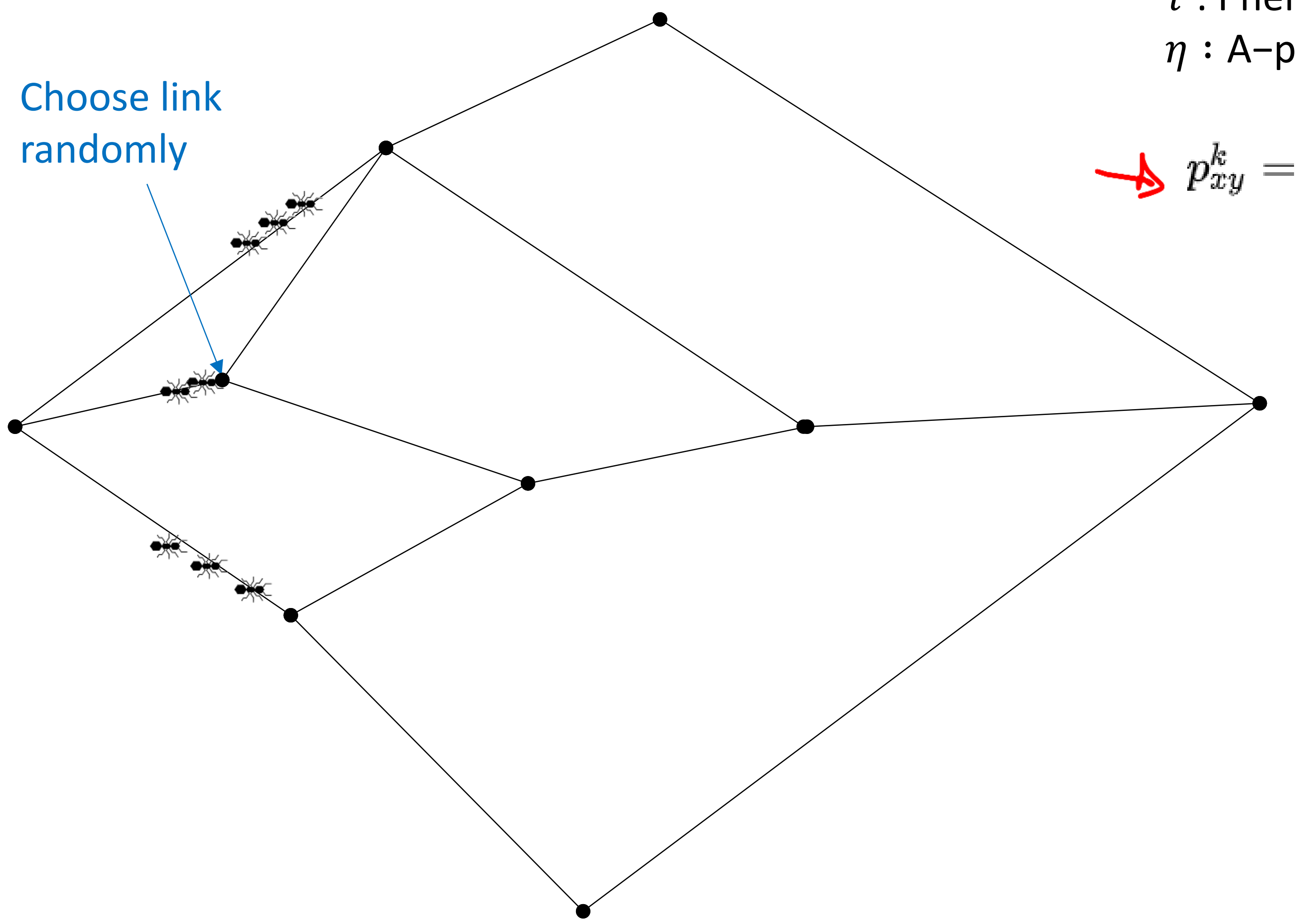
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A−priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \mathrm{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Ant Colony Optimisation (ACO) – TSP



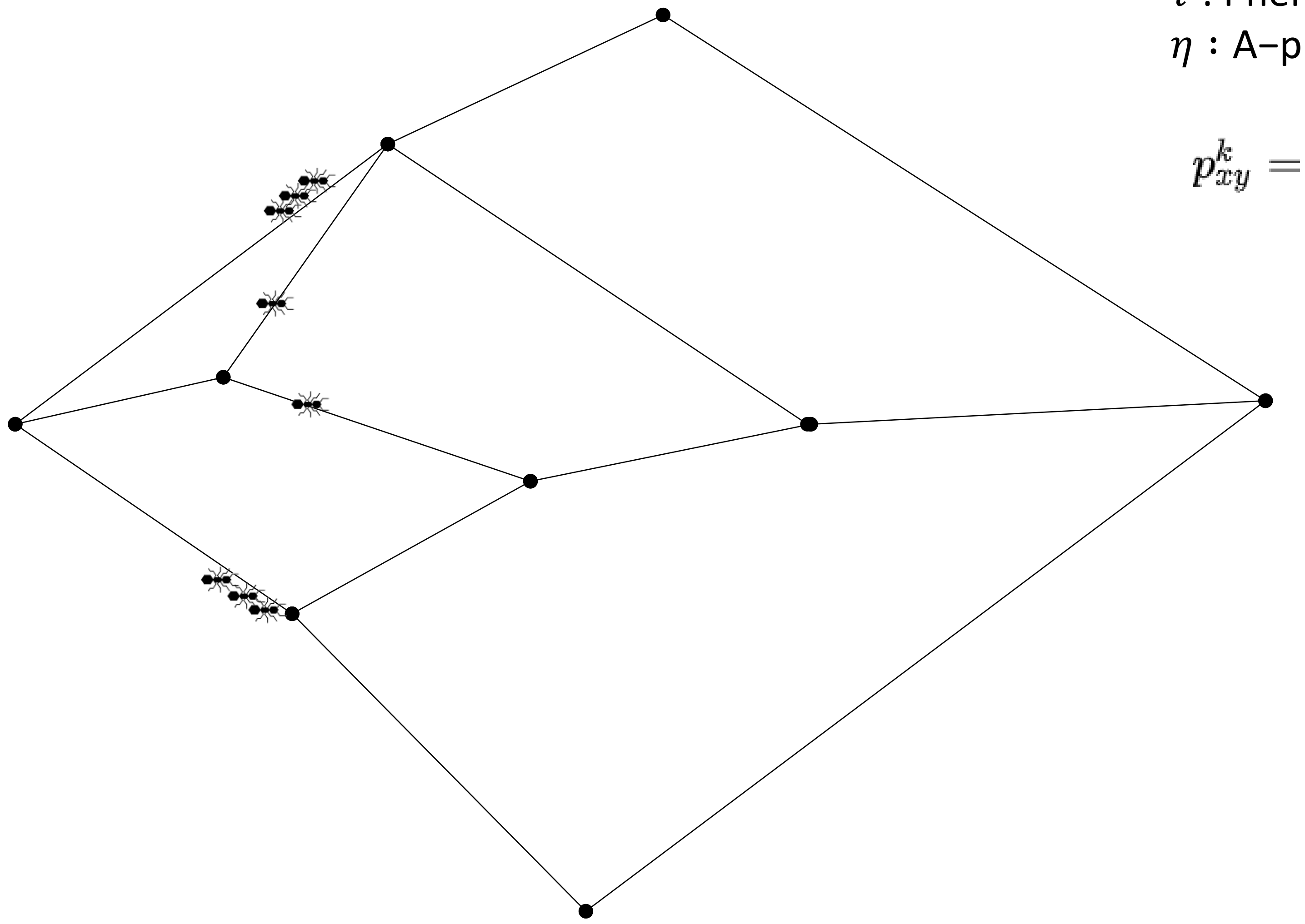
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A–priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Ant Colony Optimisation (ACO) – TSP



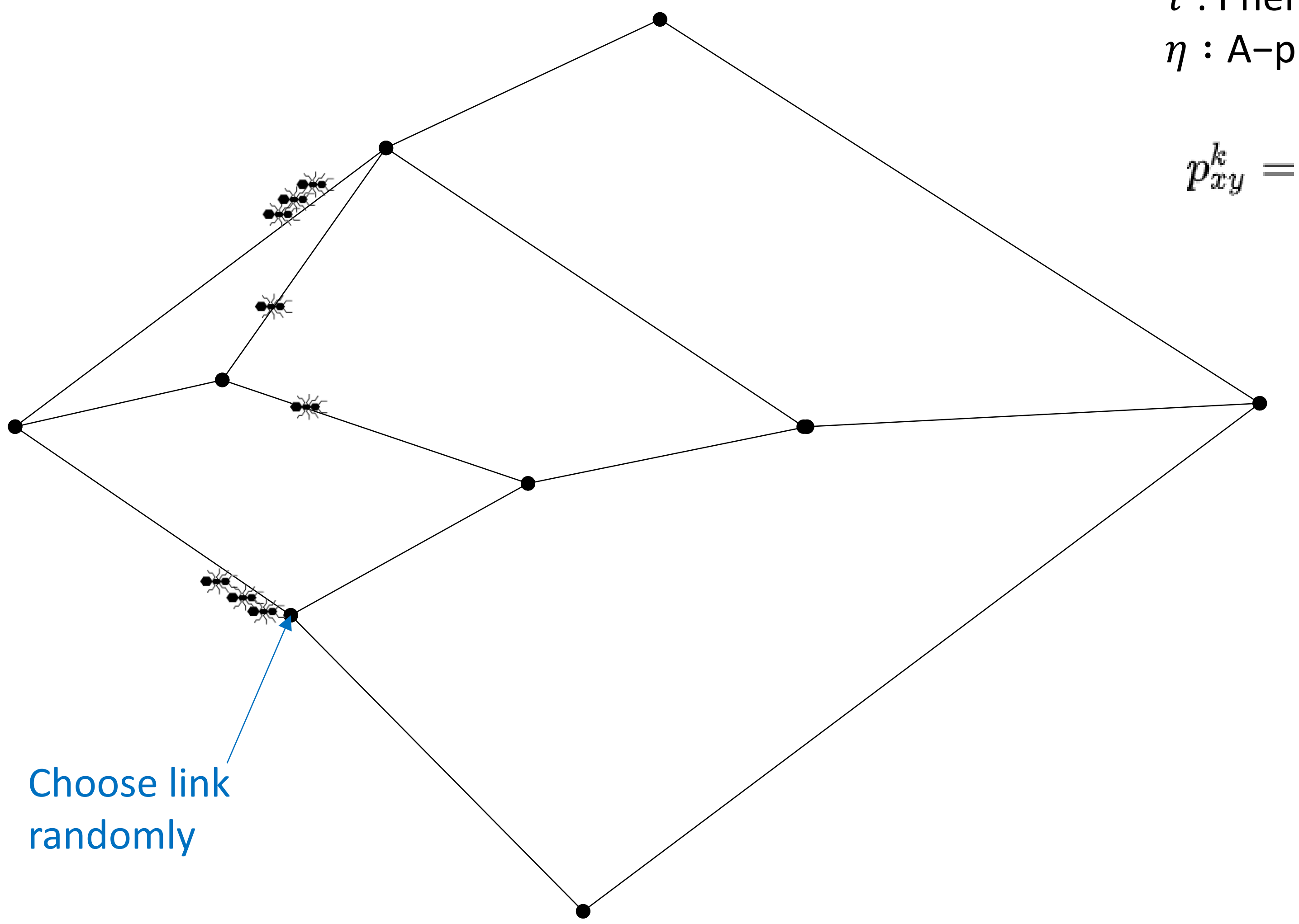
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A−priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Ant Colony Optimisation (ACO) – TSP



