

Final Exam (due 6/16/17)

Instructor: 장형수 교수님

Student: 120170162 이시영

Problem 1

Describe clearly difference and similarities among the functionalities of GA, SA, ACO, PTS in the perspective of the exploration and the exploitation process.

Solution: Meta-heuristic algorithm들은 combinatorial optimization problem에서의 해 공간을 효율적이고 효과적으로 탐색하는 것을 목표로 설계되었다는 특징을 가진다. 이러한 관점에서 각각의 Meta-heuristic algorithm들(GA, SA, ACO, PTS 등)은 좋은 해가 나오는 영역을 집중적으로 탐색(exploitation)할 것인지 탐색이 이루어지지 않은 부분으로 이동 후 탐색(exploration)을 진행할 것인지를 상황에 알맞게 동적으로 결정한다.

즉, exploitation과 exploration 간의 균형은 탐색공간에서 global optimal point를 얻기 위해 고품질의 해들의 영역을 집중적으로 파악하고, 그렇지 않은 영역에서는 탐색 시간을 낭비하지 않도록 결정한다는 점에서 Meta-heuristic Algorithm의 핵심에 해당한다.

Meta-heuristic algorithm의 탐색방법은 각기 다른 Meta-heuristic algorithm 고유의 component에 매우 의존적이며, 크게 2가지로 분류할 수 있다.

1. local search algorithm에서의 탐색을 개선한 효율적 탐색 algorithm
2. 좋은 품질을 결정하는 decision variables의 상호관계를 학습하는 algorithm

PTS와 SA 같은 algorithm이 전자의 경우에 속하며, local minimum point에 빠지지 않게 하면서 효율적인 탐색 방법을 통해 더 좋은 minimum point를 찾으므로써 global optimal point를 찾는 algorithm들이다. 이러한 algorithm의 경우, 탐색공간의 하나의 solution은 명시적인 neighbor를 가지는 neighborhood structure를 가지며, 이들 간의 전이가 Markov chain를 가지면서 최적해를 찾아간다는 공통점이 있다.

후자의 경우, GA, ACO와 같은 algorithm이 있으며, 전자와 달리 좋은 품질을 갖는 해들의 공통된 특징(GA에서는 scheme이라고 부름)과 이들 간의 상관관계를 학습하려고 한다는 점에서 차이를 갖는다. 이러한 종류의 Meta-heuristic algorithm의 탐색은 탐색공간에서의 biased sampling을 통해 이루어지며, GA에서는 해들의 recombination 과정으로, ACO에서는 pheromone과 desirability에 의해 계산되는 확률분포에 따라 전이가 결정되는 과정으로 나타난다.

Meta-heuristic에서의 exploitation stage는 elite solutions의 neighbor들에 초점을 맞추어 탐색하는 반면, exploration stage에서는 아직 탐색되지 않은 영역을 조사하거나 이전에 등장하지 않았던

해들을 생성하는데 초점을 맞춘다. 즉, exploration stage에 의해, Meta-heuristic이 local minimum point에 빠지지 않고, exploitation stage을 통해 global minimal point를 찾아간다고 할 수 있다.

Meta-heuristic algorithm의 exploitation과 exploration의 성격은 각각의 algorithm의 component의 효과로 인해 나타나며, 이러한 component를 각각의 algorithm내에서 살펴보면 다음과 같다.

Meta-heuristic	Component
PTS	adaptive tabu list
SA	probabilistic acceptance criterion + cooling schedule
ACO	pheromone update, probabilistic construction
GA	mutation, selection, recombination

GA에서는 recombination, mutation, selection 연산을 통한 탐색 전략을 이용해 최적해를 찾는다. 각각의 연산이 갖는 exploration, exploitation적 성격을 살펴보면 다음과 같다.

- recombination 연산: 주어진 population에서의 해들의 조합을 통해, 기존에 탐색되지 않았던 해를 생성한다는 점에서 exploration 효과를 가진다.
- mutation 연산: recombination에서 발생하지 않는 pattern을 찾는다라는 점에서 exploration 효과를 가지지만, 특정 확률로 binary string으로 표현된 해의 bit를 바꾸어주는 mutation 연산을 적용하는 경우에는 상대적으로 "Hamming distance"가 가까운 해로 mutation될 확률이 높기 때문에, exploitation 효과도 가지고 있음을 알 수 있다.
- selection 연산: 추가적으로 생성된 자식 해에서 품질이 좋은 해들을 선택하여, 다음 세대로 전달하는 과정은 exploitation 효과를 가진다.

SA의 경우에는 temperature 변수가 점점 감소하면서 탐색의 범위가 줄어드는 cooling schedule과 탐색된 해를 받아들일지를 결정하는 acceptance criterion

$$\min(1, e^{-\frac{\Delta}{T}}) \text{ where } \Delta = \text{cost of new state} - \text{cost of current state}$$

에 의해, SA의 탐색전략이 변화한다. acceptance criterion이 목적함수의 변화량이 반영되어 설정되고, 탐색된 상태로의 전이가 확률적으로 일어나면서 탐색이 이루어지며, 추가적으로 temperature 변수의 감소(cooling schedule)되면서 탐색 전략이 exploration에서 exploitation으로 변하여 결과적으로 SA가 global solution을 찾을 수 있도록 한다.

ACO에서는 pheromone values을 update하는 과정에서 나타난다. Algorithm 내에 pheromone을 update하는 과정에서, ant들이 다른 상태로 전이하는 확률을 결정하는 확률 분포를 바꿈으로써 탐색과정이 진행된다. 이 같은 과정은 탐색공간에서의 해를 sampling하는 확률 분포를 바꾸는 과정으로, ant가 효과적이며 효율적으로 global solution을 탐색할 수 있도록 한다. 다시 말하면, ACO에서는 해들간의 잠재적 특징(latent variables)이 construction graph에서의 edge로 나타나기 때문에 ant가 construction graph에서 traverse하면서 좋은 해들에서 등장하는 edge에 더 많은 양의 pheromone이 누적함으로써, 전역적 해를 찾게된다. 이 메커니즘은 탐색 전략에서의 exploitation을 바탕으로 하지만, 확률 분포를 계산하는 과정에서 pheromone 뿐만아니라 desirability 요소가 고려되므로, exploration 효과도 나타난다.

PTS에서는 tabu list를 이용한 현재 찾은 해로부터 이웃을 선택하는 방법을 통해 최적의 해를 찾는다. Tabu list에 최근에 방문했던 state에 대한 정보를 저장함으로써, 최근 탐색했던 해들이 neighborhood에 포함되지 않도록 하기 때문에 exploration 효과를 내며, 제한된 neighborhood 내에서 최상의 neighbor를 선택한다는 점에서는 exploitation 효과를 갖는다. 이 두 가지 효과의 균형은 tabu list의 길이에 따라 달라질 수 있으며, tabu list의 길이가 짧을수록 exploration 효과가 낮아지고, 길이가 길어질수록 exploration 효과가 커진다. 이러한 관점에서 PTS는 다른 TS와는 다르게 tabu list가 적응적으로 바뀐다는 장점을 가진다.

위에서 살펴본 바와 같이, 각각의 Meta-heuristic algorithm의 component는 단순히 exploration 또는 exploitation의 역할만 하는 것이 아닌 exploration/exploitation을 수행한다는 공통점을 가지지만, 효율적 탐색을 기반으로 하는 알고리즘인지 해들의 잠재적인 특성들의 상관관계를 학습하는 알고리즘인지와 같은 차이점이 있다. 또한 해당 component의 성격이 조금 더 exploration 또는 exploitation에 맞추어져 있는지에 대한 차이점이 존재한다. 이러한 점에서 PTS의 tabu list는 다른 algorithm의 component에 비해 좋은 exploration 능력을 갖고 있다고 할 수 있다.

최근에는 이러한 기본적인 meta-heuristic algorithm에서 등장하는 exploitation, exploration의 관계를 분석하여, 각 component의 장점을 혼합한 hybrid algorithm을 설계하기도 한다. 그 예로, PTS에서의 tabu list를 이용한 exploration 기법은 다른 Meta-heuristic algorithm에서 나타나는 local minimum point에 자주 빠지는 문제를 효율적 해결한다는 점에서, GA, SA 등과 혼합하여 사용한다.

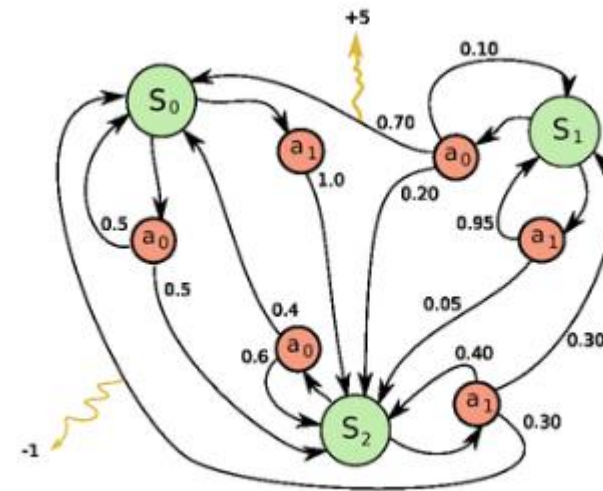
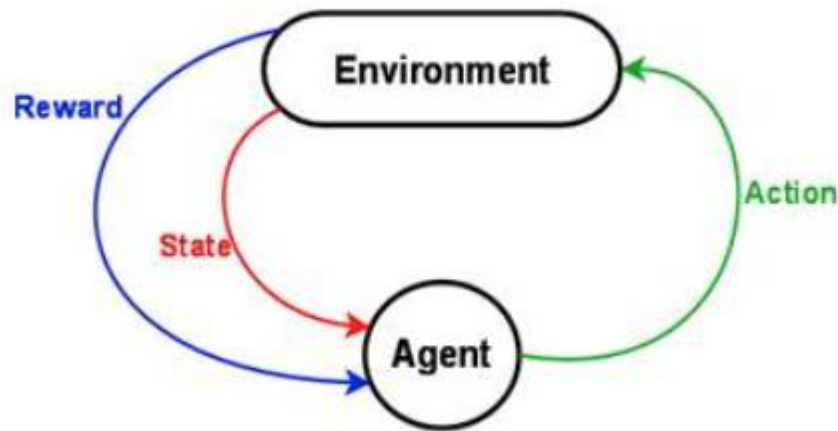
References

- [BCA03] BLUM, CHRISTIAN, AND ANDREA ROLI, *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*, ACM Computing Surveys (CSUR) 35.3 (2003): 268-308.

