Ant Colony Optimization and the Minimum Cut Problem

Timo Kötzing, Per Kristian Lehre, Frank Neumann, Pietro S. Oliveto

March 25, 2010
We want to analyze the use of Ant Colony Optimization (ACO) for the Minimum Cut Problem.

As input, the ACO algorithm gets a weighted undirected graph $G$ on $n$ vertices.

The ACO algorithm iteratively computes partitions of $G$'s vertices into two non-empty sets, one per iteration.

The algorithm keeps track of the best so far candidate solution.

We analyze the random variable of the number of iterations required until an optimal solution is found.
We want to analyze the use of Ant Colony Optimization (ACO) for the Minimum Cut Problem.

As input, the ACO algorithm gets a weighted undirected graph $G$ on $n$ vertices.

The ACO algorithm iteratively computes partitions of $G$’s vertices into two non-empty sets, one per iteration.

The algorithm keeps track of the best so far candidate solution.

We analyze the random variable of the number of iterations required until an optimal solution is found.
We want to analyze the use of Ant Colony Optimization (ACO) for the Minimum Cut Problem.

As input, the ACO algorithm gets a weighted undirected graph \( G \) on \( n \) vertices.

The ACO algorithm iteratively computes partitions of \( G \)'s vertices into two non-empty sets, one per iteration.

The algorithm keeps track of the best so far candidate solution.

We analyze the random variable of the number of iterations required until an optimal solution is found.
We want to analyze the use of Ant Colony Optimization (ACO) for the Minimum Cut Problem.

As input, the ACO algorithm gets a weighted undirected graph \( G \) on \( n \) vertices.

The ACO algorithm iteratively computes partitions of \( G \)'s vertices into two non-empty sets, one per iteration.

The algorithm keeps track of the best so far candidate solution.

We analyze the random variable of the number of iterations required until an optimal solution is found.
We want to analyze the use of Ant Colony Optimization (ACO) for the Minimum Cut Problem. As input, the ACO algorithm gets an weighted undirected graph $G$ on $n$ vertices. The ACO algorithm iteratively computes partitions of $G$’s vertices into two non-empty sets, one per iteration. The algorithm keeps track of the best so far candidate solution. We analyze the random variable of the number of iterations required until an optimal solution is found.
We want to analyze the use of Ant Colony Optimization (ACO) for the Minimum Cut Problem.

As input, the ACO algorithm gets a weighted undirected graph $G$ on $n$ vertices.

The ACO algorithm iteratively computes partitions of $G$’s vertices into two non-empty sets, one per iteration.

The algorithm keeps track of the best so far candidate solution.

We analyze the random variable of the number of iterations required until an optimal solution is found.
Idea for Constructing Solutions (Karger and Stein):

Any forest of \( n - 2 \) edges constitutes a partition into two sets (the sets of vertices of the two trees).

Karger and Stein give an algorithm with expected runtime \( O(n^2) \).

Our ACO algorithm lets ants choose (sequentially) \( n - 2 \) edges to build candidate solutions (without creating cycles).

The probability for an edge \( e \) to be picked depends on two value associated with that edge:

- its weight \( w(e) \); and
- the pheromone value \( \tau_e \) on \( e \).
Idea for Constructing Solutions (Karger and Stein):

- Any forest of \( n - 2 \) edges constitutes a partition into two sets (the sets of vertices of the two trees).
- Karger and Stein give an algorithm with expected runtime \( O(n^2) \).
- Our ACO algorithm lets ants choose (sequentially) \( n - 2 \) edges to build candidate solutions (without creating cycles).
- The probability for an edge \( e \) to be picked depends on two value associated with that edge:
  - its weight \( w(e) \); and
  - the pheromone value \( \tau_e \) on \( e \).
Idea for Constructing Solutions (Karger and Stein):

- Any forest of \( n - 2 \) edges constitutes a partition into two sets (the sets of vertices of the two trees).
- Karger and Stein give an algorithm with expected runtime \( O(n^2) \).
- Our ACO algorithm lets ants choose (sequentially) \( n - 2 \) edges to build candidate solutions (without creating cycles).
- The probability for an edge \( e \) to be picked depends on two values associated with that edge:
  - its weight \( w(e) \); and
  - the pheromone value \( \tau_e \) on \( e \).
Idea for Constructing Solutions (Karger and Stein):