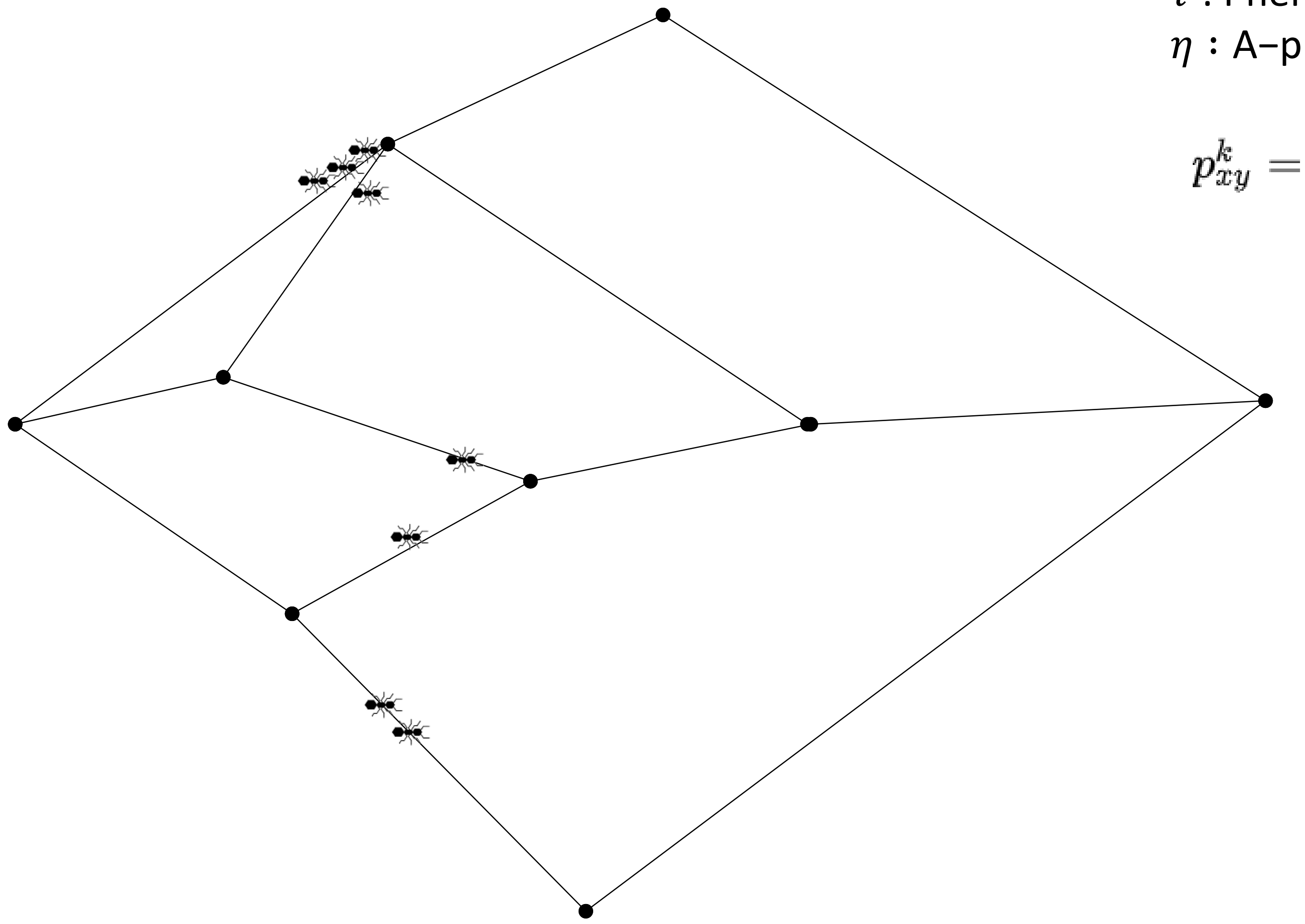
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A−priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \mathrm{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Ant Colony Optimisation (ACO) – TSP



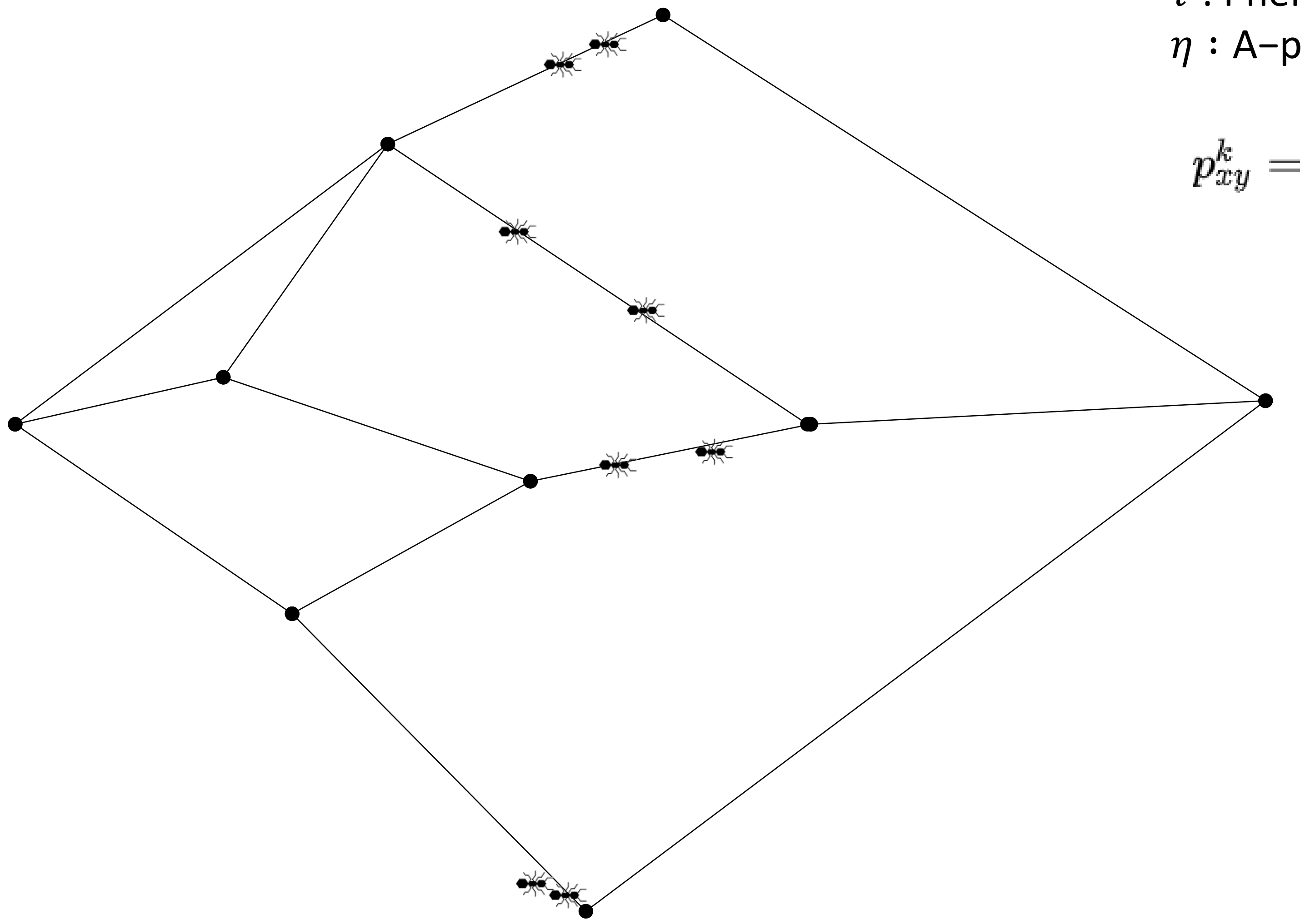
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A−priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