Problem 2

Introduce any metaheuristic algorithm other than the ones we covered in the class by providing its algorithmic description and discussing convergence results.

Solution: MDP, Q-learning



Introduction

■ Markov Decision Process

- A mathematical framework for modeling decision making in situations where outcome are partly random and partly under the control of a decision marker.

■ Goal

- To find an “Optimal policy”
 - A policy π that specifies the action $\pi(s) = a$ that the agent will choose a when in state s
- To choose a policy π that will **maximize some cumulative function** of the random reward, typically the expected discount sum over a potentially infinite horizon.

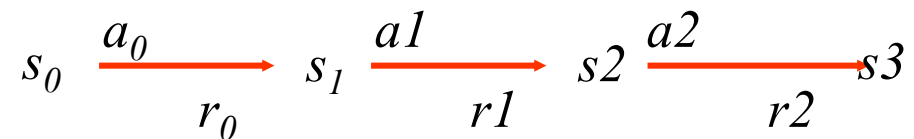
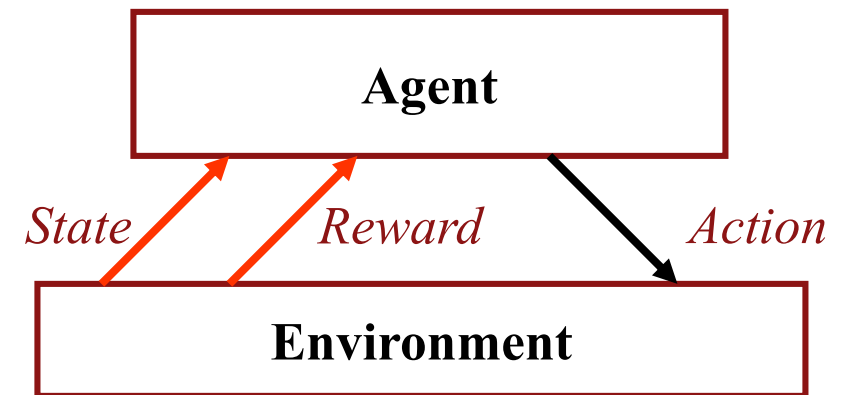


Figure 1: Markov Decision process

Representation

■ Terminology

- Markov Decision Process: 4-tuple (S, A, P, r)

Definition 1. Markov Decision Process: 4-tuple (S, A, P, r)

- S : finite set of states
- A : finite set of actions
- P : transition probabilities
- r : the reward function $(S \times A \times S \rightarrow \mathbb{R})$
 - Assigning a reward $r(x, a, y)$ everytime a transition from x to y occurs due to action a

Representation

▪ Markov Decision Process Graph

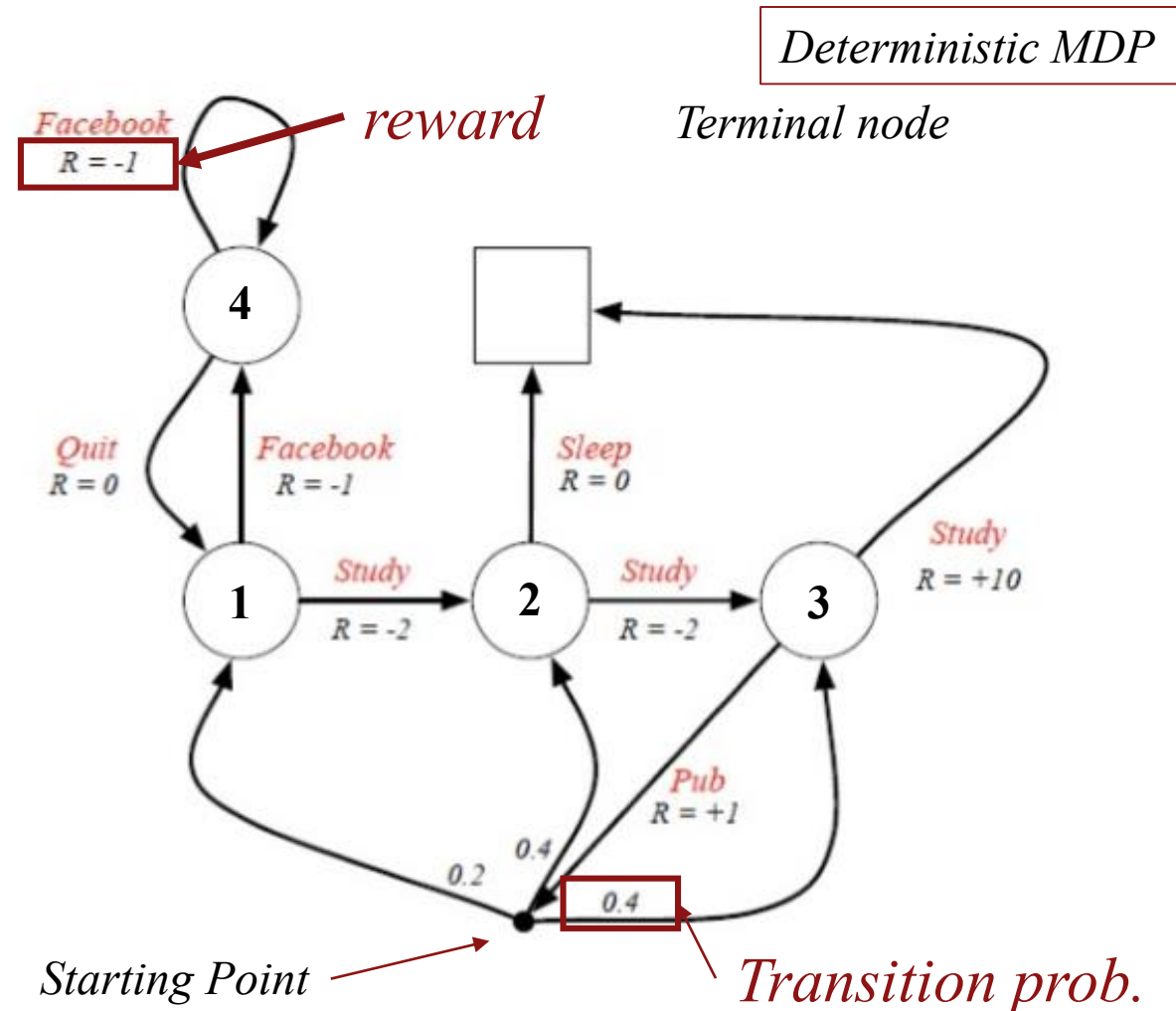
- e.g) Student state

- State set $S =$

$\{ \textcircled{1}, \textcircled{2}, \textcircled{3}, \textcircled{4}, \square \}$

- Action set A
= {Facebook, Quit, study, Sleep, Pub}
 - Transition prob. P
 - Reward R

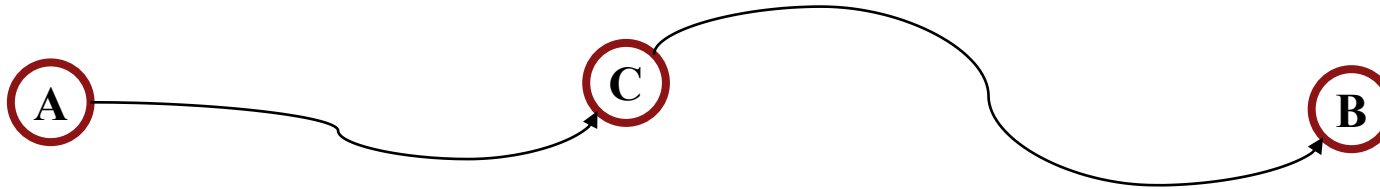
- Deterministic MDP
 - If agent A selects an action, then A can do this action with probability “1”
- Non-deterministic MDP
 - If agent A selects an action, then A can do this action with any probability



Introduction

- Idea of algorithm

- Using optimal substructure! → Dynamic Programming!
 - If C belongs to an optimal path from A to B , then the sub-path A to C and to B are also optimal.
 - Hence, all sub-path of an optimal path is optimal



Definition 1. Bellman's Optimality Equation

$$V^*(s) = \sum_{s'} P(s, s', \pi(s)) (R(s, s', \pi(s)) + \gamma V^*(s'))$$
$$\pi^*(s) = \operatorname{argmax}_a \left\{ \sum_{s'} P(s, s', \pi(s)) (R(s, s', \pi(s)) + \gamma V^*(s')) \right\}$$

Q-learning

- Can be used to find an optimal action policy for any given MDPs
- Converges to an optimal policy
in both deterministic and non-deterministic MDPs.
- Generally, implemented by tabular method → 2D - Array

Definition 2. Q-function

Define a functions $Q^*: S \times A \rightarrow \mathbb{R}$ such that

$$Q^*(s, a) = \sum_{s' \in S} P(s, s', a) R(s, s', a) + \gamma \sum_{s' \in S} P(s, s', a) V^*(s')$$

- 
- *Q-function measures of **how good** to take an action $a \in A$ at current state $s \in S$
if **an optimal policy is followed from the possible next state of s***

Pseudocode

Algorithm 1 Q-learning

```

1: Initialize  $Q_0(s, a)$  arbitrarily.
2: repeat
3:   Choose  $a_t$  from  $s_t$  using an exploratory policy  $\phi_t$ .
4:   Take action  $a_t$ , observe  $r, s_{t+1}$ .
5:   Update  $Q$  value function such that
6:    $Q_{t+1}(s_t, a_t) := Q_t(s_t, a_t) + \alpha[r + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)]$ 
7:    $t := t + 1$ 
8: until policy-table do not change
  
```

■ Learning rate α

- Determine to what extent the newly acquired information will override the old information
 - If $\alpha = 0$, the agent **do not learning anything.**
 - If $\alpha = 1$, the agent consider **only the most recent information**

■ Discount factor γ

- Determine the importance of future rewards.
 - If $\gamma = 0$, the agent consider only current reward.
 - If $\gamma = 1$, the agent consider future reward for a long term.

