Computing Locally Coherent Discourses

Alexander Koller
Saarland University

joint work with
Ernst Althaus and Nikiforos Karamanis

PARC
19 October 2004

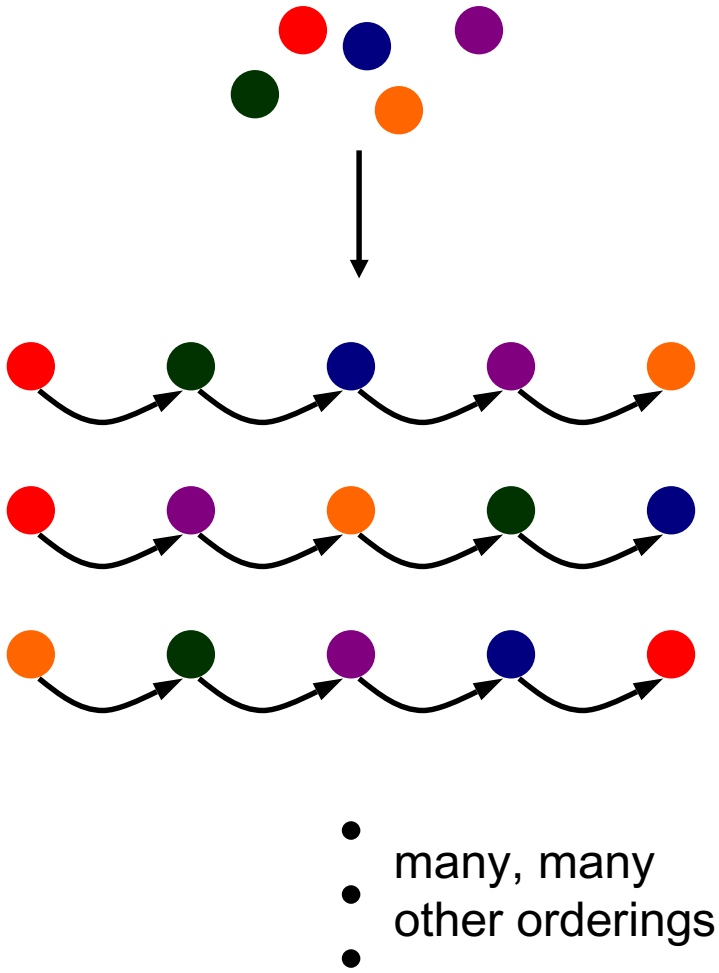
Overview

- Local coherence and the Discourse Ordering Problem
- Some examples for local coherence measures
- Equivalence of Discourse Ordering and TSP
- Computing discourse orderings
- Evaluation

Local coherence

- Human-written text is not a random ordering of discourse units.
- Discourse units are ordered to maximise coherence.
- Local coherence: Coherence is measured in terms of unit-to-unit transition costs.