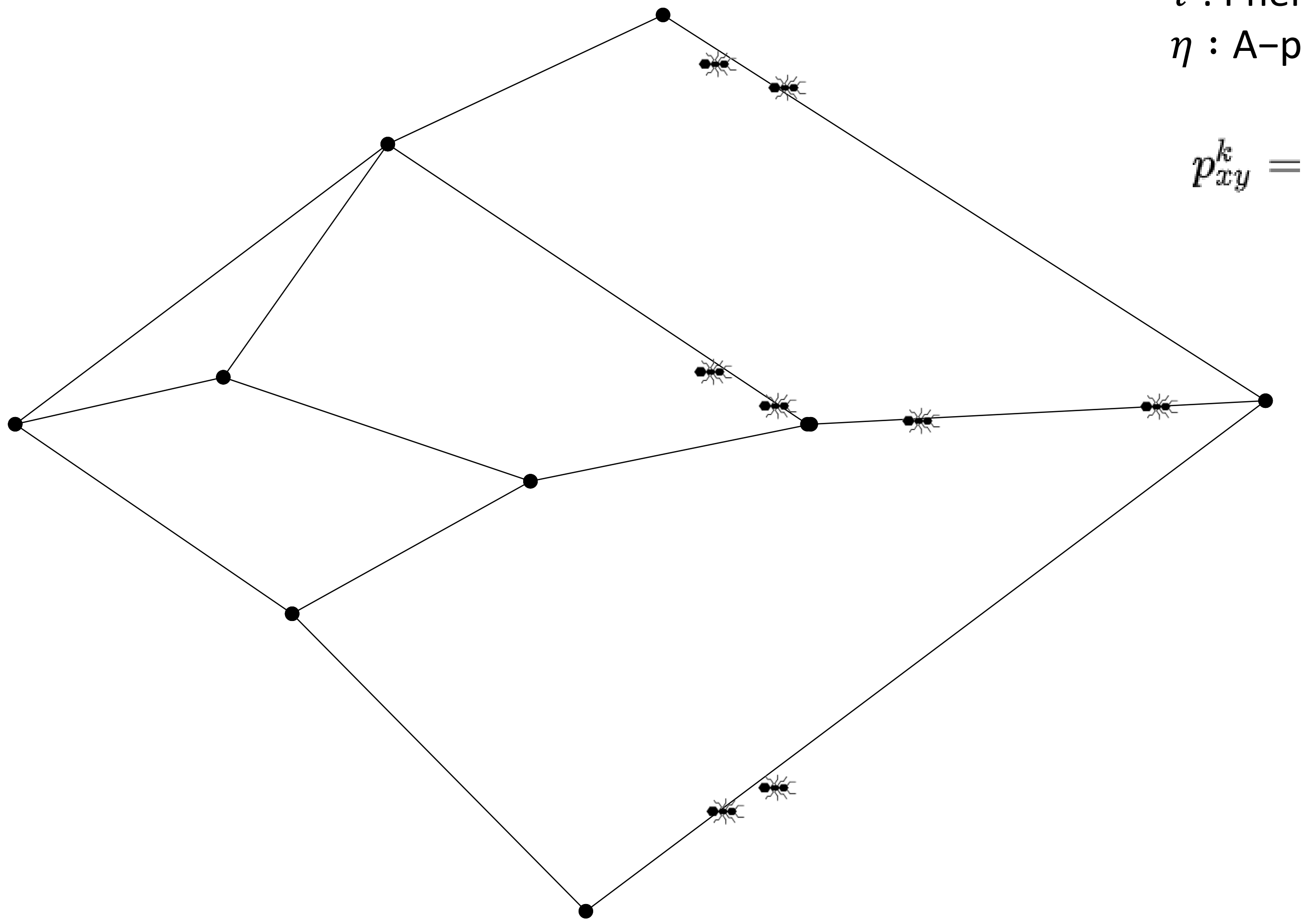
Ant Colony Optimisation (ACO) – TSP



Probability that ant chooses arc *xy*
τ : Pheromone strength
η : A−priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Ant Colony Optimisation (ACO) – TSP



Probability that ant chooses arc *xy*
τ : Pheromone strength
η : A−priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

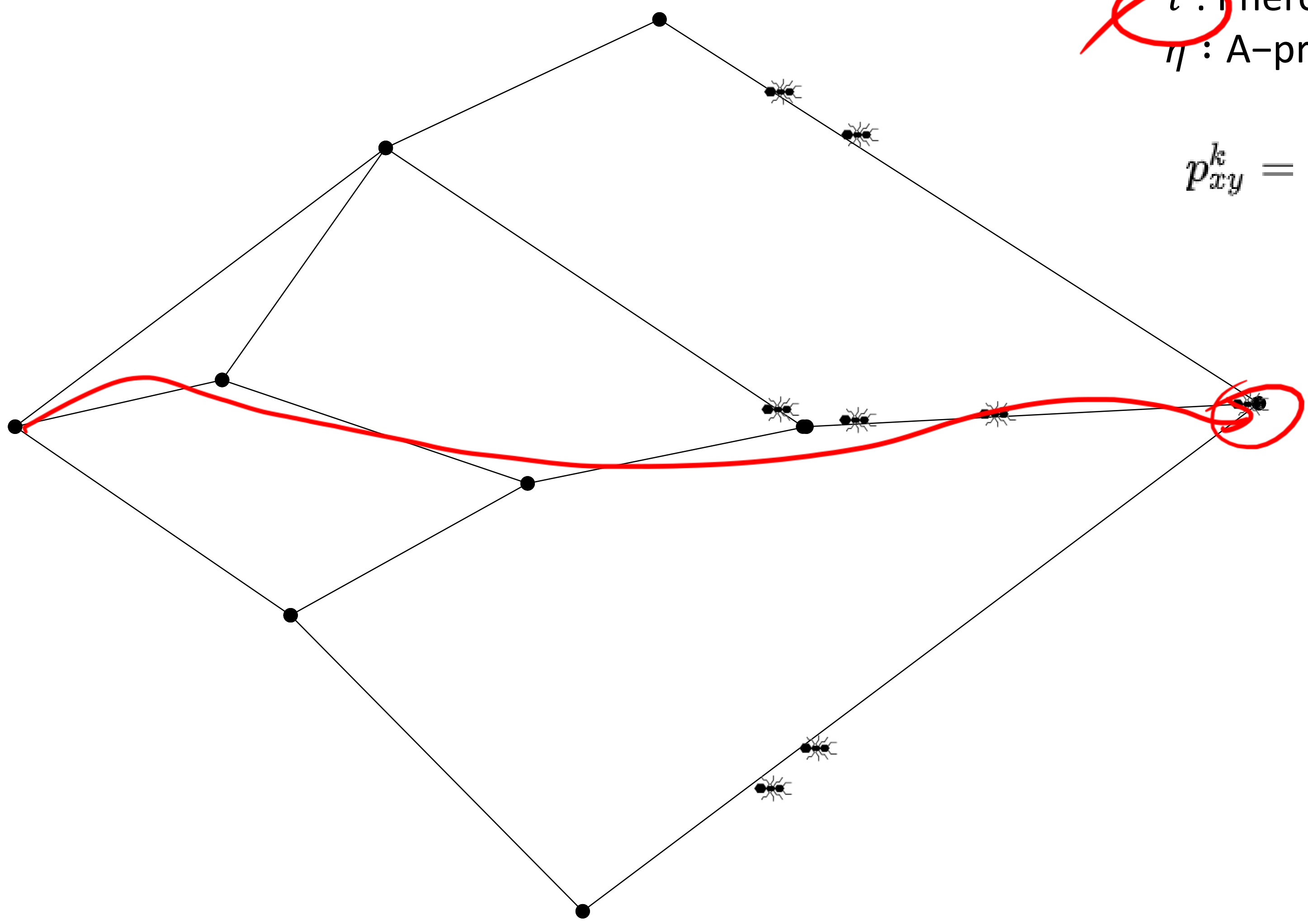
Ant Colony Optimisation (ACO) – TSP

Probability that ant chooses arc *xy*

τ : Pheromone strength

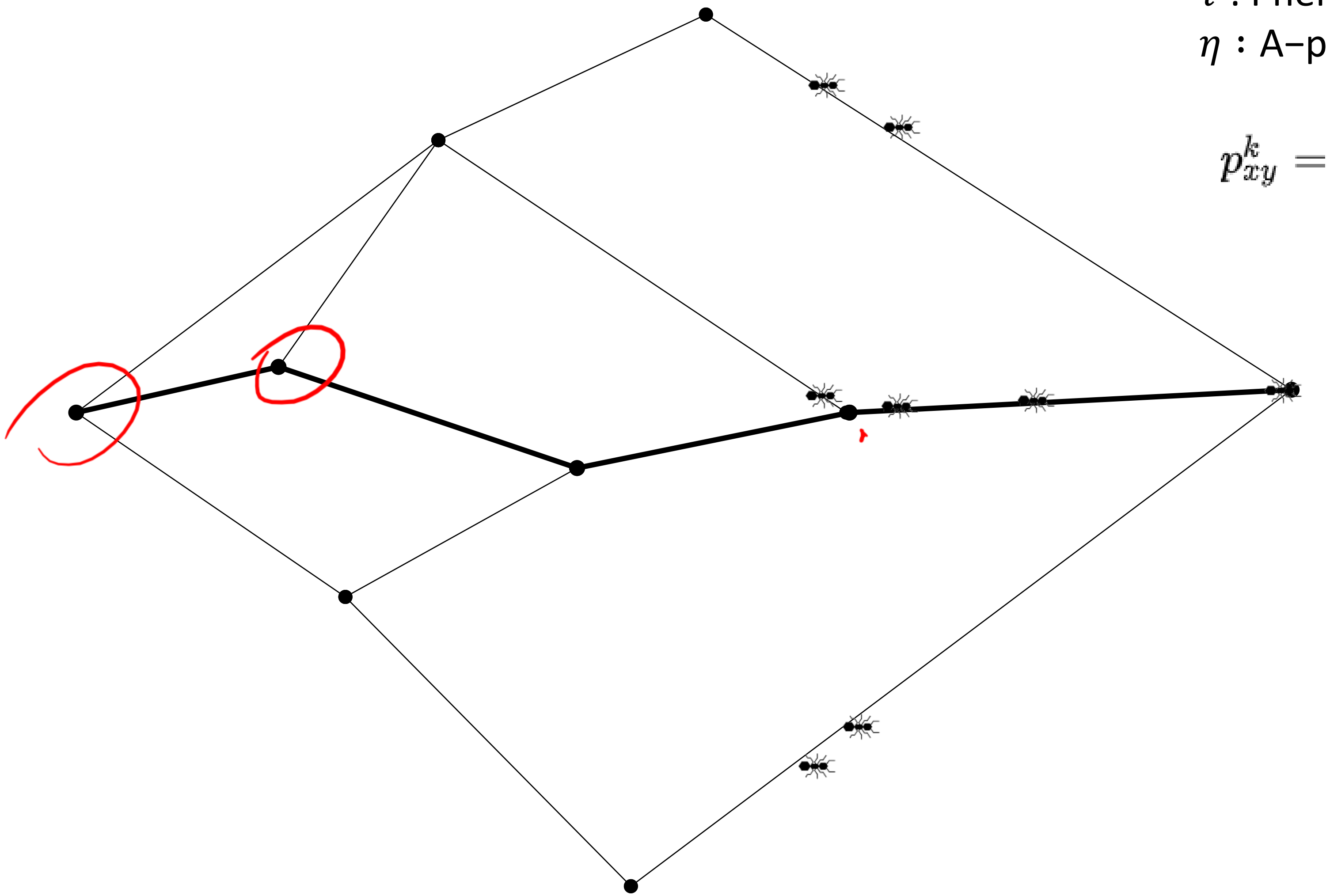
η : A–priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$