Pseudocode

Algorithm 1 Q-learning

```

1: Initialize  $Q_0(s, a)$  arbitrarily.
2: repeat
3:   Choose  $a_t$  from  $s_t$  using an exploratory policy  $\phi_t$ .
4:   Take action  $a_t$ , observe  $r, s_{t+1}$ .
5:   Update  $Q$  value function such that
6:    $Q_{t+1}(s_t, a_t) := Q_t(s_t, a_t) + \alpha[r + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)]$ 
7:    $t := t + 1$ 
8: until policy-table do not change

```

■ ϵ -greedy policy

- At time t ,

$$\phi_t = \begin{cases} \text{with prob. } \epsilon_t(s) = \frac{c}{n_t(s)}, \text{ select action } a \in A \text{ with prob. } \frac{1}{|A|} \\ \text{with prob. } 1 - \epsilon_t(s), \text{ select an action in "argmax}_{a \in A} [Q_t(s_t, a)] \end{cases}$$

✓ ϕ_t becomes more greedy selection rule with respect to Q_t at t .

$$\text{if } \lim_{t \rightarrow \infty} n_t(s) = \infty, \lim_{t \rightarrow \infty} \epsilon_t(s) = 0$$

✓ * $n_t(s)$: the number of visits to state s in time step t , $0 < c < 1$

Convergence behavior to an optimal solution

Definition 1. Markov Decision Process: 4-tuple (X, A, P, r)

- X : finite set of states
- A : finite set of actions
- P : transition probabilities
- r : the reward function $(X \times A \times X \rightarrow \mathbb{R})$
 - Assigning a reward $r(x, a, y)$ everytime a transition from x to y occurs due to action a

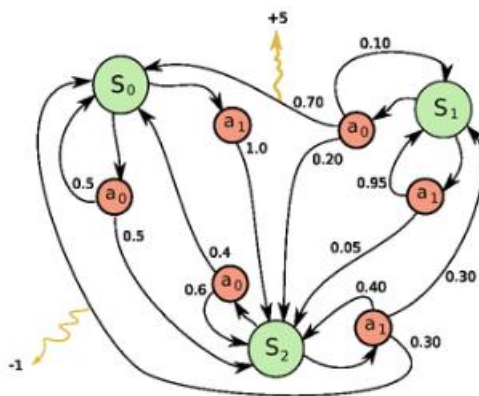


Figure 2: A example of Markov Decision process graph

Convergence behavior to an solution

■ Terminology

- The value of a state x
 - For a sequence $\{A_t\}$,

$$J(x, \{A_t\}) = E\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) | X_0 = x\right]$$

- The optimal value function

For each $x \in X$,

$$V^* = \max_{A_t} J(x, \{A_t\})$$

$$= \max_{a \in A} \sum_{y \in X} P_a(x, y) [r(x, a, y) + \gamma V^*(y)]$$

■ Preliminaries

- Supremum Norm $\|f\|_{\infty}$

$$\|f\|_{\infty} = \|f\|_{\infty, S} = \sup \{ |f(x)| : x \in S \}.$$

- The optimal Q -function is a contraction operator \mathbf{H} , defined for a generic function $q: X \times A \rightarrow \mathbb{R}$ as

$$(\mathbf{H}q)(x, a) = \sum_{y \in X} P_a(x, y) [r(x, a, y) + \gamma \max_{b \in A} q(y, b)]$$

Convergence behavior to an solution

■ Preliminaries

$$\begin{aligned}
 \bullet \quad \| \mathbf{H}q_1 - \mathbf{H}q_2 \|_\infty &= \max_{x,a} \left| \sum_{y \in X} P_a(x,y) [r(x,a,y) + \gamma \max_{b \in A} q_1(y,b) - r(x,a,y) + \gamma \max_{b \in A} q_2(y,b)] \right| \\
 &= \max_{x,a} \gamma \left| \sum_{y \in X} P_a(x,y) [\max_{b \in A} q_1(y,b) - \max_{b \in A} q_2(y,b)] \right| \\
 &\leq \max_{x,a} \gamma \sum_{y \in X} P_a(x,y) \left| \max_{b \in A} q_1(y,b) - \max_{b \in A} q_2(y,b) \right| \\
 &\leq \max_{x,a} \gamma \sum_{y \in X} P_a(x,y) \max_{z,b} |q_1(z,b) - q_2(z,b)| \\
 &\leq \max_{x,a} \gamma \sum_{y \in X} P_a(x,y) \|q_1 - q_2\|_\infty \\
 &= \gamma \|q_1 - q_2\|_\infty
 \end{aligned} \tag{1}$$

Convergence behavior to an optimal solution

Theorem 1.

Suppose that $\sum_t \alpha_t(x, a) = \infty$, $\sum_t \alpha_t^2(x, a) = \infty$ for all $(x, a) \in X \times A$

Given a finite MDP (X, A, P, r) , the Q-learning algorithm, given by the update rule

$$Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \alpha_t(x_t, a_t) \left[r_t + \gamma \max_{b \in A} Q_t(x_{t+1}, b) - Q_t(x_t, a_t) \right] \dots (2)$$

Converge w.p.1 to the optimal Q-function.

Proof.

By rewriting (2), we can lead to

$$Q_{t+1}(x_t, a_t) = (1 - \alpha_t(x_t, a_t))Q_t(x_t, a_t) + \alpha_t(x_t, a_t)[r_t + \gamma \max_{b \in A} Q_t(x_{t+1}, b)]$$

Convergence behavior to an optimal solution

Proof.

By subtracting $Q^*(x_t, a_t)$ from both sides and letting $\Delta_t(x, a) = Q_t(x, a) - Q^*(x, a)$,

$$\Delta_t(x_t, a_t) = (1 - \alpha_t(x_t, a_t))\Delta_t(x_t, a_t) + \alpha_t(x, a)[r_t + \gamma \max_{b \in A} Q_t(x_{t+1}, b) - Q^*(x_t, a_t)]$$

And let $F_t(x, a) = r(x, a, X(x, a)) + \gamma \max_{b \in A} Q_t(y, b) - Q^*(x, a)$ where $X(x, a)$ is a random sample state obtained from the Markov chain (X, P_a) .

we can induce that

$$\begin{aligned} \mathbf{E}(F_t(x, a) | F_t) &= \sum_{y \in X} P_a(x, y)[r(x, a, y) + \gamma \max_{b \in A} Q_t(y, b) - Q^*(x, a)] \\ &= (\mathbf{H}Q_t)(x, a) - Q^*(x, a) \end{aligned}$$

Convergence behavior to an optimal solution

Proof.

By using " $Q^* = \mathbf{H}Q^*$ ", we have

$$\mathbf{E}(F_t(x, a)|F_t) = (\mathbf{H}Q_t)(x, a) - (\mathbf{H}Q^*)(x, a)$$

Therefore, by (1)[page. 11]

$$\|\mathbf{E}(F_t(x, a)|F_t)\|_\infty \leq \gamma \|Q_t - Q^*\|_\infty = \gamma \|\Delta\|_\infty$$

And,

$$\begin{aligned} & \text{var}(F_t(x)|F_t) \\ &= \mathbf{E} \left[\left(r(x, a, X(x, a)) + \gamma \max_{b \in A} Q_t(y, b) - Q^*(x, a) - (\mathbf{H}Q_t)(x, a) + Q^*(x, a) \right)^2 \right] \\ &= \mathbf{E} \left[\left(r(x, a, X(x, a)) + \gamma \max_{b \in A} Q_t(y, b) - (\mathbf{H}Q_t)(x, a) \right)^2 \right] \end{aligned}$$

Convergence behavior to an optimal solution

Proof.

$$\begin{aligned}
 \text{var}(F_t(x)|F_t) &= \mathbf{E} \left[\left(r(x, a, X(x, a)) + \gamma \max_{b \in A} Q_t(y, b) - Q^*(x, a) - (\mathbf{H}Q_t)(x, a) + Q^*(x, a) \right)^2 \right] \\
 &= \mathbf{E} \left[\left(r(x, a, X(x, a)) + \gamma \max_{b \in A} Q_t(y, b) - (\mathbf{H}Q_t)(x, a) \right)^2 \right] \\
 &= \text{var} \left[r(x, a, X(x, a)) + \gamma \max_{b \in A} Q_t(y, b) | F_t \right] \leq C(1 + \|\Delta_t\|_w^2) \quad (\because r \text{ is bounded})
 \end{aligned}$$

Then by lemma 2.(next page),

Δ_t converges to zero w. p. 1

So, we can deduce that Q_t converges to Q^* w.p.1.

Convergence behavior to an optimal solution

Lemma 2. (ref [2])

The random process $\{\Delta_t\}$ taking values in \mathbb{R}^n and defined as

$$\Delta_{t+1}(x) = (1 - \alpha_t(x))\Delta_t(x) + \alpha_t(x)F_t(x)$$

converges to zero w.p.1 under the following assumptions

- $0 \leq \alpha_t \leq 1$, $\sum_t \alpha_t(x) = \infty$ and $\sum_t \alpha_t^2(x) = \infty$
- $\| \mathbf{E} [F_t(x)|F_t] \|_w \leq \gamma \|\Delta_t\|_w$, with $\gamma < 1$
- $\text{var} [F_t(x)|F_t] \leq C(1 + \|\Delta_t\|_w^2)$, for $C > 0$.

$\|\cdot\|_w$: Weighted maximum norm

References

- Melo, Francisco S. "Convergence of Q-learning: A simple proof." *Institute Of Systems and Robotics, Tech. Rep* (2001): 1-4.
- Jaakkola, Tommi, Michael I. Jordan, and Satinder P. Singh. "On the convergence of stochastic iterative dynamic programming algorithms." *Neural computation* 6.6 (1994): 1185-1201.

Problem 3

Suppose you wish to apply ACO to a minimum spanning tree problem. Provide pseudocode and convergence results of your algorithm. Show how your algorithm works with a small toy problem.

Problem Setting

Problem 1. Minimum Spanning Tree Problem

Given an undirected connected graph $G = (V, E)$ with edge costs (weights) $w: E \rightarrow \mathbb{N}_{\geq 1}$, Find a spanning tree $E^* \subseteq E$ such that the total cost $\sum_{e \in E^*} w(e)$ become minimal

Denote $n := |V|$ and $m := |E|$ and assume that $E = \{1, 2, \dots, m\}$.

