Ant Colony Optimization for the maximum Disjoint Paths Problem

Maria J. Blesa

LSI, Universitat Politècnica de Catalunya

Christian Blum

IRIDIA, Université Libre de Bruxelles

Partially supported by the FET Programme of the EU IST-1999-14186 (ALCOM-FT) and by the Spanish CICYT project TIC-2002-04498-C05-03 (**TRACER**). M. Blesa is also supported by the Catalan Research Council (2001FI-00659). C. Blum acknowledges support by the *Metaheuristics Network*, a Research Training Network funded by the Improving Human Potential program of the CEC (HPRN-CT-1999-00106).

Outline

- The (maximum) disjoint paths problem
- Greedy approaches
- Ant Colony Optimization (ACO) approach
- Further work to do

The maximum disjoint paths problem

The problem...

- Let $G = (V, E)$ be a graph and

  $$T = \{(s_j, t_j) \mid j = 1, \ldots, k; s_j \neq t_j \in V\},$$

  a list of commodities (or terminal pairs) of vertices in G.
  It is said that T is *realizable* in G if all the pairs $(s_j, t_j) \in T$ can be joined by mutually disjoint paths from s_j to t_j.

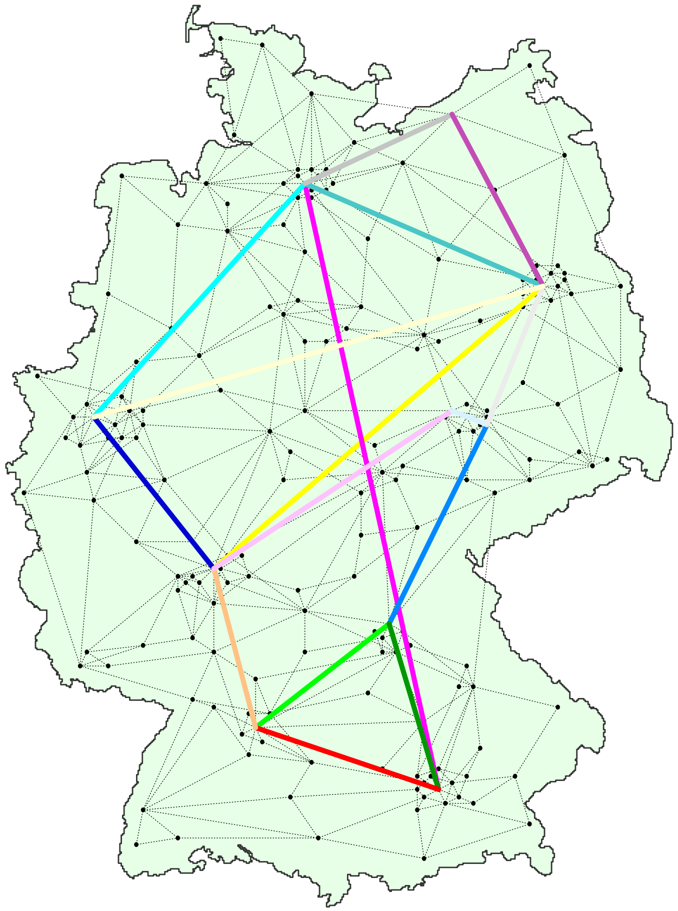
- The disjoint paths problem consists of determining whether T is realizable. It is NP-complete.[a]

- The optimization version consists on finding a realizable subset of T of maximum cardinality. We call it MDP problem.