- Any forest of \( n - 2 \) edges constitutes a partition into two sets (the sets of vertices of the two trees).
- Karger and Stein give an algorithm with expected runtime \( O(n^2) \).
- Our ACO algorithm lets ants choose (sequentially) \( n - 2 \) edges to build candidate solutions (without creating cycles).
- The probability for an edge \( e \) to be picked depends on two value associated with that edge:
  - its weight \( w(e) \); and
  - the pheromone value \( \tau_e \) on \( e \).
Idea for Constructing Solutions

- **Idea** for Constructing Solutions (Karger and Stein):
- Any forest of $n - 2$ edges constitutes a partition into two sets (the sets of vertices of the two trees).
- Karger and Stein give an algorithm with expected runtime $O(n^2)$.
- Our ACO algorithm lets ants choose (sequentially) $n - 2$ edges to build candidate solutions (without creating cycles).
- The probability for an edge $e$ to be picked depends on two value associated with that edge:
  - its weight $w(e)$; and
  - the pheromone value $\tau_e$ on $e$. 
Idea for Constructing Solutions

- **Idea** for Constructing Solutions (Karger and Stein):
- Any forest of $n - 2$ edges constitutes a partition into two sets (the sets of vertices of the two trees).
- Karger and Stein give an algorithm with expected runtime $O(n^2)$.
- Our ACO algorithm lets ants choose (sequentially) $n - 2$ edges to build candidate solutions (without creating cycles).
- The probability for an edge $e$ to be picked depends on two value associated with that edge:
  - its weight $w(e)$; and
  - the pheromone value $\tau_e$ on $e$. 
Idea for Constructing Solutions (Karger and Stein):

- Any forest of $n - 2$ edges constitutes a partition into two sets (the sets of vertices of the two trees).
- Karger and Stein give an algorithm with expected runtime $O(n^2)$.
- Our ACO algorithm lets ants choose (sequentially) $n - 2$ edges to build candidate solutions (without creating cycles).
- The probability for an edge $e$ to be picked depends on two value associated with that edge:
  - its weight $w(e)$; and
  - the pheromone value $\tau_e$ on $e$. 
Idea for Constructing Solutions (Karger and Stein):

Any forest of $n - 2$ edges constitutes a partition into two sets (the sets of vertices of the two trees).

Karger and Stein give an algorithm with expected runtime $O(n^2)$.

Our ACO algorithm lets ants choose (sequentially) $n - 2$ edges to build candidate solutions (without creating cycles).

The probability for an edge $e$ to be picked depends on two value associated with that edge:

- its weight $w(e)$; and
- the pheromone value $\tau_e$ on $e$. 
Pheromones

- Pheromones are additional information on the edges.
- A higher pheromone value on an edge $e$ means that $e$ is more likely to be chosen for the next solution.
- Initially, all pheromone values are the same.
- After that, the pheromone value of an edge $e$ that is used in the best-so-far solution has a pheromone value $h$.
- All others have a pheromone value $l$.
Pheromones are additional information on the edges.

A higher pheromone value on an edge $e$ means that $e$ is more likely to be chosen for the next solution.

Initially, all pheromone values are the same.

After that, the pheromone value of an edge $e$ that is used in the best-so-far solution has a pheromone value $h$.

All others have a pheromone value $l$. 
Pheromones

- Pheromones are additional information on the edges.
- A higher pheromone value on an edge $e$ means that $e$ is more likely to be chosen for the next solution.
- Initially, all pheromone values are the same.
- After that, the pheromone value of an edge $e$ that is used in the best-so-far solution has a pheromone value $h$.
- All others have a pheromone value $l$. 
Pheromones

- Pheromones are additional information on the edges.
- A higher pheromone value on an edge $e$ means that $e$ is more likely to be chosen for the next solution.
- Initially, all pheromone values are the same.
- After that, the pheromone value of an edge $e$ that is used in the best-so-far solution has a pheromone value $h$.
- All others have a pheromone value $l$. 
Pheromones are additional information on the edges.

A higher pheromone value on an edge $e$ means that $e$ is more likely to be chosen for the next solution.

Initially, all pheromone values are the same.

After that, the pheromone value of an edge $e$ that is used in the best-so-far solution has a pheromone value $h$.