First to arrive

Ant Colony Optimisation (ACO) – TSP



Probability that ant chooses arc *xy*

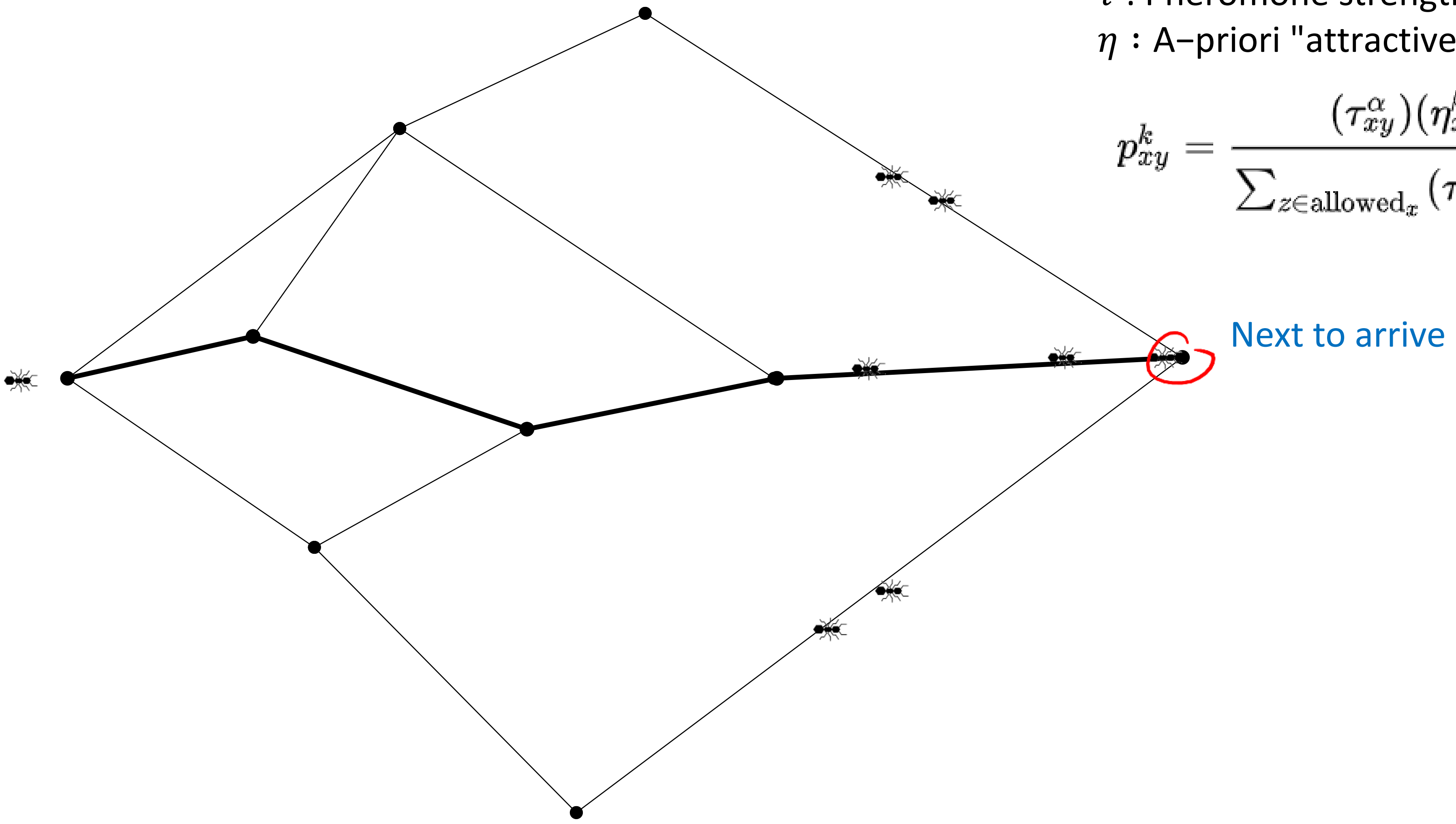
τ : Pheromone strength

η : A–priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \mathrm{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Update pheromones

Ant Colony Optimisation (ACO) – TSP



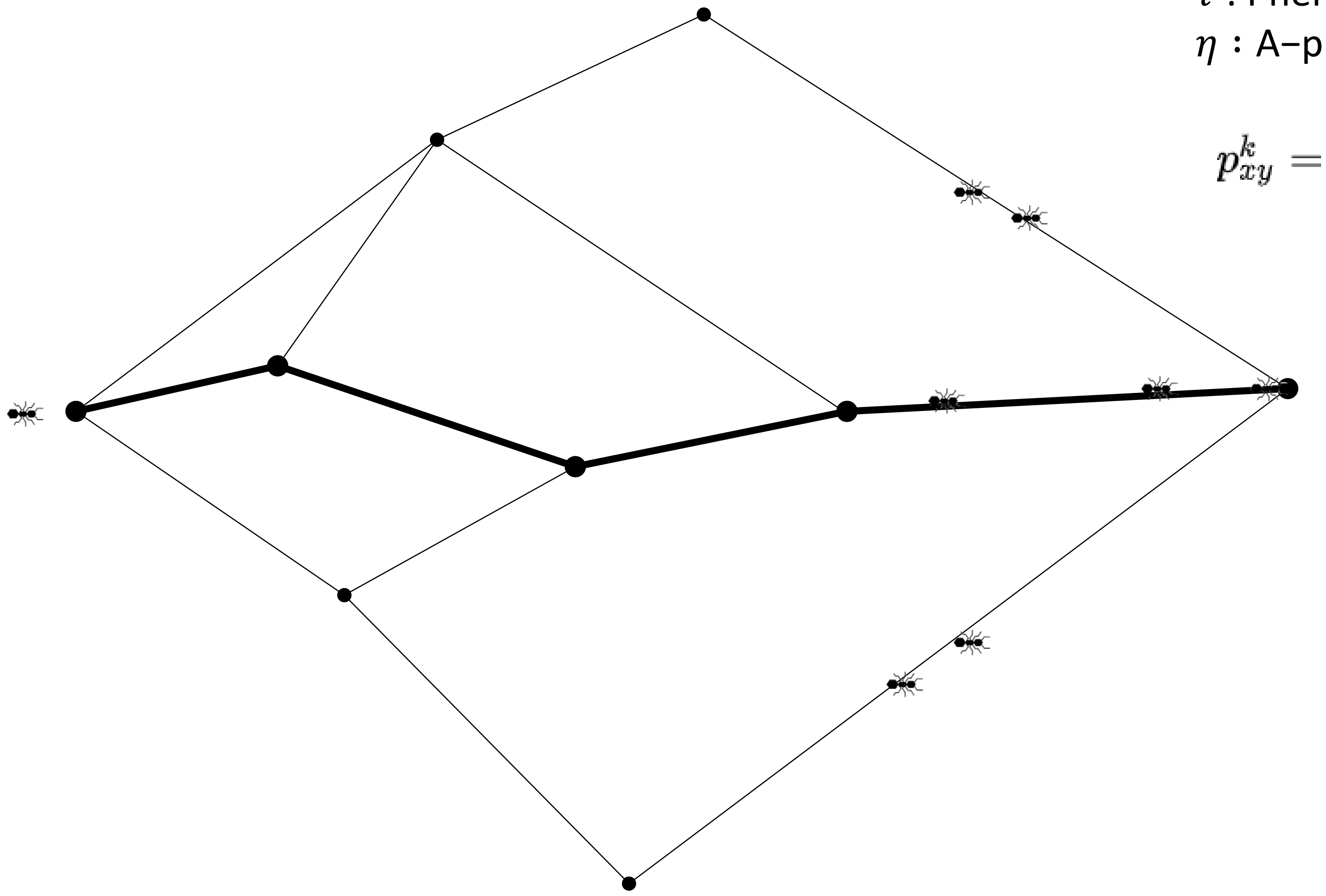
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A–priori "attractiveness"

$$p_{xy}^k = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Ant Colony Optimisation (ACO) – TSP



Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A–priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \mathrm{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Update pheromones