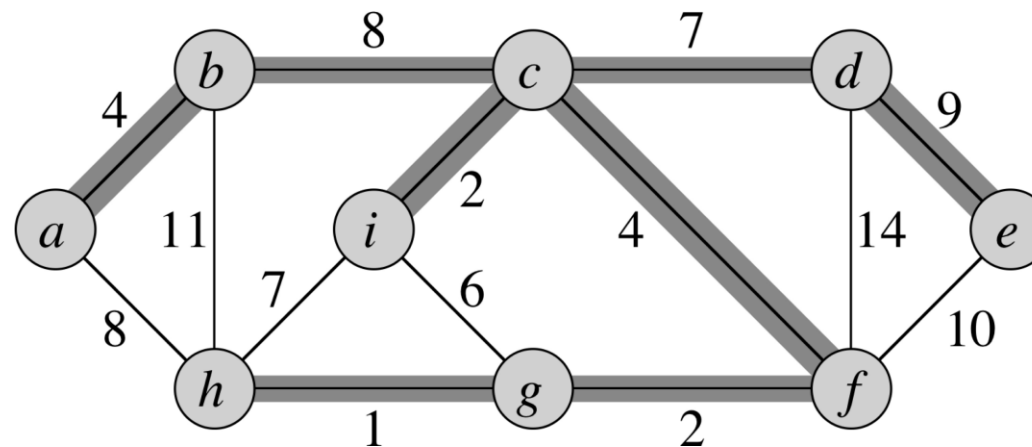


Figure 2: Minimum spanning tree

Representation

■ Construction Graph

We will consider the case 1-ANT

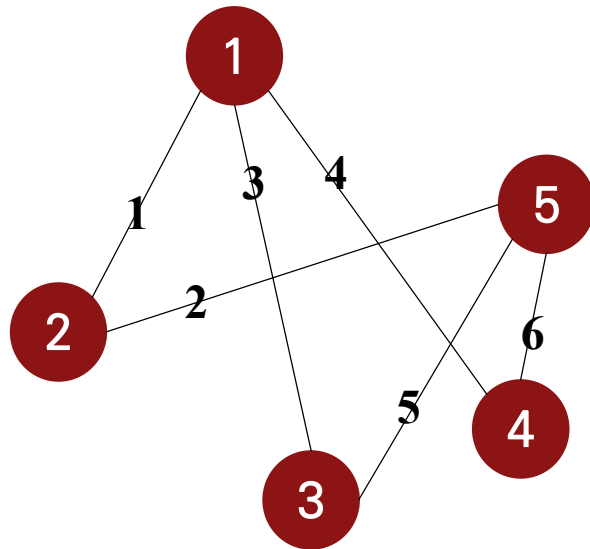
- the construction graph $C(G)$ for the MST problem
 - Node: $m + 1$ nodes $\{0, 1, \dots, m\} = E \cup \{s\}$ with the designated start node $s := 0$.
 - Edge: $A := \{(i, j) | 0 \leq i \leq m, 1 \leq j \leq m, i \neq j\}$
 - * $C(G)$ is obtained from the complete directed graph by removing all self-loops and the edges pointing to s
- If the “Ant” visits node e in $C(G)$, then we will construct a spanning tree using e .
- To ensure that a walk of the “Ant” actually construct a spanning tree, we define the feasible neighborhood $N(v_k)$ of node v_k in $C(G)$ depending on the nodes v_1, \dots, v_k visited so far:

$$N(v_k) := (E \setminus \{v_1, \dots, v_k\}) \setminus \{e \in E | (V, \{v_1, \dots, v_k, e\}) \text{ contains a cycle}\}$$

✓ Note that the feasible neighborhood depends on the memory of the ant about the path followed so far.

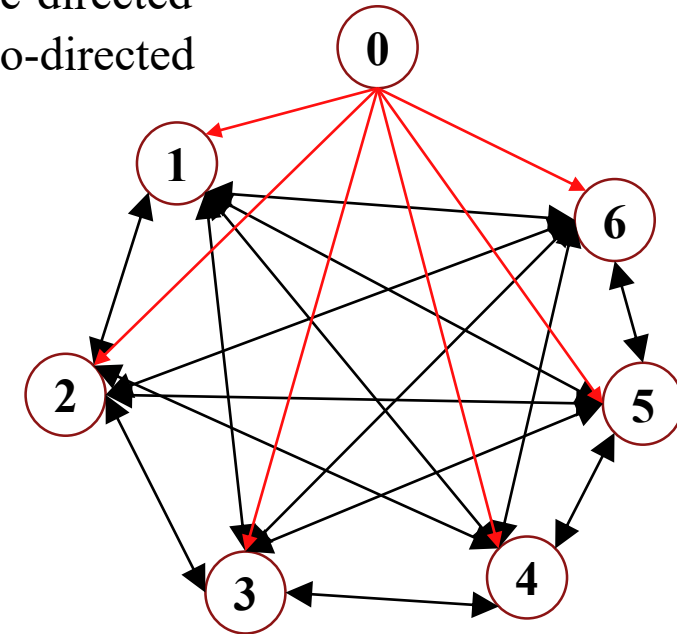
Representation

■ Construction Graph



undirected connected graph $G = (V, E)$

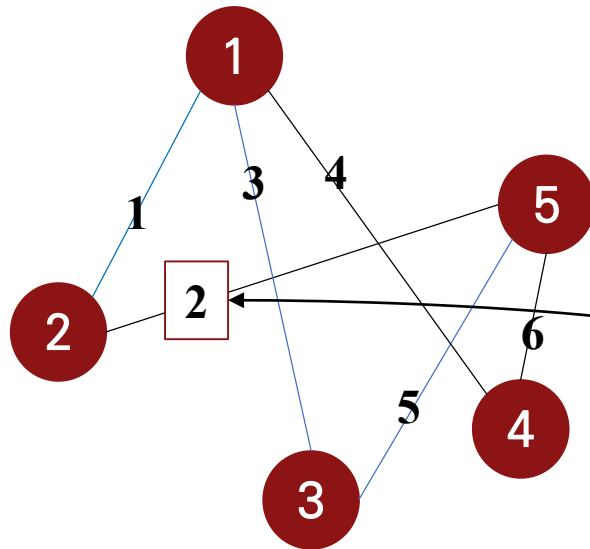
→ : one-directed
 ↔ : two-directed



Construction graph $C(G)$

Representation

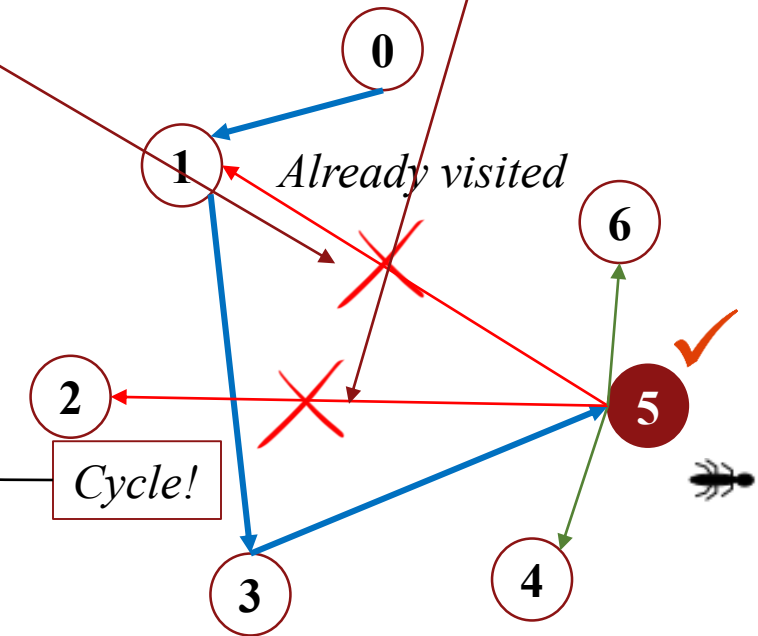
■ Construction Graph



undirected connected graph $G = (V, E)$

$$N(v_k) := (E \setminus \{v_1, \dots, v_k\}) \setminus \{e \in E \mid (V, \{v_1, \dots, v_k, e\}) \text{ contains a cycle}\}$$

feasible neighborhood $N(v_3 = 3) = \{(5,4), (5,6)\}$



→ : Path the ant crossed

→ : possible!

→ : impossible!

Construction graph $C(G)$

Representation

- Fitness $f(p)$ of path $p = (v_0, \dots, v_{n_1})$ generated by “the Ant”
 - the cost of the corresponding tree.
 - $f(p) = w(v_1) + w(v_2) + \dots + w(v_{n-1})$
- Update pheromone
 - after each update, the τ -value of each edge in $C(G)$ attains
either the upper bound h or lower bound l .
 - to allow the ant to rediscover the edges of the previous spanning tree equivalently in each order, we will update pheromone for all edges point to nodes from path p except s .

$$\text{The new pheromone values } \tau'_{(u,v)} = \begin{cases} (m - n + 1)(\log n)l & \text{if } v \in p \text{ and } v \neq s \\ l & \text{otherwise} \end{cases} \quad \text{for some } l \in \mathbb{R}$$

- Desirability values $\eta_{(u,v)}$
 - $\eta_{(u,v)}$ -value of edge (u, v) : the inverse of the weight of the edge of G corresponding to the node v in $C(G)$.

Pseudocode

Algorithm 1 1-ANT

set $\tau_{u,v} = 1/|A|$ for all $(u,v) \in A$.

Choose an arbitrary $x \in X$.

Set $x^* := x$.

while termination conditions not met **do**

 Using a construction graph, compute a solution x .

if $f(x) \leq f(x^*)$ **then**

 Update the pheromone values.

$x^* := x$

We will consider the case 1-ANT

Algorithm 2 Construct($C(G), \tau, \eta$)

$v_0 := s; k := 0$.

while $N(v_k) \neq \emptyset$ **do**

 Let $R := \sum_{y \in N(v_k)} [\tau_{(v_k,y)}]^\alpha [\eta_{(v_k,y)}]^\beta$.

 Choose neighbor $v_{k+1} \in N(v_k)$ with probability $\frac{[\tau_{(v_k,v_{k+1})}]^\alpha [\eta_{(v_k,v_{k+1})}]^\beta}{R}$.

$k := k + 1$.

return the path $p = (v_0, \dots, v_k)$.

Convergence behavior to an optimal solution

* the expected optimization time of the 1-ANT

To analysis the runtime behavior of this 1-ANT algorithm, we consider the expected number of solutions that a reconstructed by the algorithm until an MST has been obtained for the first time.

Theorem 1.