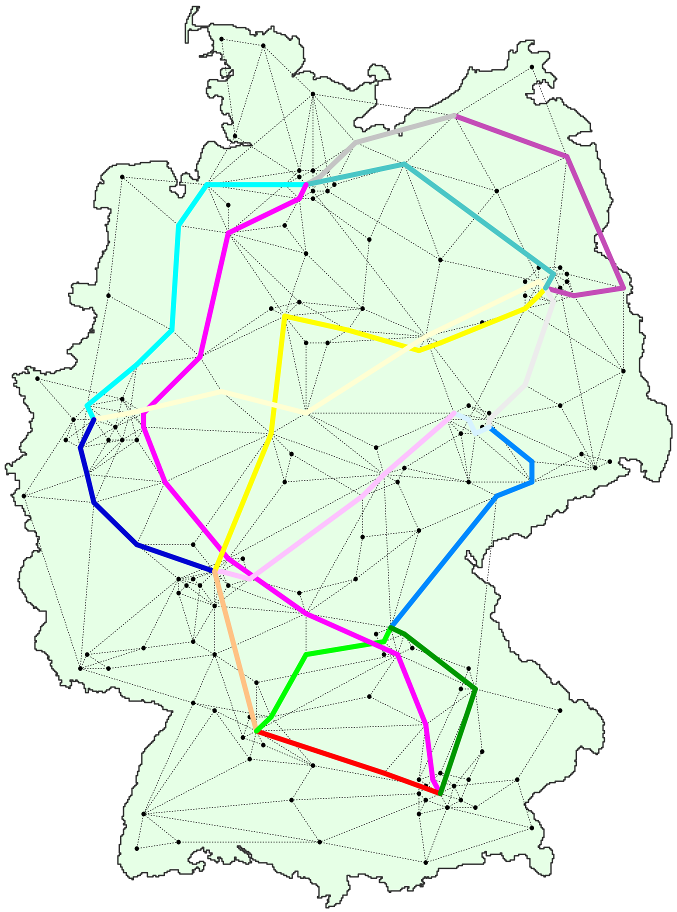
---

a. R.M. Karp. Compexity of Computer Computations. In *Reducibility among combinatorial problems*, pages 85–103. Plenum Press, New York, 1972.

The maximum disjoint paths problem



(a) Instance:Network platform with demands



(b) Solution: Pairwise node-disjoint paths

The maximum disjoint paths problem

Applications and interests in different areas:

- VLSI design
- Scheduling
- Load balancing
- Routing and admission control in modern networks
- Information dissemination: broadcasting and multicasting

The problem has been very much theoretically studied[a] but there is a lack of efficient algorithms to solve it in nowadays' communication networks.

- Branch and Bound, by the Technische Universität Berlin and the Deutsche Telekom AG, Darmstadt.
- Few greedy approaches, mainly used for demonstrating some theoretical approximations.

---

a. J. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, MIT, Cambridge, 1996.

Greedies for the maximum disjoint paths problem

A natural way of approximating the problem is to proceed through the terminal pairs in T in order, routing each commodity $T_i = (s_i, t_i)$ on the disjoint shortest path P_i in G, if possible[a].

Simple greedy algorithm (SGA)

---

input: a problem instance (G, T)

$\hat{E} \leftarrow E$, $P \leftarrow \emptyset$

for $i = 1, ..., |T|$ **do**

  if s_i and t_i can be connected by a path in $G = (V, \hat{E})$ **then**

    $P_i \leftarrow$ shortest path from s_i to t_i in $G = (V, \hat{E})$

    $P \leftarrow P \cup P_i$

    $\hat{E} \leftarrow \hat{E} \setminus \{e \mid e \in P_i\}$

  end if

end for

output: a set of paths P

---

[a]. J. Kleinberg. *Approximation algorithms for disjoint paths problems.* PhD thesis, MIT, Cambridge, 1996.

Greedies for the maximum disjoint paths problem

It is better to process commodities according to the length of the disjoint shortest path linking them, since more edges remain free for future usage[a].

Greedy path algorithm (GPA)

---

input: a problem instance (G, T)
$\tilde{T} \leftarrow T$, $P \leftarrow \emptyset$, $\hat{E} \leftarrow E$
while $\tilde{T} \neq \emptyset$ **do**
  $\tilde{T} \leftarrow \{T_i \in \tilde{T} \mid s_i$ and t_i can be connected by a path in $G = (V, \hat{E})\}$
  for $T_i \in \tilde{T}$ **do**
    $P_i \leftarrow$ shortest path from s_i to t_i in $G = (V, \hat{E})$
  end for
  $P_j \leftarrow \mathsf{argmin}\{l(P_i) \mid T_i \in \tilde{T}\}$
  $\tilde{T} \leftarrow \tilde{T} \setminus T_j$, $P \leftarrow P \cup P_j$, $\hat{E} \leftarrow \hat{E} \setminus \{e \mid e \in P_j\}$
end while
output: a set of paths P

---

[a]. S. Kolliopoulos and C. Stein. Approximating disjoint-paths problems using (greedy algorithms and) packing integer programs. In *6th Integer Programming and Combinatorial Optimization Conference*, LNCS 1412, pages 153–168, Springer, 1998.