All others have a pheromone value $l$. 
Pheromones

- **Pheromones** are additional information on the edges.
- A higher pheromone value on an edge $e$ means that $e$ is more likely to be chosen for the next solution.
- Initially, all pheromone values are the same.
- After that, the pheromone value of an edge $e$ that is used in the best-so-far solution has a pheromone value $h$.
- All others have a pheromone value $l$. 
Heuristic Information vs. Pheromone Values

- Remember: The probability for an edge $e$ to be picked depends on the two values $w(e)$ and $\tau_e$.
- How do we balance these two values?
- We use two parameters, $\alpha$ and $\beta$.
- For an edge $e$ with associated pheromone value $\tau_e$ and weight $w(e)$, the ant chooses $e$ proportionally to $\tau_e^\alpha \cdot (w(e))^\beta$. 
Heuristic Information vs. Pheromone Values

- Remember: The probability for an edge $e$ to be picked depends on the two values $w(e)$ and $\tau_e$.
- How do we balance these two values?
- We use two parameters, $\alpha$ and $\beta$.
- For an edge $e$ with associated pheromone value $\tau_e$ and weight $w(e)$, the ant chooses $e$ proportionally to

$$\tau_e^\alpha \cdot (w(e))^\beta.$$
Remember: The probability for an edge $e$ to be picked depends on the two values $w(e)$ and $\tau_e$.

How do we balance these two values?

We use two parameters, $\alpha$ and $\beta$.

For an edge $e$ with associated pheromone value $\tau_e$ and weight $w(e)$, the ant chooses $e$ proportionally to

$$\tau_e^\alpha \cdot (w(e))^\beta.$$
Heuristic Information vs. Pheromone Values

- Remember: The probability for an edge $e$ to be picked depends on the two values $w(e)$ and $\tau_e$.
- How do we balance these two values?
- We use two parameters, $\alpha$ and $\beta$.
- For an edge $e$ with associated pheromone value $\tau_e$ and weight $w(e)$, the ant chooses $e$ proportionally to

$$\tau_e^\alpha \cdot (w(e))^\beta.$$
Remember: The probability for an edge $e$ to be picked depends on the two values $w(e)$ and $\tau_e$.

How do we balance these two values?

We use two parameters, $\alpha$ and $\beta$.

For an edge $e$ with associated pheromone value $\tau_e$ and weight $w(e)$, the ant chooses $e$ proportionally to

$$\tau_e^\alpha \cdot (w(e))^{\beta}.$$
Heuristic Information vs. Pheromone Values

- Remember: The probability for an edge $e$ to be picked depends on the two values $w(e)$ and $\tau_e$.
- How do we balance these two values?
- We use two parameters, $\alpha$ and $\beta$.
- For an edge $e$ with associated pheromone value $\tau_e$ and weight $w(e)$, the ant chooses $e$ proportionally to
  \[ \tau_e^{\alpha} \cdot (w(e))^{\beta}. \]
$\tau_e^\alpha \cdot \left( w(e) \right)^\beta$

We proved the following expected optimization times of ACO.

- If $\alpha = 0$ and $\beta = 1$ (greedy only), $O(n^2)$.
- If $\alpha = 1$ and $\beta = 1$ with constant pheromone bounds, times are still polynomially bounded.
- If $\alpha = 1$ and $\beta = 1$ with at least linear pheromone bound ratio, times are not polynomially bounded.
- If $\beta > 1$, times are not polynomially bounded.
- If $\alpha = 1$ and $\beta = 0$ for sensible pheromone bounds, times are gain not polynomially bounded.
We proved the following expected optimization times of ACO.

- If $\alpha = 0$ and $\beta = 1$ (greedy only), $O(n^2)$.
- If $\alpha = 1$ and $\beta = 1$ with constant pheromone bounds, times are still polynomially bounded.
- If $\alpha = 1$ and $\beta = 1$ with at least linear pheromone bound ratio, times are not polynomially bounded.
- If $\beta > 1$, times are not polynomially bounded.
- If $\alpha = 1$ and $\beta = 0$ for sensible pheromone bounds, times are gain not polynomially bounded.
\[ \tau_e^\alpha \cdot (w(e))^{\beta} \]

We proved the following expected optimization times of ACO.