Choosing $\alpha = 1$ and $\beta = 0$, the expected optimization time of the 1-ANT with construction $C(G)$ is bounded by $O(mn(\log n + \log w_{\max}))$

Proof of Idea.

by showing that the probability of obtaining from the current tree T^* to $T = T^* \setminus \{e\} \cup \{e_0\}$ is lower bounded by $\Omega(\frac{1}{mn})$, We can conclude that

$$\text{"the expected optimization time"} \leq O(mn(\log n + \log w_{\max}))$$

Convergence behavior to an optimal solution

Proof.

First, we will show that for some tree T ,

T is constructed with probability $\Omega(1/(nm))$.

Let e_1, \dots, e_{n-1} be the edges of T^* and suppose that the edges of T are e_1, \dots, e_{n-2}, e' where $e' \neq e_i$ for $1 \leq i \leq n-1$.

With probability $\Omega(1)$, exactly $n-2$ (but not $n-1$) out of the $n-1$ nodes visited by the 1-ANT in $C(G)$ form a uniformly random subset of $\{e_1, \dots, e_{n-1}\}$.

therefore, e_{n-1} is missing with probability $\frac{1}{n-1}$.

Furthermore, the probability of visiting e_0 rather than e_{n-1} as the missing node has probability at least $\Omega(1/m)$. Hence, T is constructed with probability $\Omega(1/(nm))$.

Convergence behavior to an optimal solution

- Proof.

To prove theorem 1, it suffices to show the following.

Suppose the 1-ANT has constructed the spanning tree T^*
in the last accepted solution.

Let $T = (T^* \setminus \{e\}) \cup \{e_0\}$ be any spanning tree.

Then, the probability of producing T by the next constructed solution is $\Omega\left(\frac{1}{nm}\right)$.

Convergence behavior to an optimal solution

■ Proof.

To show this, define events E_i , $1 \leq i \leq n - 1$, as the first $i - 1$ and the last $n - i - 1$ nodes visited by the 1-ANT correspond to edges of T^* , whereas the i -th one doesn't.

Since edges in $C(G)$ pointing to nodes of T^* have pheromone value h and all remaining edges have value l ,


if $j - 1$ edges of T^* have been found, the probability of not choosing another edge of T^* by the next node visited in $C(G)$ is at most

$$\frac{(m - (n - 1))l}{((n - 1) - (j - 1))h} = \frac{1}{(n - j) \log n}$$

Convergence behavior to an optimal solution

■ Proof.

Therefore, the first $i - 1$ and last $n - i - 1$ nodes visited correspond to edges of T^* with probability at least

$$1 - \sum_{j=1}^{n-2} \frac{1}{(n-j) \log n} \geq 1 - \frac{\ln(n-1) + 1}{\log n} + \frac{1}{\log n} \geq 1 - \frac{\ln n}{\log n} = \Omega(1)$$


$(n-1)$ - th Harmonic number $H_{n-1} = \sum_{k=1}^{n-1} 1/k \approx \ln(n-1) + 1$

by using the symmetry of the update scheme, each subset of T^* of size $n - 2$ is equally likely. i.e. it has probability $\Omega(1/n)$.

Convergence behavior to an optimal solution

■ Proof.

Additionally, the probability of choosing by the $i - th$ visited node an edge e' not contained in T^* is

$$\frac{l}{(n - i)h + kl} \geq \frac{1}{(n - i + 1)(m - n + 1) \log n}$$

,where k is the number of edges outside T^* that can still be chosen.

Hence, with probability at least $\frac{c}{(n-i+1)mn \log n}$ for some small enough constant c , E_i occurs and the tree T is constructed.

Convergence behavior to an optimal solution

Therefore, since the E_i are mutually disjoint events, T is constructed instead of T^* with probability at least

$$\sum_{i=1}^{n-1} \frac{c}{(n-i+1)mn \log n} = \Omega(1/mn)$$



Convergence behavior to an optimal solution

Suppose $\beta \geq 6w_{max} \log n$ and $\alpha = 0$.

This method is similar to Kruskal algorithm by using only heuristic information η .

But, since the computational efforts in a run of the $C(G)$ and for initializing random number generators must not be neglected, this method doesn't improve upon Kruskal's algorithm.

Theorem 2.

Choosing $\alpha = 0$ and $\beta \geq 6w_{max} \log n$, the expected optimization time of the 1-ANT with construction $C(G)$ is constant.

Proof of Idea.

by showing that the probability of next solution that the 1-ANT constructs is at least $1/e$, where $e = \sum_{n=0}^{\infty} 1/n$,
We can conclude that

$$\text{"the expected optimization time"} = O(1)$$

It implies the expected number of solutions that have to be constructed from some spanning tree to a MST is bounded above by e .

Convergence behavior to an optimal solution

■ Proof

Let $(w_1, w_2, \dots, w_{n-1})$ the weights of edges of an MST and for $1 \leq i \leq n - 2$, w_i increases, i.e. $w_i \leq w_{i+1}$, and we will assume that the ant has already visited $i - 1$ edges that have weights w_1, \dots, w_{i-1} .

In this case, we can consider that $M = \{e_1, \dots, e_r\}$ is the set of edges that can be included without creating a cycle where $w(e_i) \leq w(e_{i+1})$ for $1 \leq i \leq r - 1$ and $M_i = \{e_1, \dots, e_s\}$ be the subset of M that includes all edges of weight w_i .

Convergence behavior to an optimal solution

■ Proof

The probability p of choosing an edge of M_i in the next step

$$p = \frac{\sum_{k=1}^s \eta(e_k)^\beta}{\sum_{l=1}^r \eta(e_l)^\beta} = \frac{\sum_{k=1}^s \eta(e_k)^\beta}{\sum_{l=1}^s \eta(e_l)^\beta + \sum_{l=1}^r \eta(e_l)^\beta} \text{ where } \eta(e_j) = \frac{1}{w(e_j)}.$$

Denote $\sum_{k=1}^s \eta(e_k)^\beta$, $\sum_{l=1}^r \eta(e_l)^\beta$ as a , b respectively.

$$\text{If } b \leq a/n, \text{ then } p = \frac{a}{a+b} \geq \frac{a}{a+\frac{a}{n}} = \frac{n}{n+1} = 1 - \frac{1}{n}.$$

Since the number of edges in $M \setminus M_i \leq m$ and

the weight of such an edge $\geq w_i + 1$, we have $b \leq m \left(\frac{1}{w_i + 1} \right)^\beta$

Convergence behavior to an optimal solution

■ Proof

To satisfy $m \left(\frac{1}{w_i+1} \right)^\beta \leq \frac{s}{n} \left(\frac{1}{w_i} \right)^\beta$ ($\leftarrow b \leq a/n$),

we can select β such that $\beta \geq \frac{\log\left(\frac{mn}{s}\right)}{\log\left(\frac{w_i+1}{w_i}\right)} = \frac{\log\left(\frac{mn}{s}\right)}{\log\left(1+\frac{1}{w_i}\right)}$

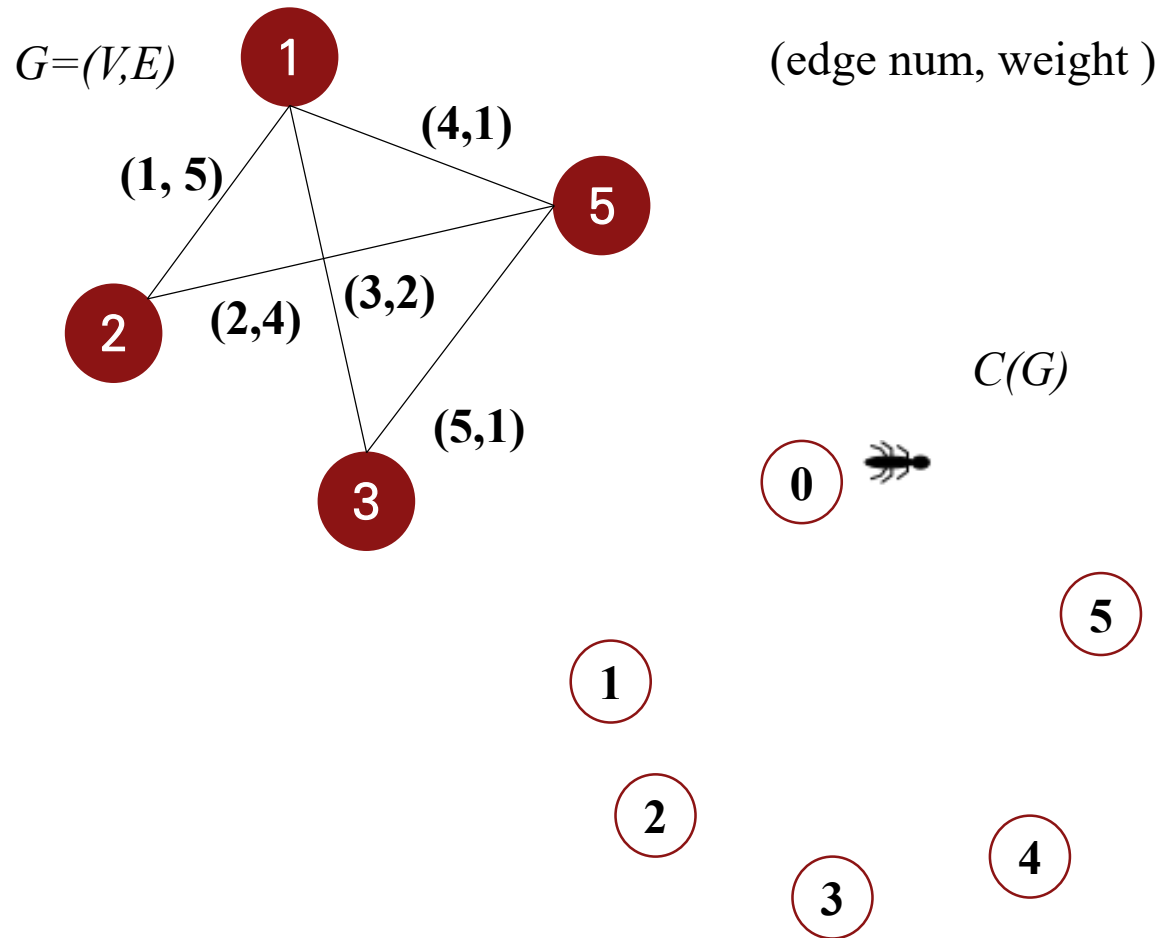
which is at most $\frac{\log\left(\frac{mn}{s}\right)}{\frac{1}{2w_i}} \leq 6w_{max} \log n$

since $mn \leq n^3$ and $e^x < 1 + 2x$ for $0 \leq x \leq 1$.