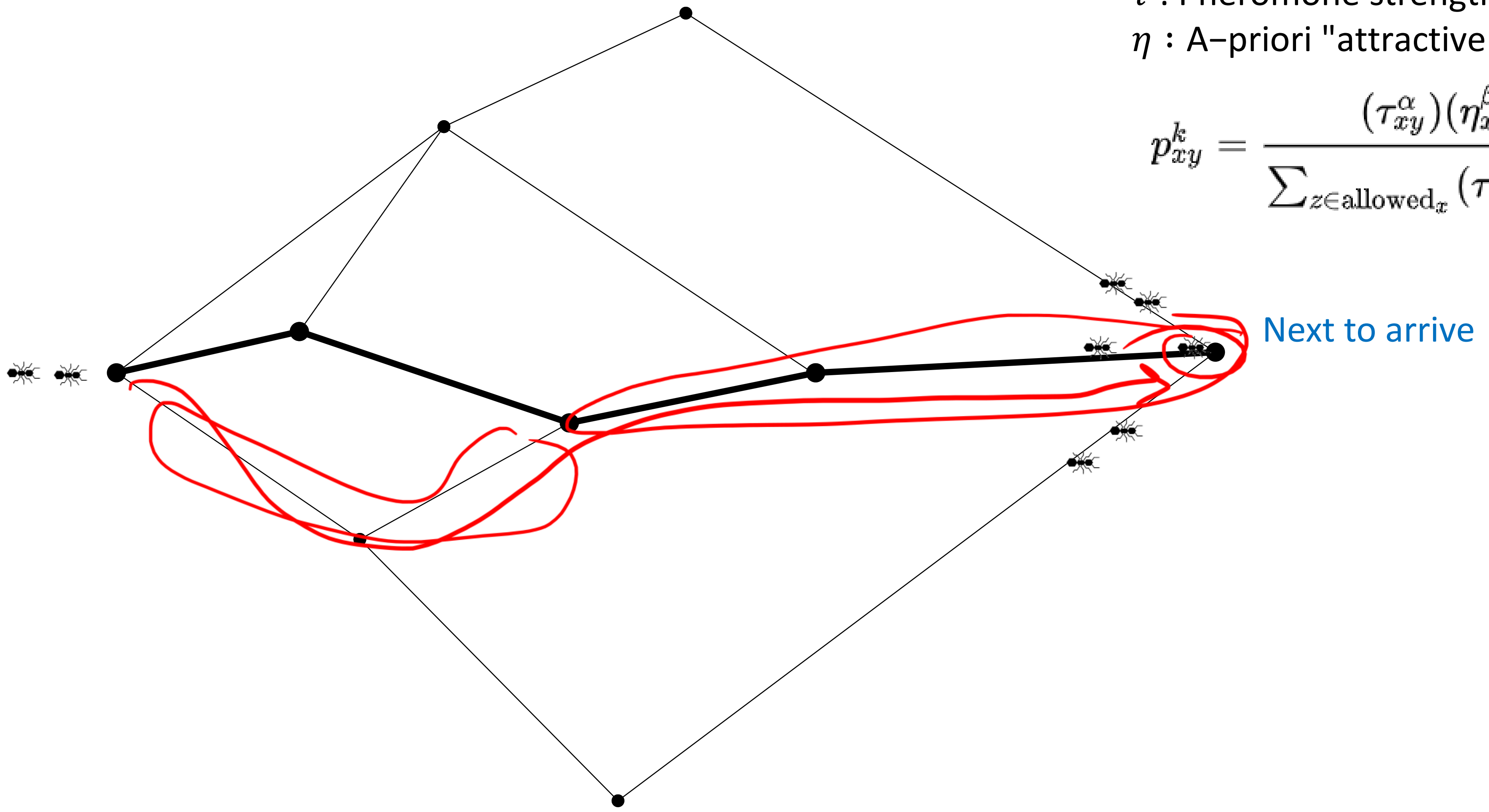
Ant Colony Optimisation (ACO) – TSP

Probability that ant chooses arc *xy*

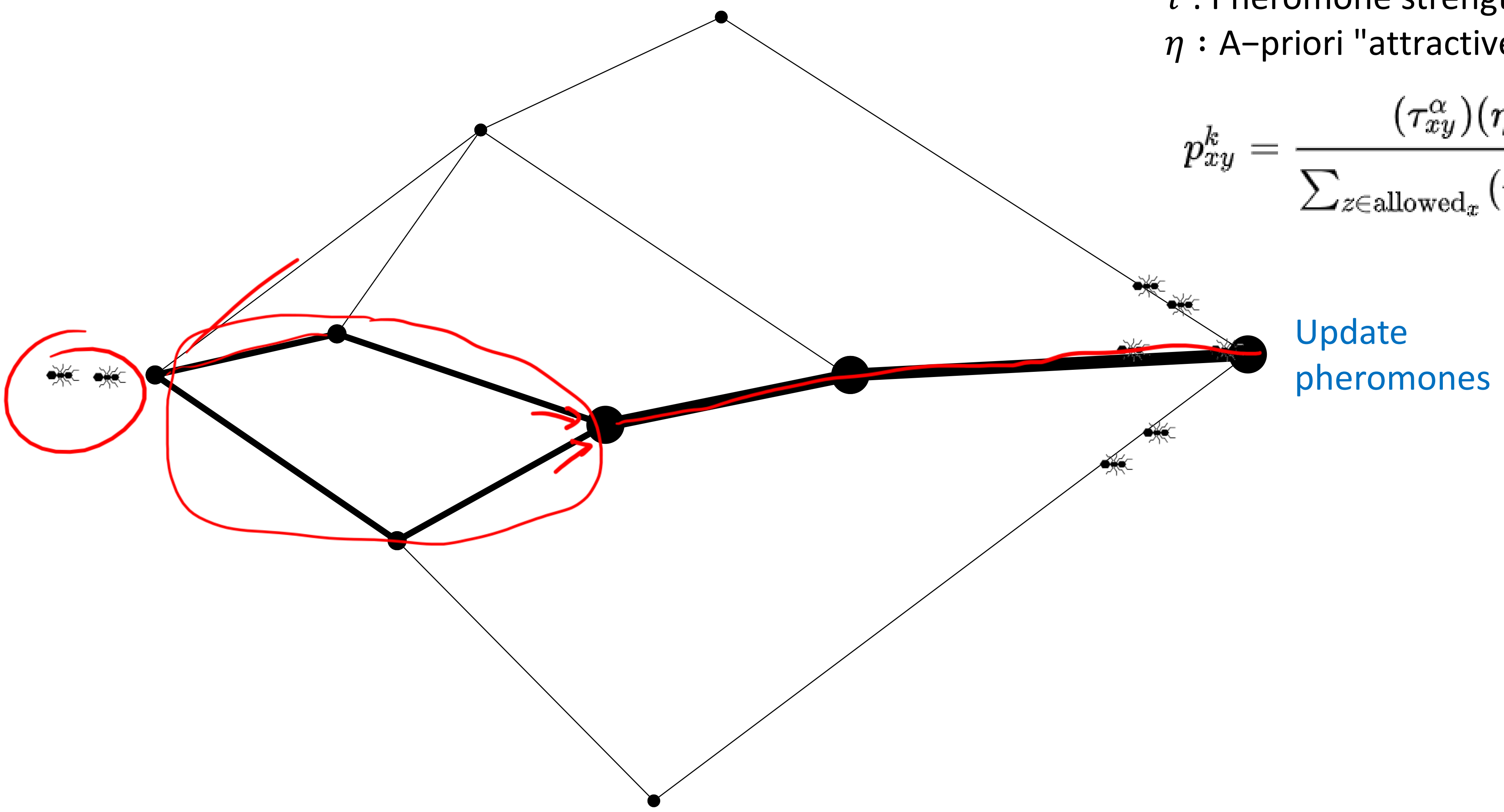
τ : Pheromone strength

η : A–priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$



Ant Colony Optimisation (ACO) – TSP



Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A–priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \mathrm{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Ant Colony Optimisation (ACO) – TSP