Metaheuristics for the maximum disjoint paths problem

What about metaheuristics?

- Neighborhood-based metaheuristics:

  We need to define the notion of neighborhood. It does not seem easy for a problem like the maximum disjoint paths.

- Population-based metaheuristics:

  They seem more natural. Each individual can take care of linking one of the commodities in T.

Now that I am here... why not to try with Ant Colony Optimization[a] :-) ?

- One ant is assigned to each commodity $T_j \in T$, then $N_{ants} = |T|$
- Ant assigned to $T_j \in T$ uses pheromone values $\tau^j = \{\tau_e^j \mid e \in E\}$

---

[a]. M. Dorigo, G. DiCaro, L. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

ACO for the MDP problem

ACO for the maximum disjoint paths problem

input: a problem instance (G, T)
InitializePheromoneValues(τ)
while termination conditions not met **do**
  for $i = 1$ to N_{sols} **do**
    $S_i \leftarrow \emptyset$
    for $j = 1$ to N_{ants} **do**
      $P_j \leftarrow$ ConstructPath(T_j,τ^j)
      $S_i \leftarrow S_i \cup P_j$
    end for
    $S_i \leftarrow$ MakeFeasible(S_i)
  end for
  UpdatePheromoneValues(τ,...)
  $T \leftarrow$ PermuteRandomly(T)
end while
output: the best solution S_{gbest}

ACO for the MDP problem: *Construction mechanism*

For constructing a path P_j between the endpoints of $T_j = (s_j, t_j)$, the jth ant is doing the following:

1st. It chooses randomly to start either from s_j or from t_j

2nd. It iteratively constructs P_j by adding available incident edges that are not already considered in P_j, and neither labeled as *forbidden* by a backtracking move.

3rd. It returns the path P_j. If it arrived to the destination node then $P_j = (s_j, \ldots, t_j)$, otherwise P_j is empty.

Backtracking is done in case the ant finds itself in a node where all the outgoing links are either already used, or labeled as forbidden.

Observe that an ant seems to work independently of the rest ... not really. Disjointness is introduced in the transition probabilities.

ACO for the MDP problem: *Construction mechanism*

The choice where to move to is done in every step as follows:

70% of the times the next edge e^* is chosen deterministically,

$$e^* = \{v_c, u\} \leftarrow \mathsf{argmax}\left\{\left(\tau_e^j\right)^\alpha \cdot \mathbf{p}(D_e)^\beta \cdot \mathbf{p}(U_e)^\gamma \mid e \in \mathcal{I}^\star_{v_c}\right\} \ ,$$

where

- v_c is the current node
- $\mathcal{I}^\star_{v_c}$ the set of allowed edges in G
- $\mathbf{p}(D_e)$ is the influence of the shortest distance $\sigma(u, v_g)$ from u to the goal vertex v_g together with the weight $w(e)$ of e

$$\mathbf{p}(D_{e=\{v_c,u\}}) \leftarrow \frac{(\sigma(u,v_g)+w(e))^{-1}}{\sum\limits_{e'=\{v_c,u'\}\in\mathcal{I}^\star_{v_c}} (\sigma(u',v_g)+w(e'))^{-1}}$$

- $\mathbf{p}(U_e)$ is the influence of the overall usage of edge e

$$\mathbf{p}(U_e) \leftarrow \frac{U(e)^{-1}}{\sum\limits_{e'\in\mathcal{I}^\star_{v_c}} U(e')^{-1}} \ , \quad \text{where} \quad U(e) = \begin{cases} 2 : e \text{ already used in } S_i \\ 1 : \text{otherwise} \end{cases}$$

ACO for the MDP problem: *Construction mechanism*

The remaining...