By choosing $\beta \geq 6w_{max} \log n$, the ant traverses the edge with w_i with probability $1 - \frac{1}{n}$.

Therefore, the prob. That in every step i such an edge is taken at least $\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{e}$ \square

Toy problem



Algorithm 1 1-ANT

set $\tau_{u,v} = 1/|A|$ for all $(u,v) \in A$.

Choose an arbitrary $x \in X$.

Set $x^* := x$.

while termination conditions not met **do**

Using a construction graph, compute a solution x .

if $f(x) \leq f(x^*)$ **then**

Update the pheromone values.

$x^* := x$

Algorithm 2 Construct($C(G), \tau, \eta$)

$v_0 := s; k := 0$.

while $N(v_k) \neq \emptyset$ **do**

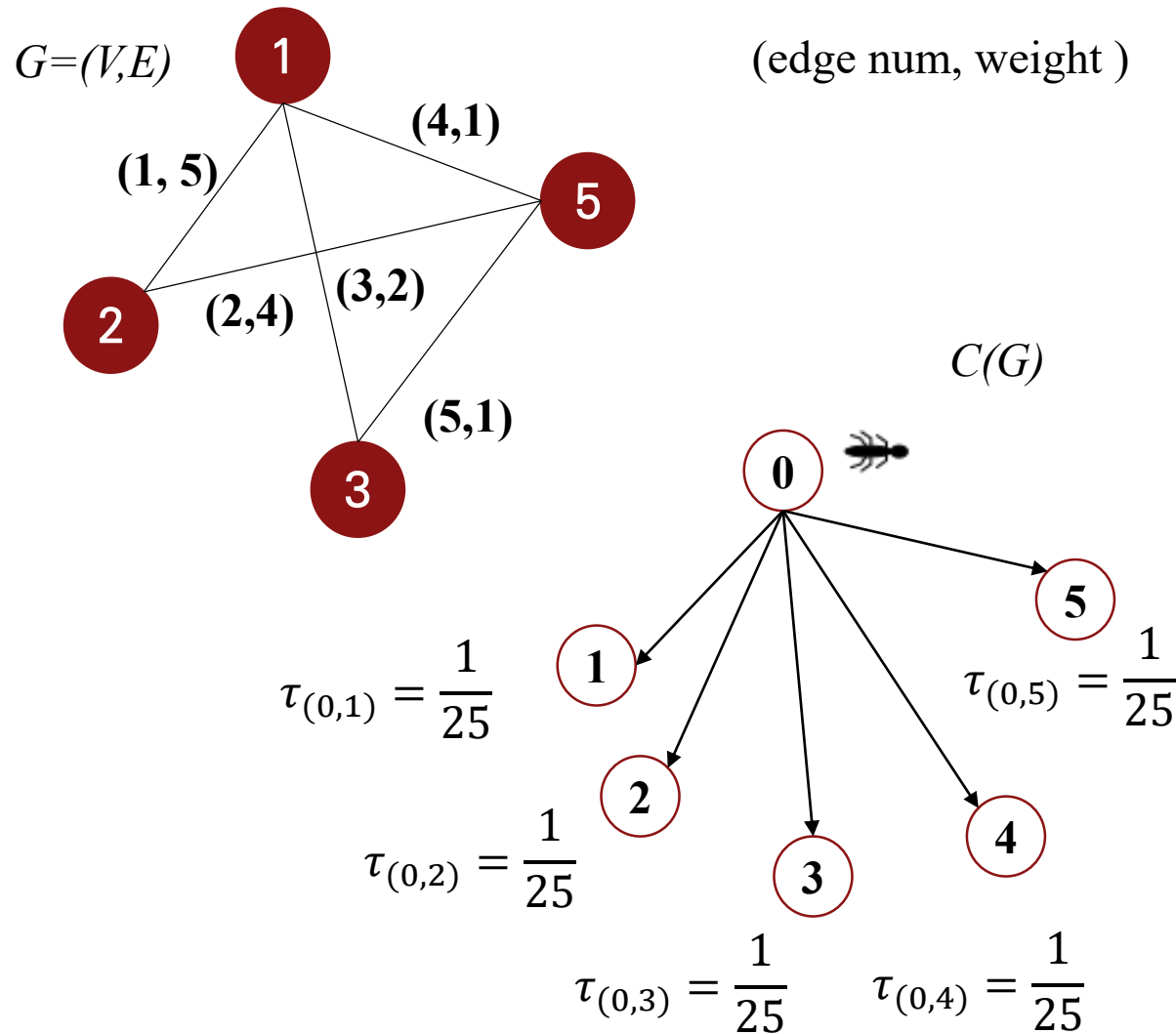
Let $R := \sum_{y \in N(v_k)} [\tau_{(v_k,y)}]^\alpha [\eta_{(v_k,y)}]^\beta$.

Choose neighbor $v_{k+1} \in N(v_k)$ with probability $\frac{[\tau_{(v_k,v_{k+1})}]^\alpha [\eta_{(v_k,v_{k+1})}]^\beta}{R}$.

$k := k + 1$.

return the path $p = (v_0, \dots, v_k)$.

Toy problem



$$\alpha = 1, \beta = 0$$

Algorithm 1 1-ANT

set $\tau_{u,v} = 1/|A|$ for all $(u,v) \in A$.

Choose an arbitrary $x \in X$.

Set $x^* := x$.

while termination conditions not met **do**

Using a construction graph, compute a solution x .

if $f(x) \leq f(x^*)$ **then**

Update the pheromone values.

$x^* := x$

$$R = \sum_{y \in N(v_k)} \frac{1}{|A|} = \frac{5}{25}$$

Algorithm 2 Construct($C(G), \tau, \eta$)

$v_0 := s; k := 0$.

while $N(v_k) \neq \emptyset$ **do**

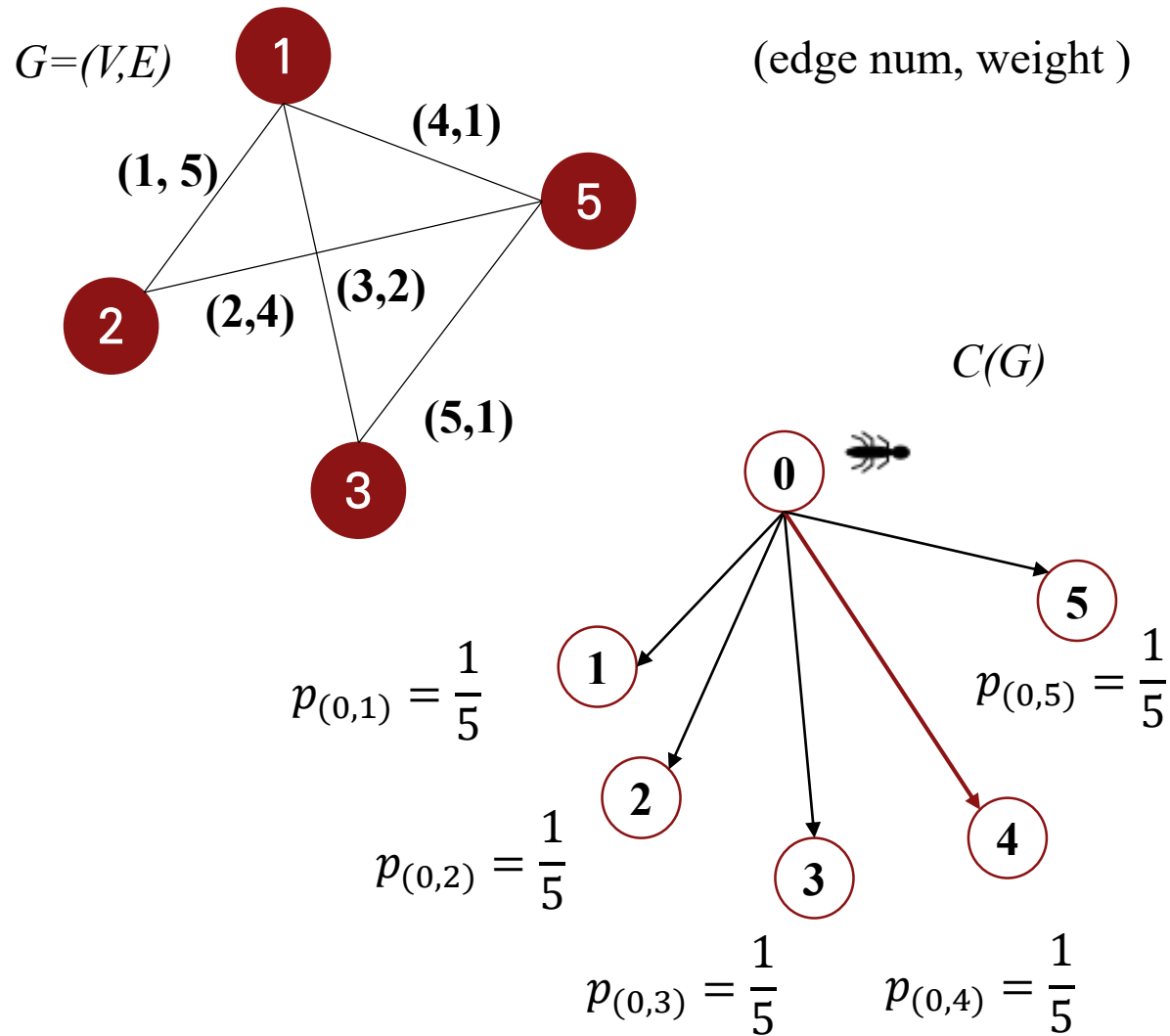
Let $R := \sum_{y \in N(v_k)} [\tau_{(v_k,y)}]^\alpha [\eta_{(v_k,y)}]^\beta$.

Choose neighbor $v_{k+1} \in N(v_k)$ with probability $\frac{[\tau_{(v_k,v_{k+1})}]^\alpha [\eta_{(v_k,v_{k+1})}]^\beta}{R}$.

$k := k + 1$.

return the path $p = (v_0, \dots, v_k)$.

Toy problem



$$\alpha = 1, \beta = 0$$

Algorithm 1 1-ANT

set $\tau_{u,v} = 1/|A|$ for all $(u,v) \in A$.