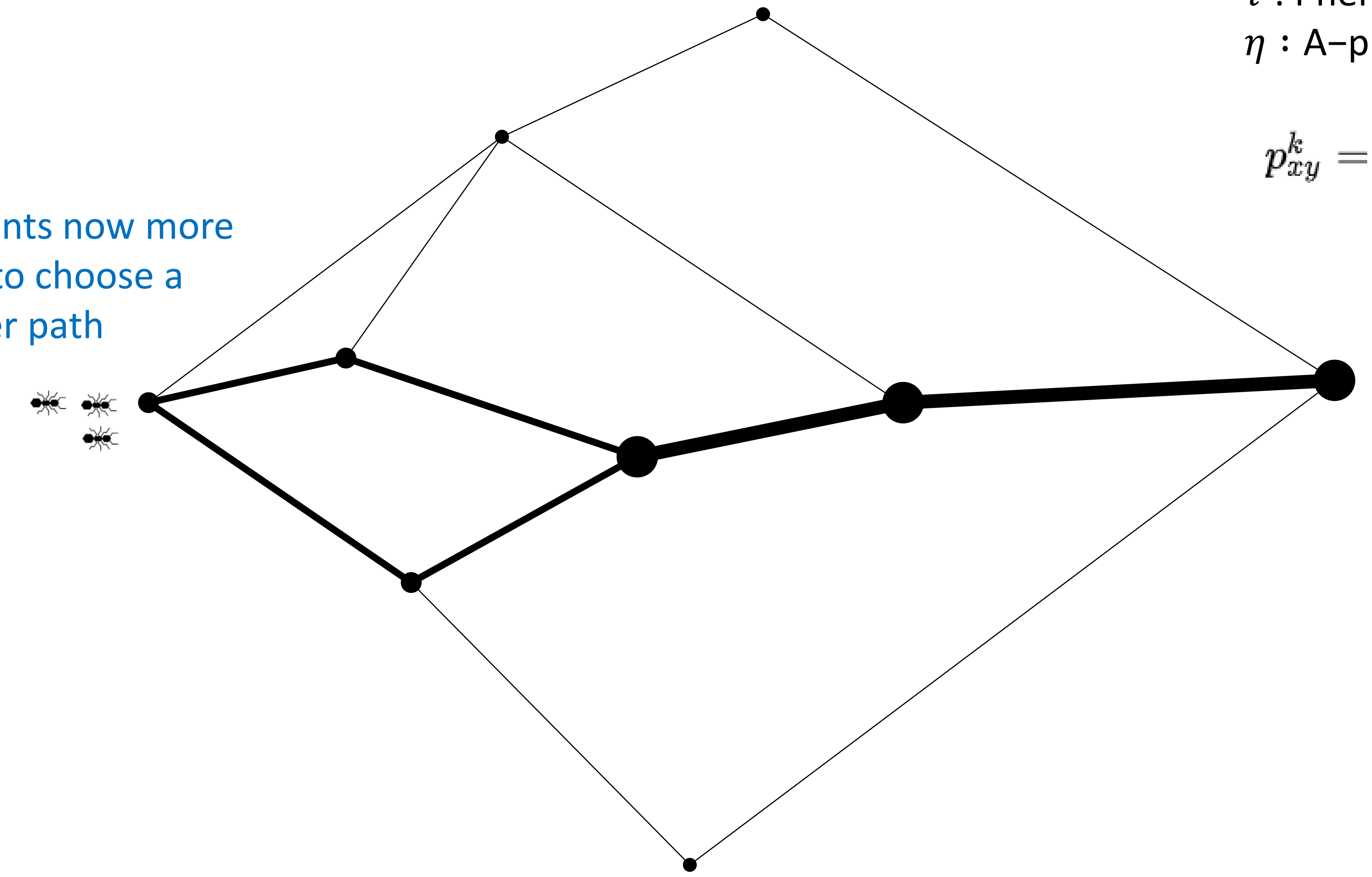
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A–priori "attractiveness"

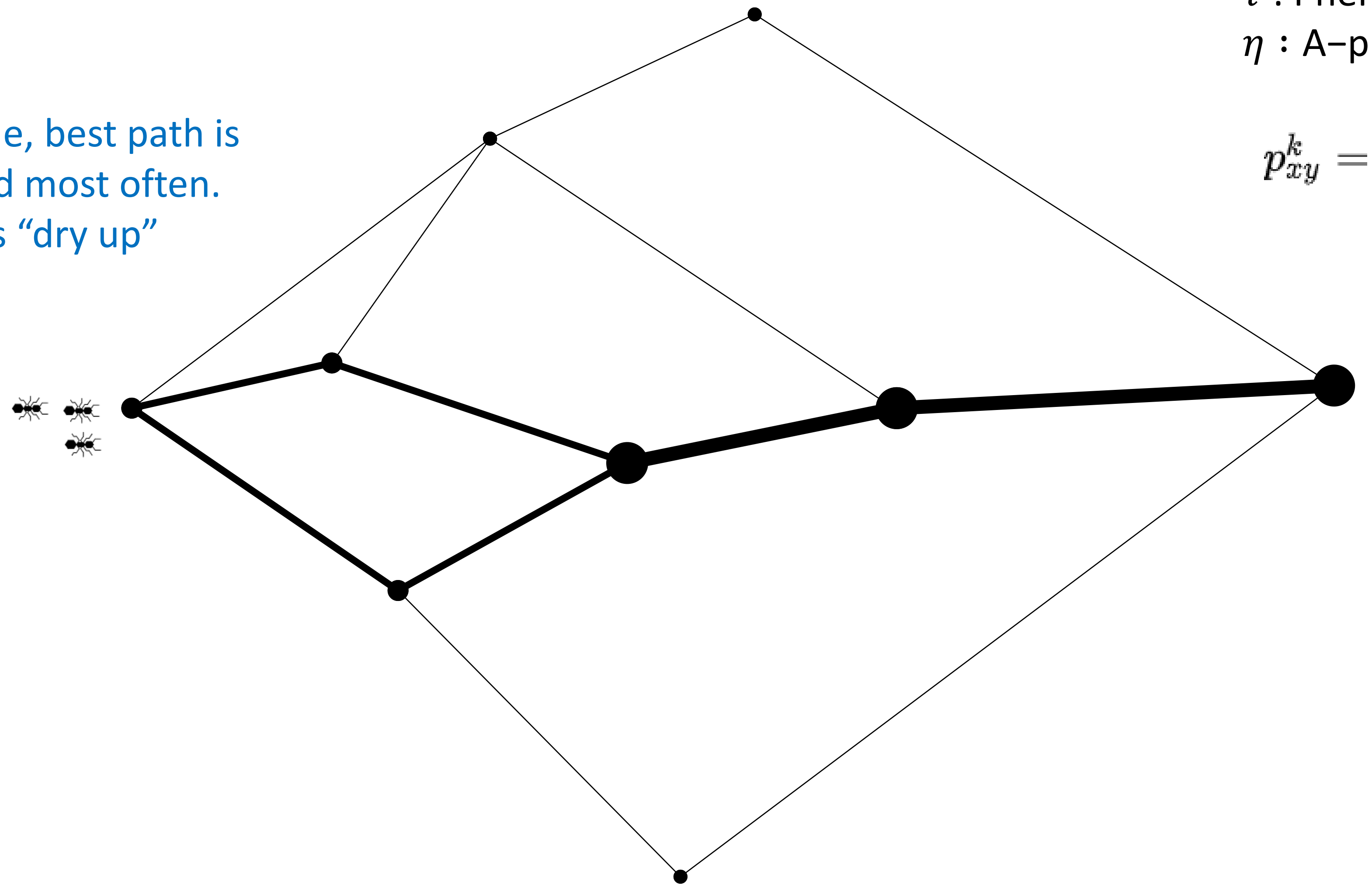
$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \mathrm{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

New ants now more likely to choose a shorter path



Ant Colony Optimisation (ACO) – TSP

Over time, best path is traversed most often. Old trails “dry up”



Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A−priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \mathrm{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Ant Colony Optimisation (ACO) – TSP

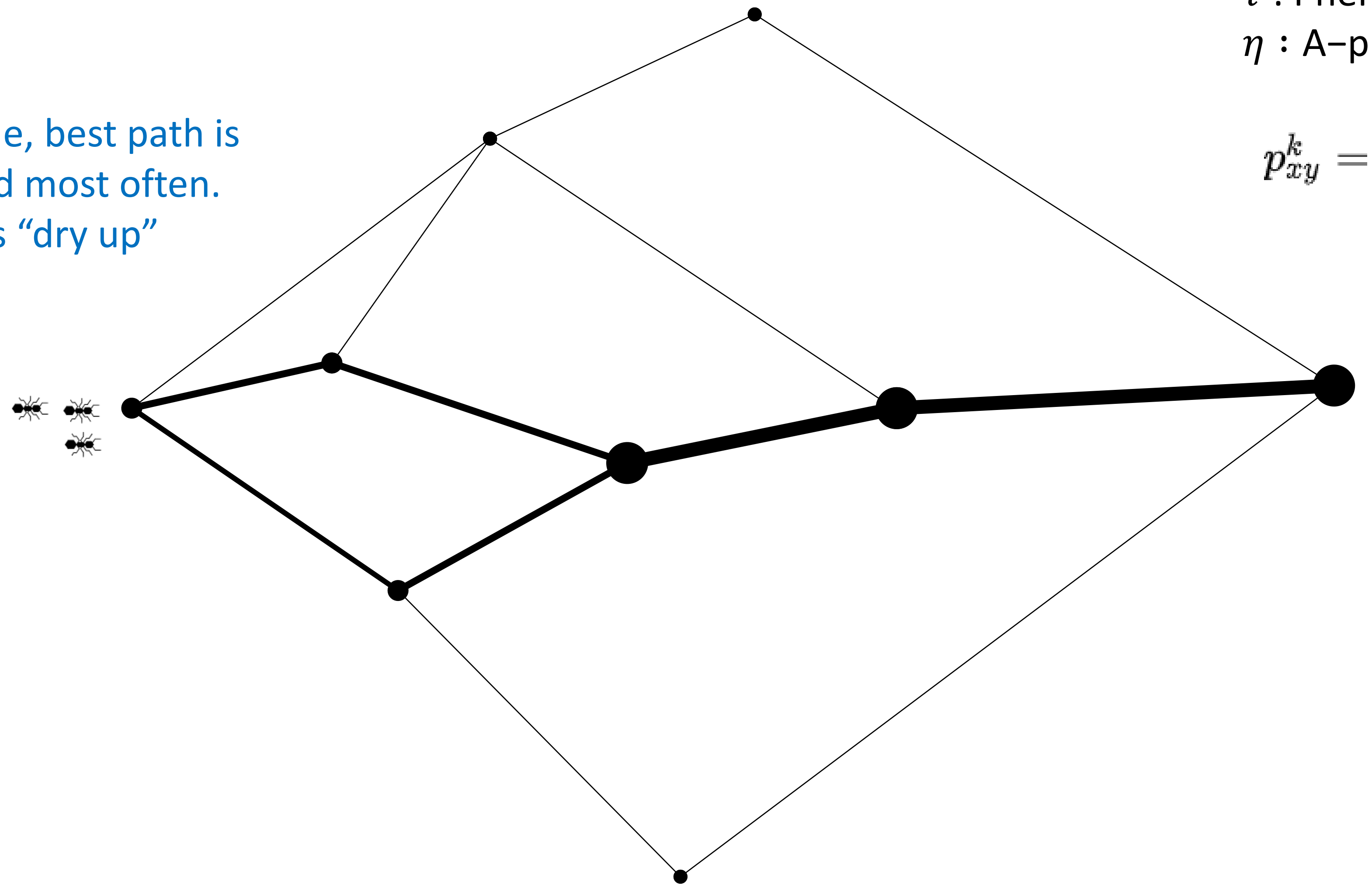
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A–priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Over time, best path is traversed most often. Old trails “dry up”