- If \( \alpha = 0 \) and \( \beta = 1 \) (greedy only), \( O(n^2) \).
- If \( \alpha = 1 \) and \( \beta = 1 \) with constant pheromone bounds, times are still polynomially bounded.
- If \( \alpha = 1 \) and \( \beta = 1 \) with at least linear pheromone bound ratio, times are not polynomially bounded.
- If \( \beta > 1 \), times are not polynomially bounded.
- If \( \alpha = 1 \) and \( \beta = 0 \) for sensible pheromone bounds, times are gain not polynomially bounded.
\[ \tau_e^\alpha \cdot (w(e))^{\beta} \]

We proved the following expected optimization times of ACO.

- If \( \alpha = 0 \) and \( \beta = 1 \) (greedy only), \( O(n^2) \).
- If \( \alpha = 1 \) and \( \beta = 1 \) with constant pheromone bounds, times are still polynomially bounded.
- If \( \alpha = 1 \) and \( \beta = 1 \) with at least linear pheromone bound ratio, times are not polynomially bounded.
- If \( \beta > 1 \), times are not polynomially bounded.
- If \( \alpha = 1 \) and \( \beta = 0 \) for sensible pheromone bounds, times are gain not polynomially bounded.
Results

\[
\tau_e^\alpha \cdot (w(e))^{\beta}
\]

We proved the following expected optimization times of ACO.

- If \( \alpha = 0 \) and \( \beta = 1 \) (greedy only), \( O(n^2) \).
- If \( \alpha = 1 \) and \( \beta = 1 \) with constant pheromone bounds, times are still polynomially bounded.
- If \( \alpha = 1 \) and \( \beta = 1 \) with at least linear pheromone bound ratio, times are not polynomially bounded.
- If \( \beta > 1 \), times are not polynomially bounded.
- If \( \alpha = 1 \) and \( \beta = 0 \) for sensible pheromone bounds, times are gain not polynomially bounded.
We proved the following expected optimization times of ACO.

- If $\alpha = 0$ and $\beta = 1$ (greedy only), $O(n^2)$.
- If $\alpha = 1$ and $\beta = 1$ with constant pheromone bounds, times are still polynomially bounded.
- If $\alpha = 1$ and $\beta = 1$ with at least linear pheromone bound ratio, times are not polynomially bounded.
- If $\beta > 1$, times are not polynomially bounded.
- If $\alpha = 1$ and $\beta = 0$ for sensible pheromone bounds, times are gain not polynomially bounded.
\[ \tau_e^\alpha \cdot (w(e))^\beta \]

We proved the following expected optimization times of ACO.

- If \( \alpha = 0 \) and \( \beta = 1 \) (greedy only), \( O(n^2) \).
- If \( \alpha = 1 \) and \( \beta = 1 \) with constant pheromone bounds, times are still polynomially bounded.
- If \( \alpha = 1 \) and \( \beta = 1 \) with at least linear pheromone bound ratio, times are not polynomially bounded.
- If \( \beta > 1 \), times are not polynomially bounded.
- If \( \alpha = 1 \) and \( \beta = 0 \) for sensible pheromone bounds, times are gain not polynomially bounded.
Conclusions

- Don’t use an ACO algorithm to solve the Min-Cut Problem.
- ACO can simulate Karger and Stein’s algorithm.
- We now understand better how ACO algorithms work.
- We now understand better how to analyze ACO algorithms.
Conclusions

▶ Don’t use an ACO algorithm to solve the Min-Cut Problem.
▶ ACO can simulate Karger and Stein’s algorithm.
▶ We now understand better how ACO algorithms work.
▶ We now understand better how to analyze ACO algorithms.
Conclusions

- Don’t use an ACO algorithm to solve the Min-Cut Problem.
- ACO can simulate Karger and Stein’s algorithm.
  - We now understand better how ACO algorithms work.
  - We now understand better how to analyze ACO algorithms.
Conclusions

- Don’t use an ACO algorithm to solve the Min-Cut Problem.
- ACO can simulate Karger and Stein’s algorithm.
- We now understand better how ACO algorithms work.
- We now understand better how to analyze ACO algorithms.
Conclusions

- Don’t use an ACO algorithm to solve the Min-Cut Problem.
- ACO can simulate Karger and Stein’s algorithm.
- We now understand better how ACO algorithms work.
- We now understand better how to analyze ACO algorithms.
Thank you.