Choose an arbitrary $x \in X$.

Set $x^* := x$.

while termination conditions not met **do**

Using a construction graph, compute a solution x .

if $f(x) \leq f(x^*)$ **then**

Update the pheromone values.

$x^* := x$

$$R = \sum_{y \in N(v_k)} \frac{1}{|A|} = \frac{5}{25}$$

Algorithm 2 Construct($C(G), \tau, \eta$)

$v_0 := s; k := 0$.

while $N(v_k) \neq \emptyset$ **do**

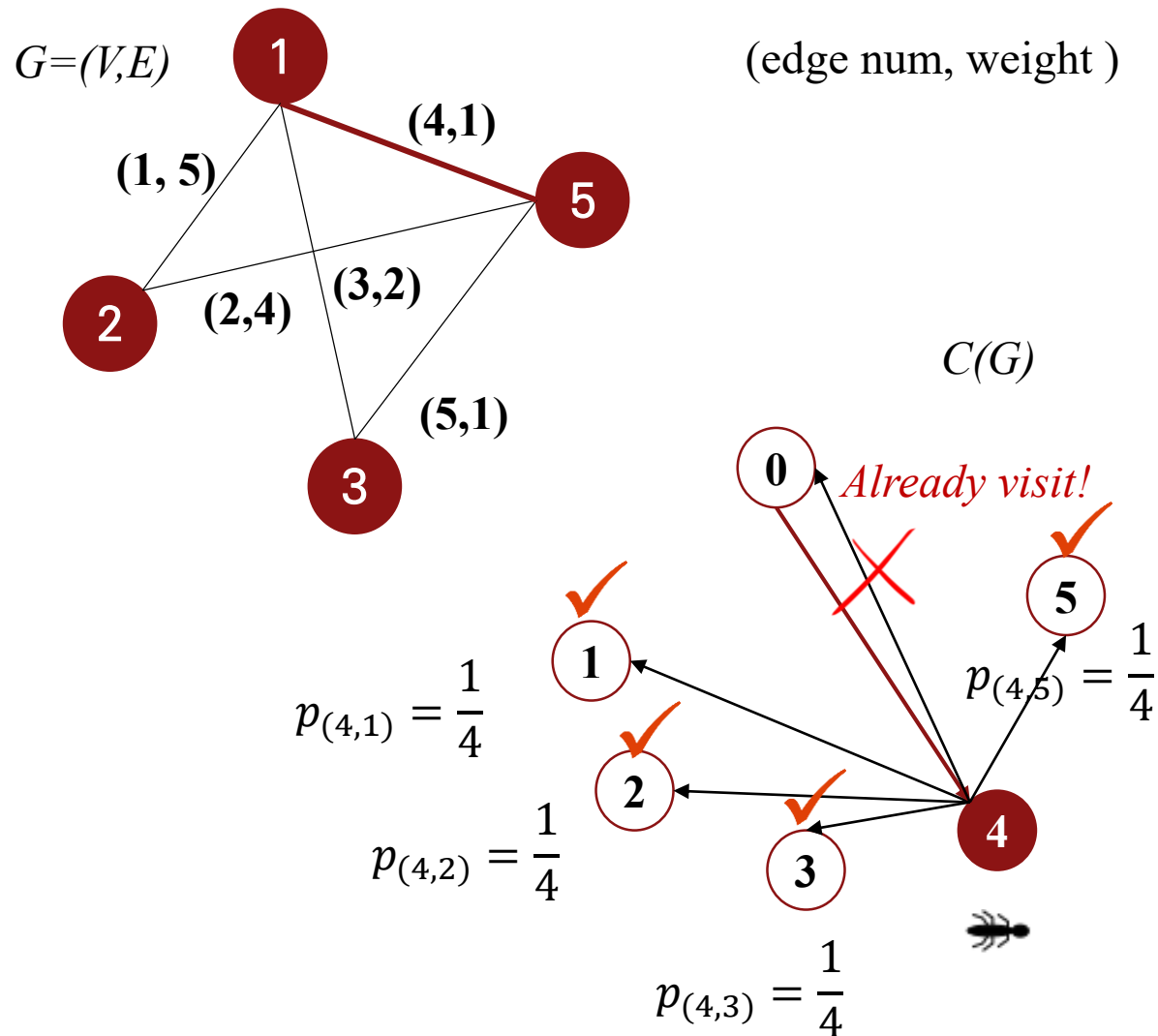
Let $R := \sum_{y \in N(v_k)} [\tau_{(v_k,y)}]^\alpha [\eta_{(v_k,y)}]^\beta$.

Choose neighbor $v_{k+1} \in N(v_k)$ with probability $\frac{[\tau_{(v_k,v_{k+1})}]^\alpha [\eta_{(v_k,v_{k+1})}]^\beta}{R}$.

$k := k + 1$.

return the path $p = (v_0, \dots, v_k)$.

Toy problem



— : selected edge

$$\alpha = 1, \beta = 0$$

Algorithm 1 1-ANT

set $\tau_{u,v} = 1/|A|$ for all $(u,v) \in A$.

Choose an arbitrary $x \in X$.

Set $x^* := x$.

while termination conditions not met **do**

Using a construction graph, compute a solution x .

if $f(x) \leq f(x^*)$ **then**

Update the pheromone values.

$x^* := x$

✓ :feasible neighbors

$$R = \sum_{y \in N(v_k)} \frac{1}{|A|} = \frac{4}{25}$$

Algorithm 2 Construct($C(G), \tau, \eta$)

$v_0 := s; k := 0$.

while $N(v_k) \neq \emptyset$ **do**

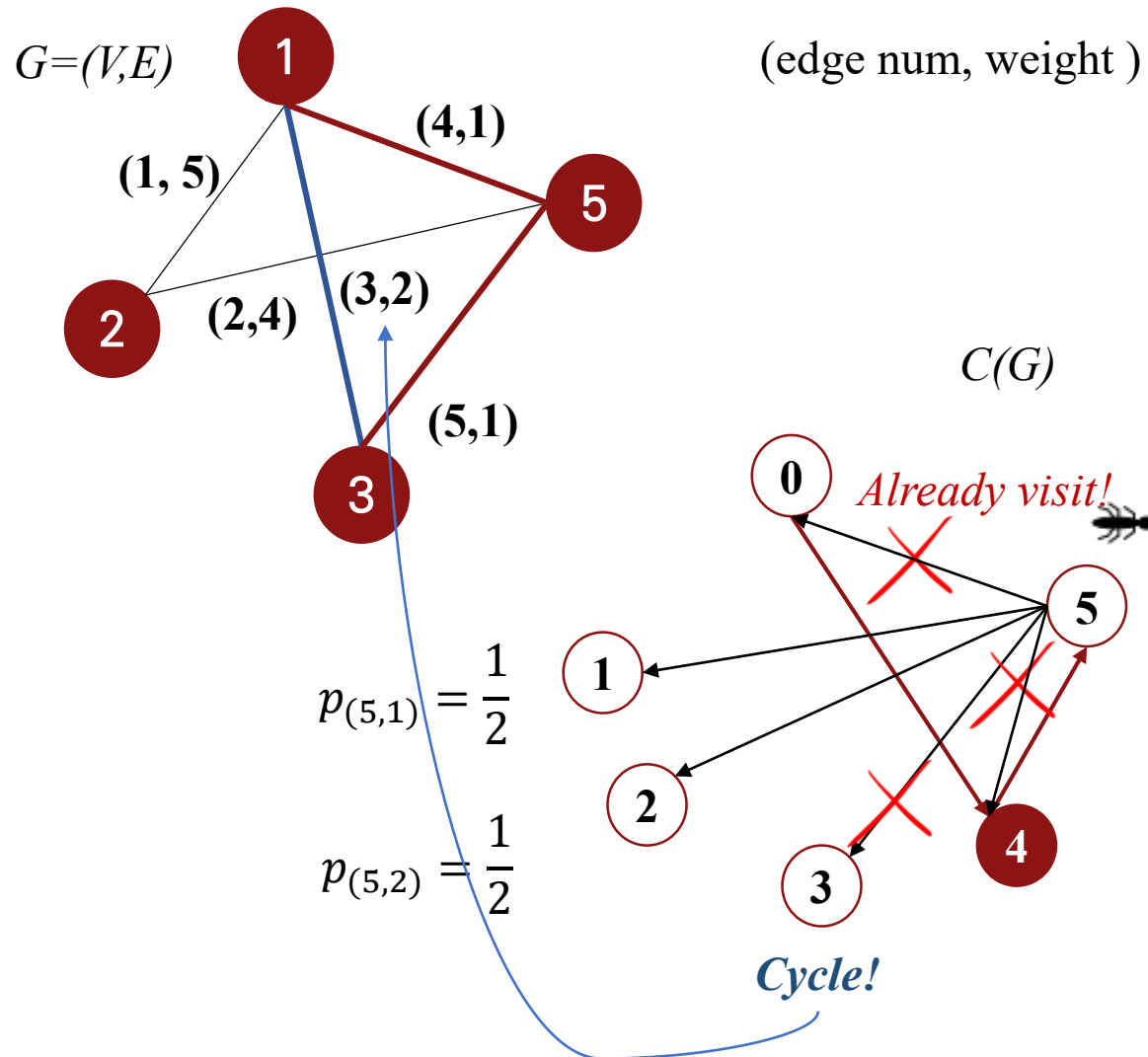
Let $R := \sum_{y \in N(v_k)} [\tau_{(v_k,y)}]^\alpha [\eta_{(v_k,y)}]^\beta$.

Choose neighbor $v_{k+1} \in N(v_k)$ with probability $\frac{[\tau_{(v_k,v_{k+1})}]^\alpha [\eta_{(v_k,v_{k+1})}]^\beta}{R}$.

$k := k + 1$.

return the path $p = (v_0, \dots, v_k)$.

Toy problem



$$\alpha = 1, \beta = 0$$

Algorithm 1 1-ANT

set $\tau_{u,v} = 1/|A|$ for all $(u,v) \in A$.

Choose an arbitrary $x \in X$.

Set $x^* := x$.

while termination conditions not met **do**

Using a construction graph, compute a solution x .

if $f(x) \leq f(x^*)$ **then**

Update the pheromone values.