Ant Colony Optimisation (ACO) – TSP

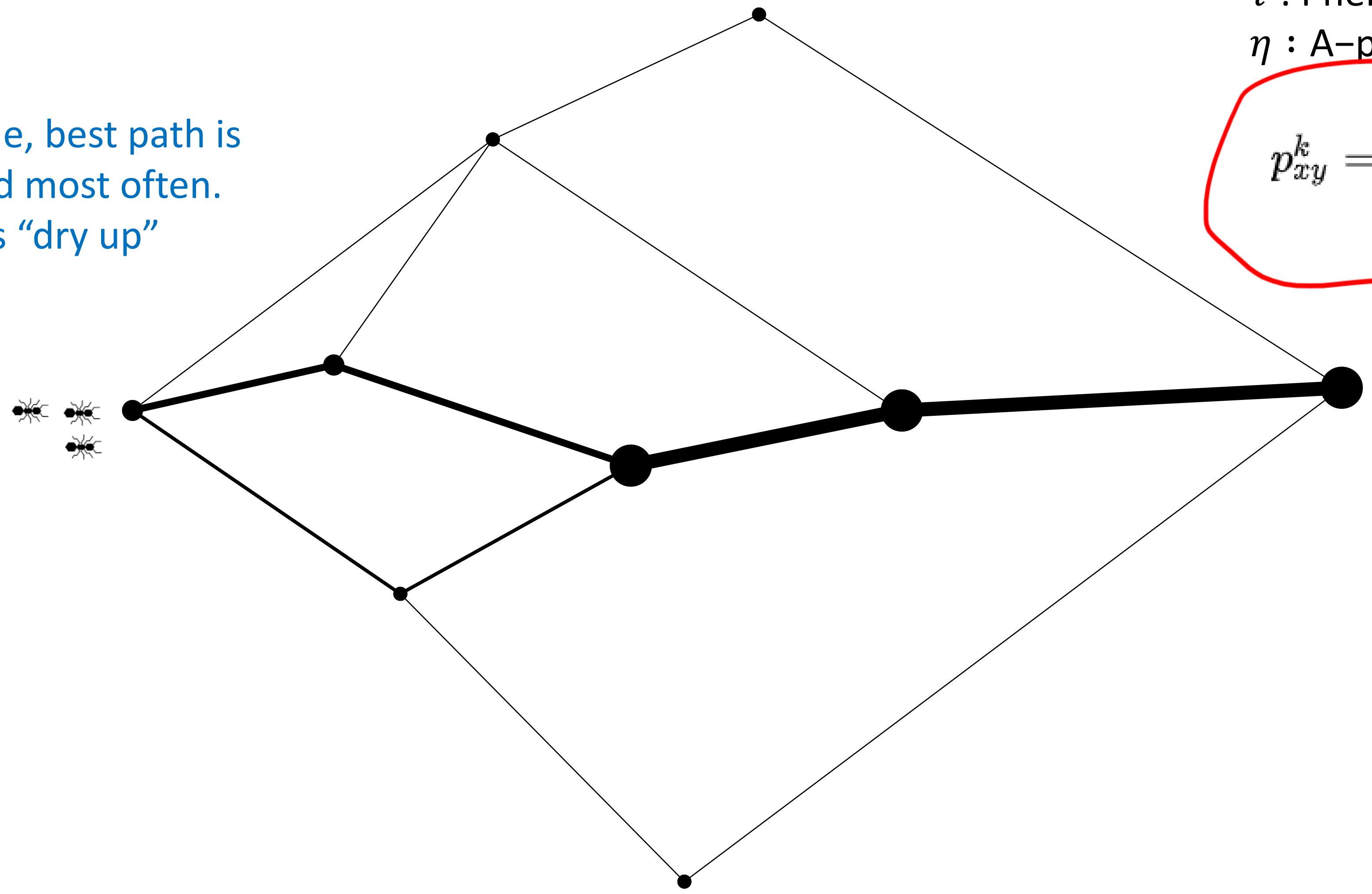
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A–priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \mathrm{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Over time, best path is traversed most often.
Old trails "dry up"



Ant Colony Optimisation (ACO) – TSP

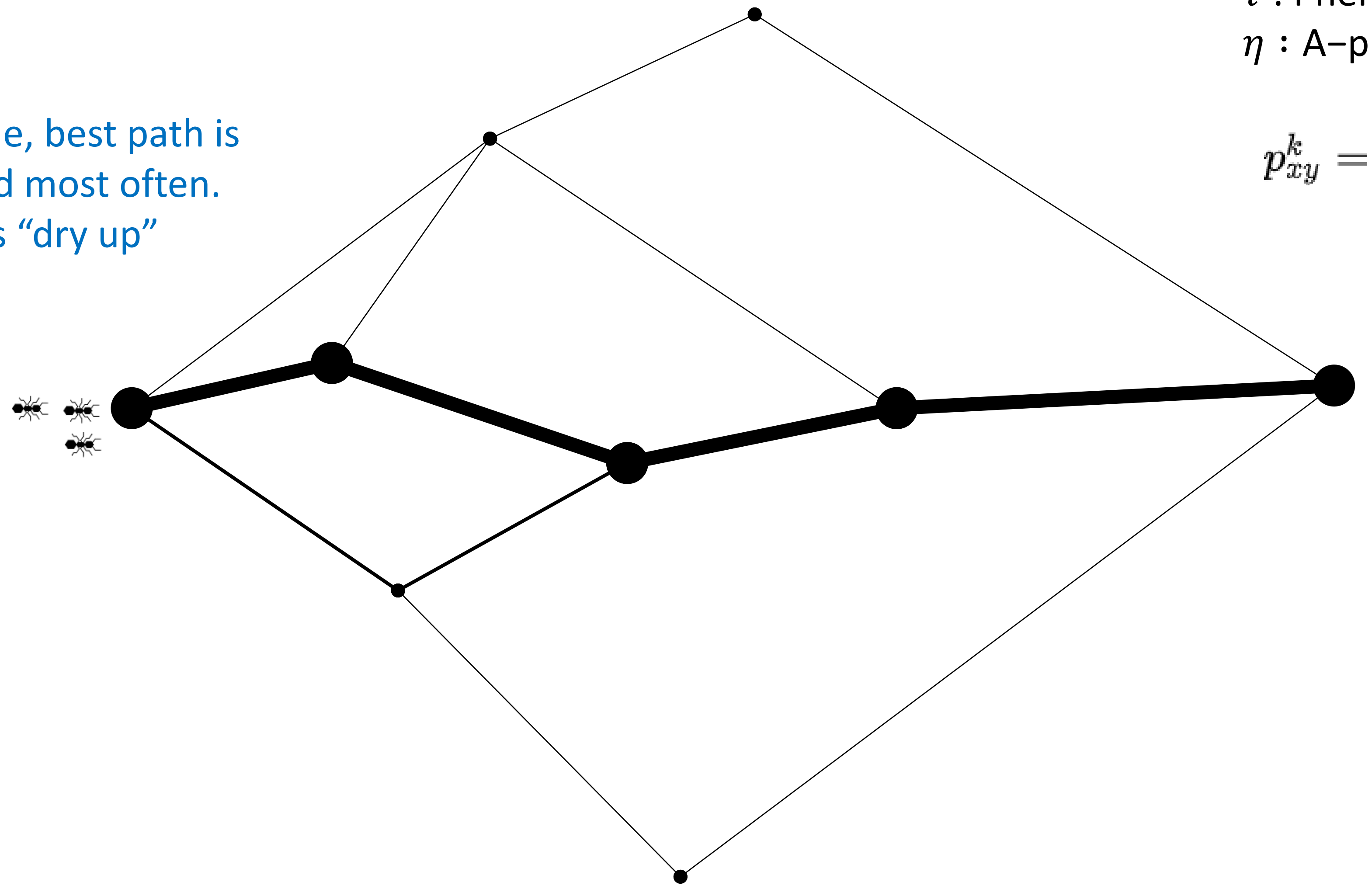
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A–priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \mathrm{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Over time, best path is traversed most often. Old trails “dry up”