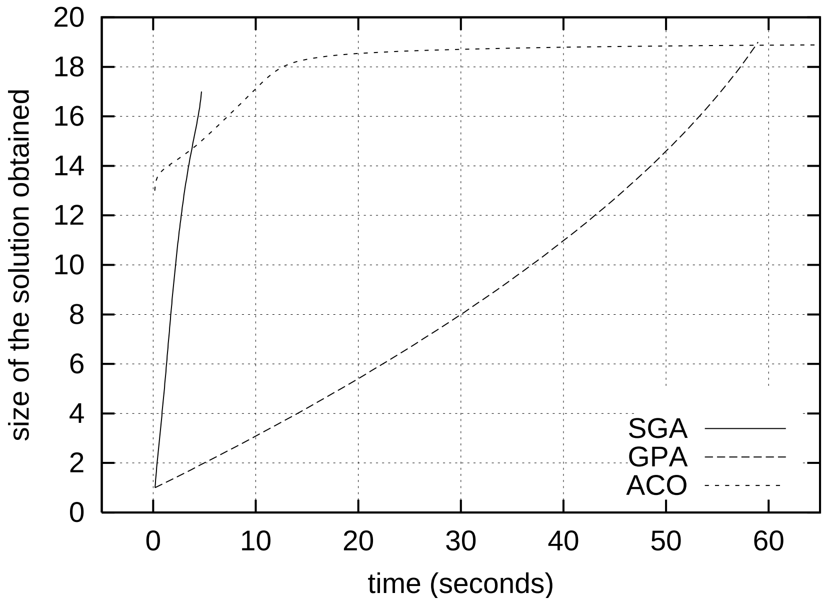
> **30% of the times** the ant choses an edge e^* according to the transition probabilities:

$$\mathbf{p}(e \mid \mathcal{I}^{\star}_{v_c}) = \begin{cases} \dfrac{(\tau^j_e)^{\alpha} \cdot \mathbf{p}(D_e)^{\beta} \cdot \mathbf{p}(U_e)^{\gamma}}{\sum\limits_{e' \in \mathcal{I}^{\star}_{v_c}} (\tau^j_{e'})^{\alpha} \cdot \mathbf{p}(D_{e'})^{\beta} \cdot \mathbf{p}(U_{e'})^{\gamma}} & : \text{ if } \quad e \in \mathcal{I}^{\star}_{v_c} \\ 0 & : \text{ otherwise.} \end{cases}$$

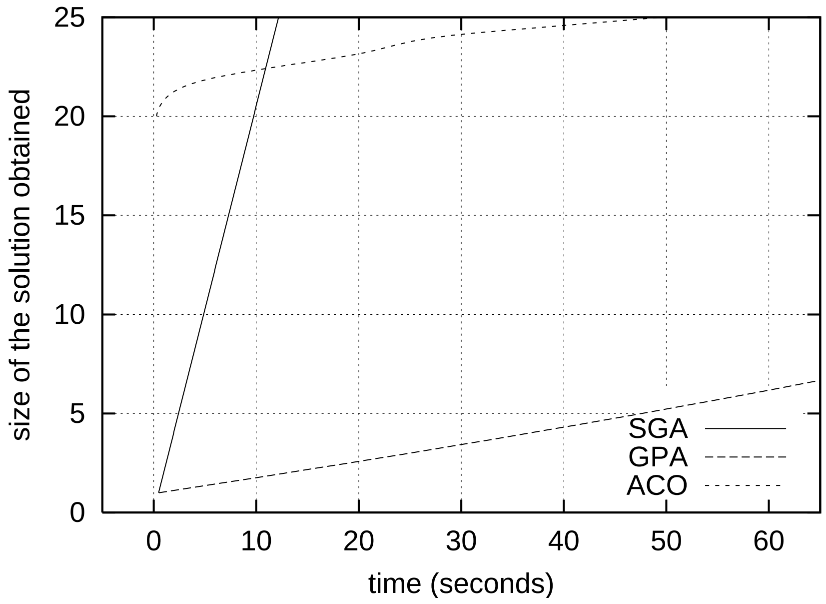
We have set $\alpha = 1.0$, $\beta = 5.0$ and $\gamma = 0.05$ in order to give a high influence to the overall usage criterion.

The high probability of doing a deterministic step leads the search already at the start of the algorithm to relatively good areas in the search space.

ACO for the MDP problem: *Little experiments*



(c) Graph **steinc5**: 500 nodes, 625 edges, 20 diameter, 1, 2.5 and 10 (min,avg,max) degrees.



(d) Graph **steinc15**: 500 nodes, 2500 edges, 16 diameter, 3, 10 and 20 (min, avg, max) degrees.

Common behavior of the SGA, GPA and ACO algorithms in their first minute of execution when trying to connect 25 random pairs.

Further work to do

- To use network representative instances, in terms of topology and size.
- To try more constructive methods, e.g.,
    - Iterated greedies,
    - Rollout methods,
    - GRASP,
    - Beam-ACO.
- To consider graphs with dynamic topologies.