$x^* := x$

$$R = \sum_{y \in N(v_k)} \frac{1}{|A|} = \frac{2}{25}$$

Algorithm 2 Construct($C(G), \tau, \eta$)

$v_0 := s; k := 0$.

while $N(v_k) \neq \emptyset$ **do**

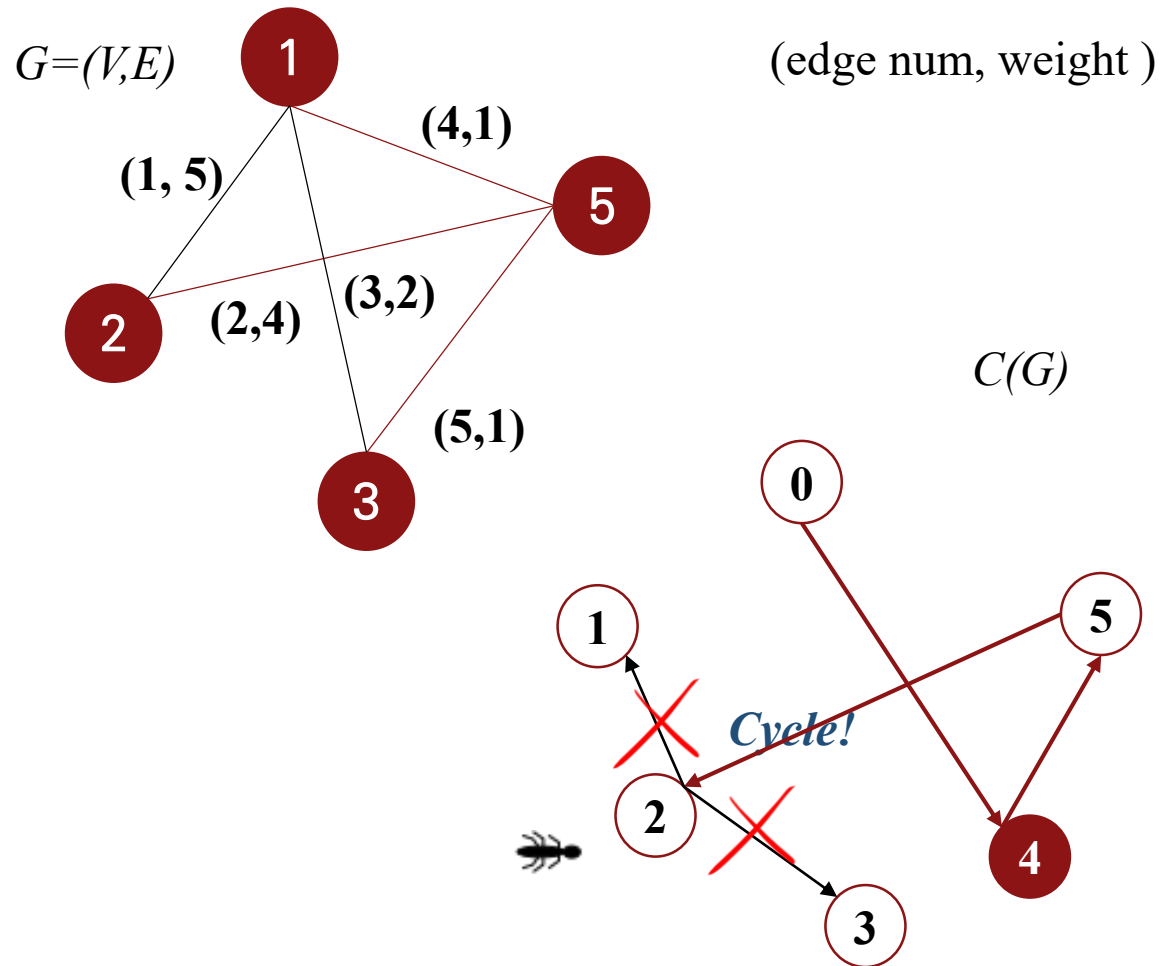
Let $R := \sum_{y \in N(v_k)} [\tau_{(v_k,y)}]^\alpha [\eta_{(v_k,y)}]^\beta$.

Choose neighbor $v_{k+1} \in N(v_k)$ with probability $\frac{[\tau_{(v_k,v_{k+1})}]^\alpha [\eta_{(v_k,v_{k+1})}]^\beta}{R}$.

$k := k + 1$.

return the path $p = (v_0, \dots, v_k)$.

Toy problem



$$\alpha = 1, \beta = 0$$

Algorithm 1 1-ANT

set $\tau_{u,v} = 1/|A|$ for all $(u,v) \in A$.

Choose an arbitrary $x \in X$.

Set $x^* := x$.

while termination conditions not met **do**

Using a construction graph, compute a solution x .

if $f(x) \leq f(x^*)$ **then**

Update the pheromone values.

$x^* := x$

Algorithm 2 Construct($C(G), \tau, \eta$)

$v_0 := s; k := 0$.

while $N(v_k) \neq \emptyset$ **do**

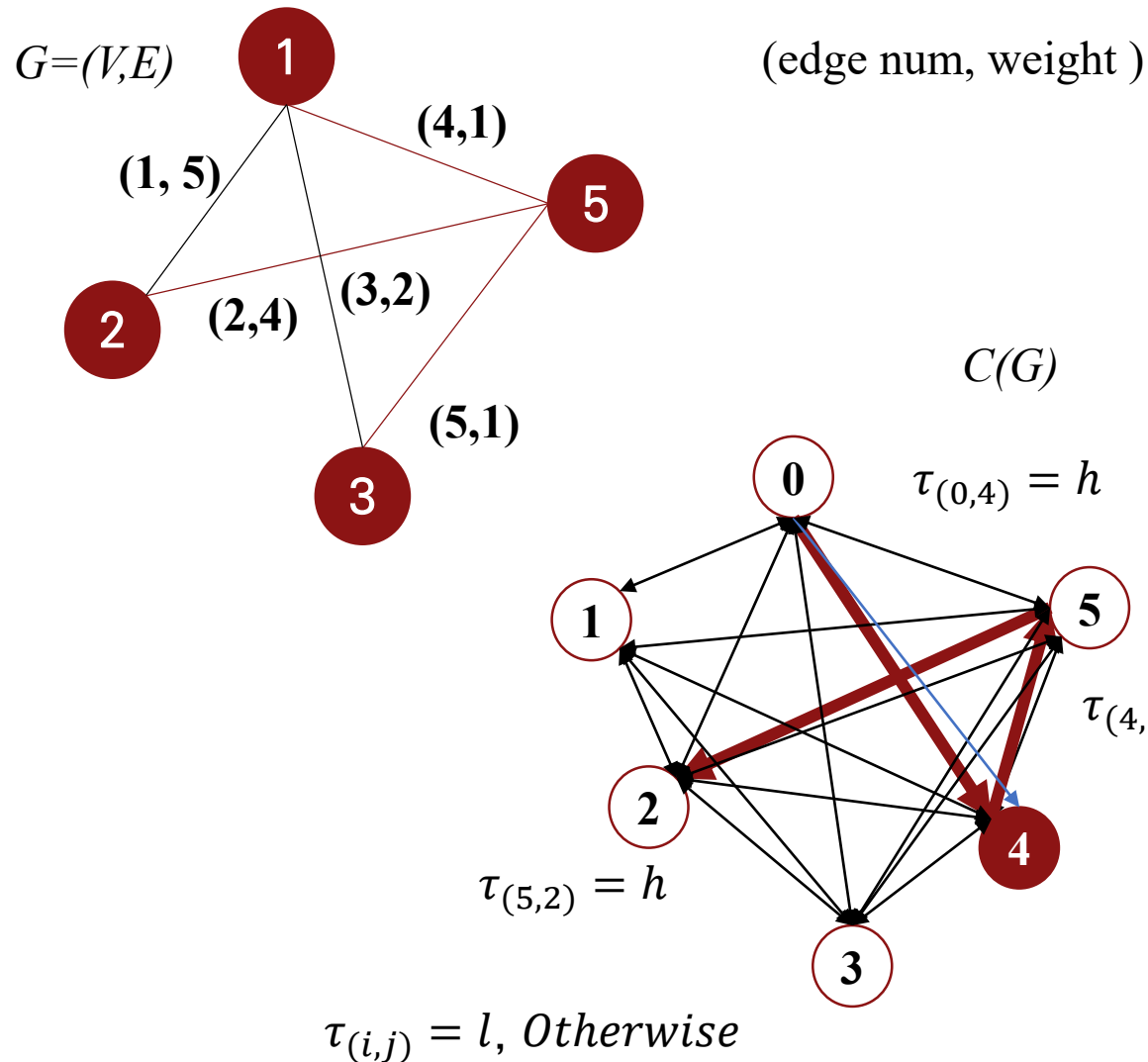
Let $R := \sum_{y \in N(v_k)} [\tau_{(v_k,y)}]^\alpha [\eta_{(v_k,y)}]^\beta$.

Choose neighbor $v_{k+1} \in N(v_k)$ with probability $\frac{[\tau_{(v_k,v_{k+1})}]^\alpha [\eta_{(v_k,v_{k+1})}]^\beta}{R}$.

$k := k + 1$.

return the path $p = (v_0, \dots, v_k)$.

Toy problem



$$\alpha = 1, \beta = 0$$

Algorithm 1 1-ANT

set $\tau_{u,v} = 1/|A|$ for all $(u,v) \in A$.

Choose an arbitrary $x \in X$.

Set $x^* := x$.

while termination conditions not met **do**

 Using a construction graph, compute a solution x .

if $f(x) \leq f(x^*)$ **then**

 Update the pheromone values.

$x^* := x$

$\rightarrow : \tau_e = h$

Algorithm 2 Construct($C(G), \tau, \eta$)

$v_0 := s; k := 0$.

while $N(v_k) \neq \emptyset$ **do**

 Let $R := \sum_{y \in N(v_k)} [\tau_{(v_k,y)}]^\alpha [\eta_{(v_k,y)}]^\beta$.

 Choose neighbor $v_{k+1} \in N(v_k)$ with probability $\frac{[\tau_{(v_k,v_{k+1})}]^\alpha [\eta_{(v_k,v_{k+1})}]^\beta}{R}$.

$k := k + 1$.

return the path $p = (v_0, \dots, v_k)$.

References

- Neumann, Frank, and Carsten Witt. "Ant colony optimization and the minimum spanning tree problem." *Theoretical Computer Science* 411.25 (2010): 2406-2413.