Ant Colony Optimisation (ACO) – TSP

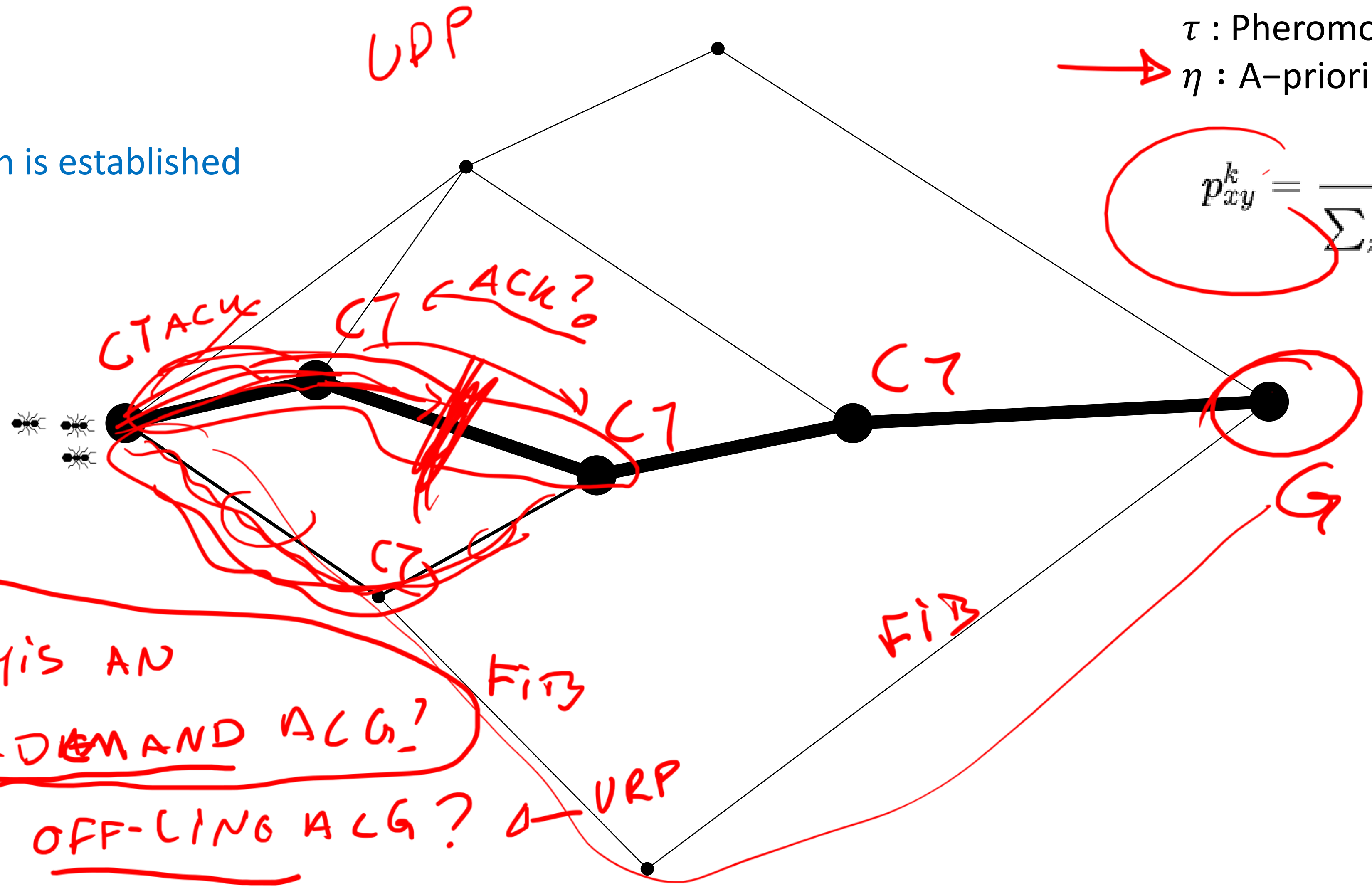
Probability that ant chooses arc *xy*

τ : Pheromone strength

η : A–priori "attractiveness"

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

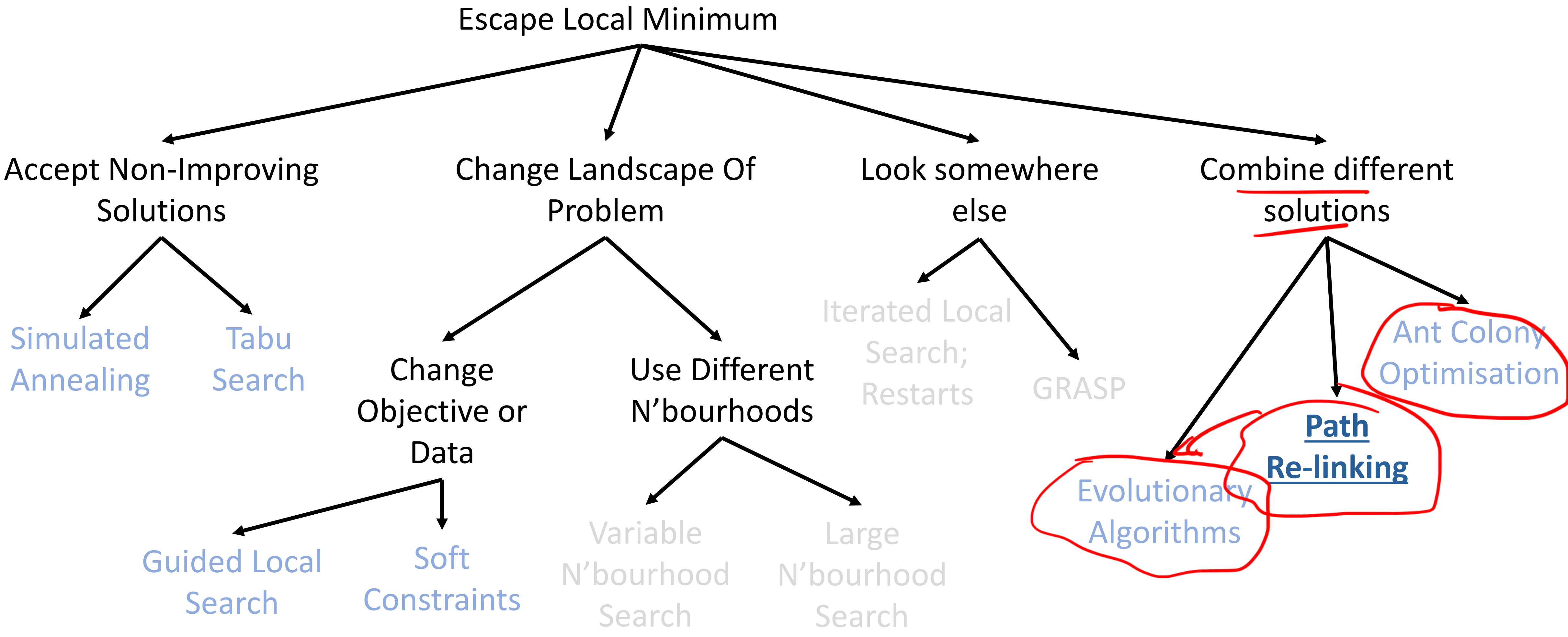
Best path is established



IS THIS AN ON-DEMAND ALG?

OR AN OFF-LINE ALG?

Meta-heuristics: An Incomplete Survey



Path Relinking

GA

POP-BASED ALG

BIG # OF OFFS

Basic idea:

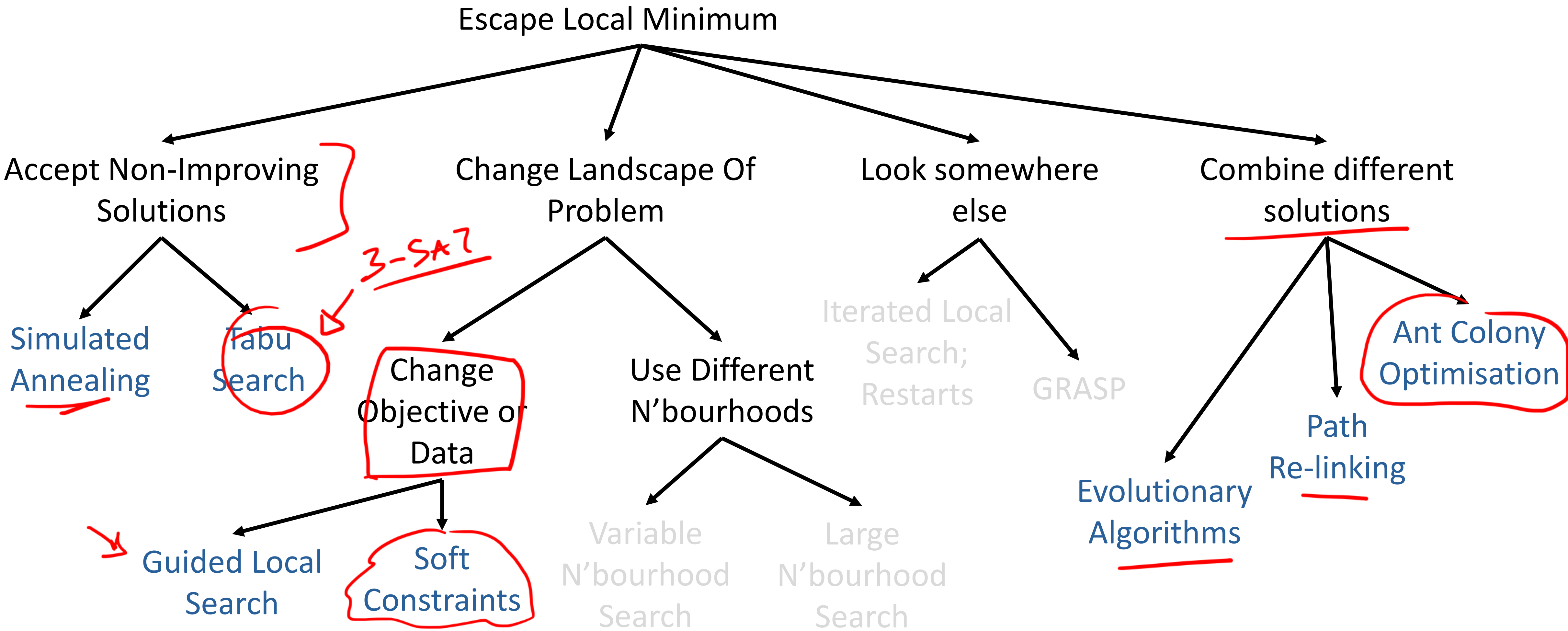
- Take two solutions
- "Walk" between them

For TSP:

4 DIF SOL.

S1	1	4	6	2	3	8	9	5	10	7
	1	10	6	2	3	8	9	5	4	7
	1	10	8	2	3	6	9	5	4	7
	1	10	8	2	5	6	9	3	4	7
	1	10	8	2	5	4	9	3	6	7
S2	1	10	8	2	5	4	9	3	6	7

Meta-heuristics



Conclusions

NAME OF BOOK

- We've looked at the characteristics of Meta-heuristics
  - Problem independent on top of Problem-dependent heuristics
  - Often Randomised – "Stochastic Local Search" (also a book by Hoos & Stützle)
  - Diversification / Intensification (Exploration / Exploitation)
- We've looked at a few Meta-heuristics
  - Accepting non-improving solutions
  - Combining solutions
- Next time
  - More Meta-heuristics

IN A2

V.N.S

L.N.S