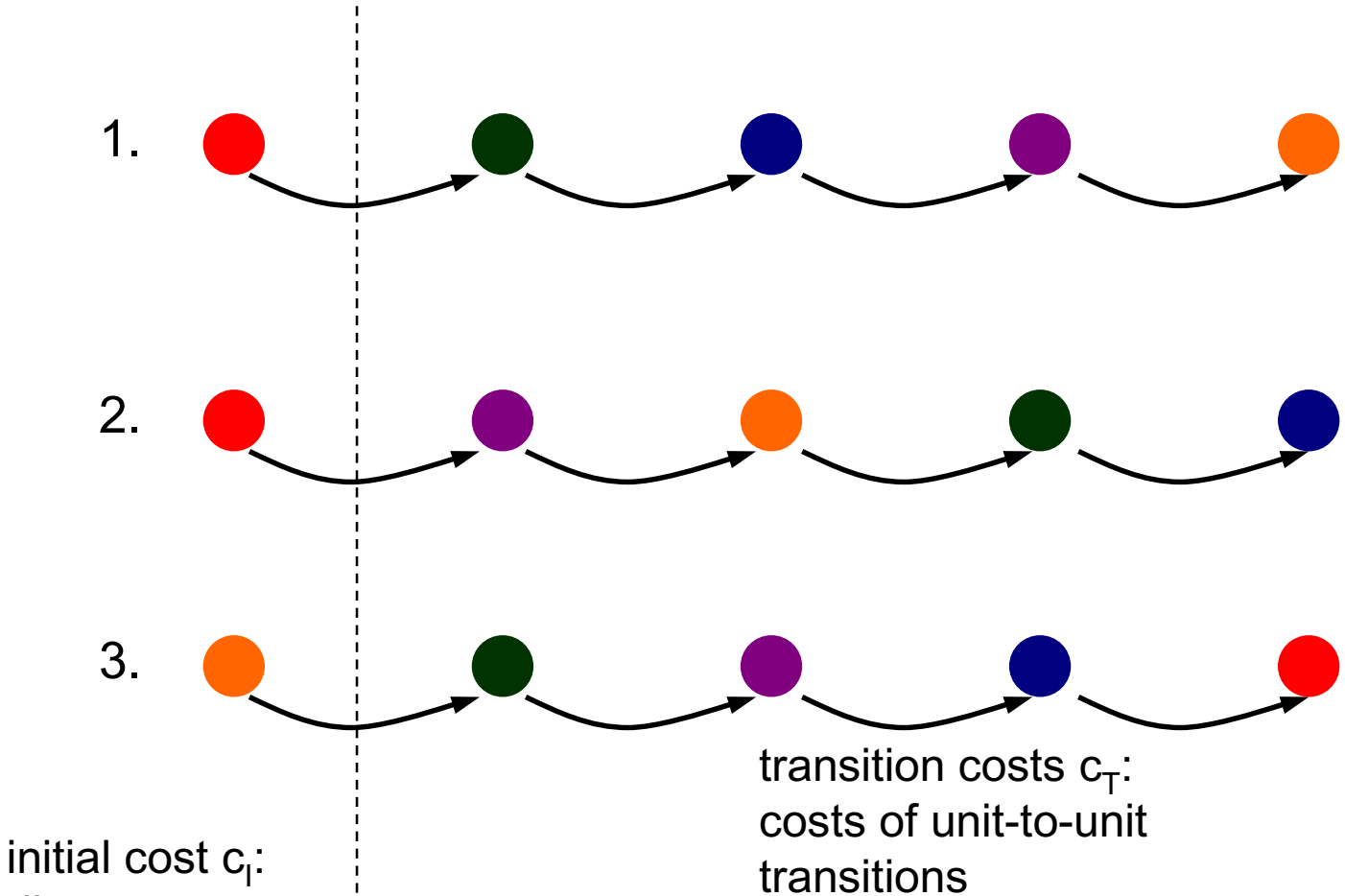
Local coherence



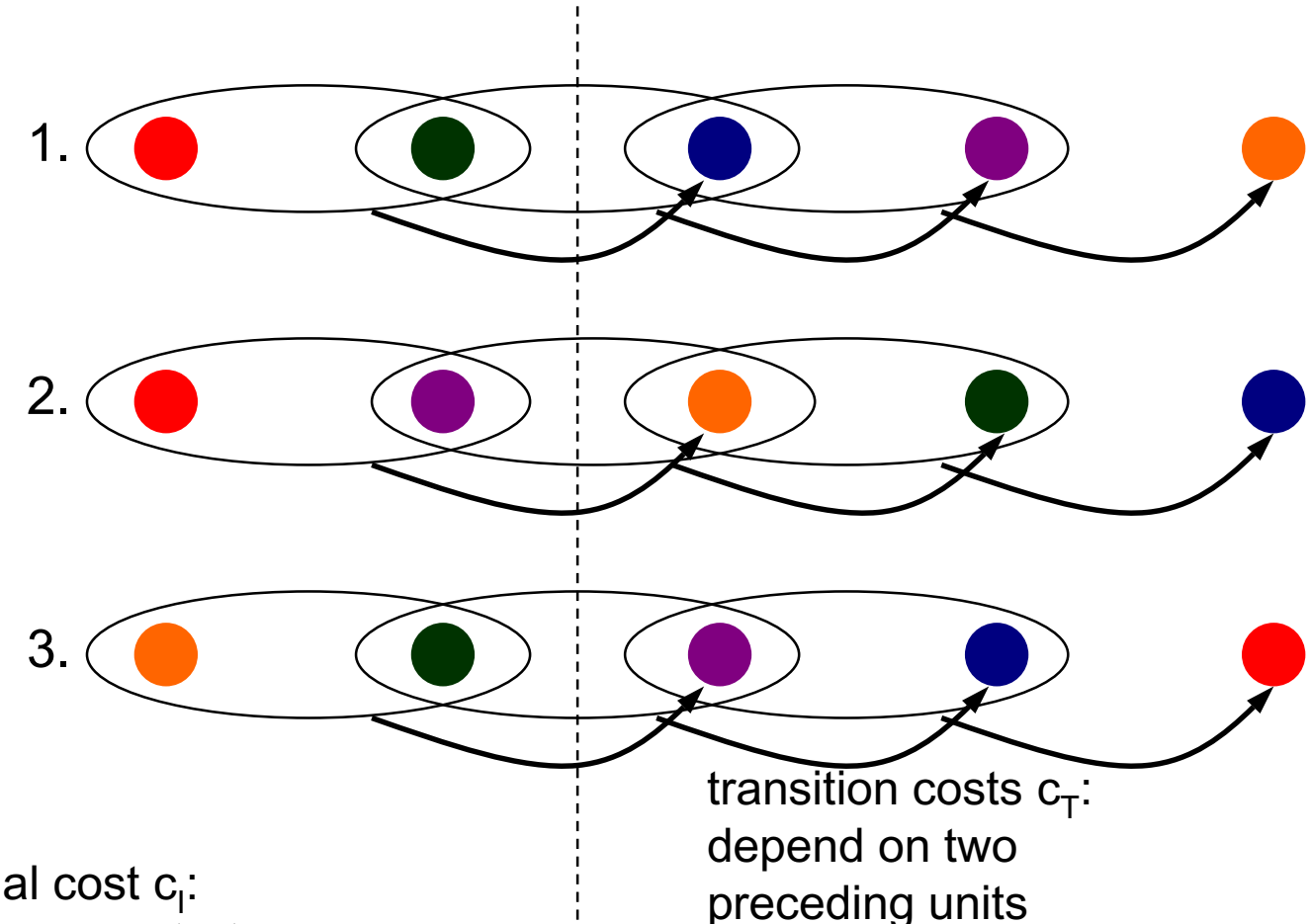
many, many
other orderings

Discourse ordering problem:
Find best ordering
according to local coherence

Local coherence: Our definition (2-place)



Local coherence: Our definition (3-place)



The Discourse Ordering Problem

- d-place Discourse Ordering Problem
- Input:
  - discourse units $u_1,...,u_n$
  - d-place transition costs c_T
  - (d-1)-place initial costs c_I
- Output:
  - a permutation π of {1,...,n} such that

$$c_I(u_{\pi(1)}, \ldots, u_{\pi(d-1)}) + \sum_{i=1}^{n-d+1} c_T(u_{\pi(i+d-1)} | u_{\pi(i)}, \ldots, u_{\pi(i+d-2)})$$

  is minimal.

Some cost functions from the literature

- Based on Centering Theory:
  - coherence of a transition is defined in terms of entity coherence
  - Karamanis & Manurung 02, Karamanis et al. 04 compare such cost functions
- Based on statistical models:
  - Lapata 03 gives cost function based on various features of adjacent sentences
  - d = 2, $c_T(u_2 \mid u_1) = - \log P(u_2 \mid u_1)$

Centering Theory

- The entities referred to in a discourse unit are called the forward-looking centers.
- Cf(u) is the list of entities in unit u, ranked by salience.
- Cp(u) -- the preferred center -- is the highest-ranked member of Cf(u).
- Cb(u_i) -- the backward-looking center -- is the highest-ranked member of Cf(u_i) that also appears in Cf(u_{i-1}).

CT-based transitions and cost functions

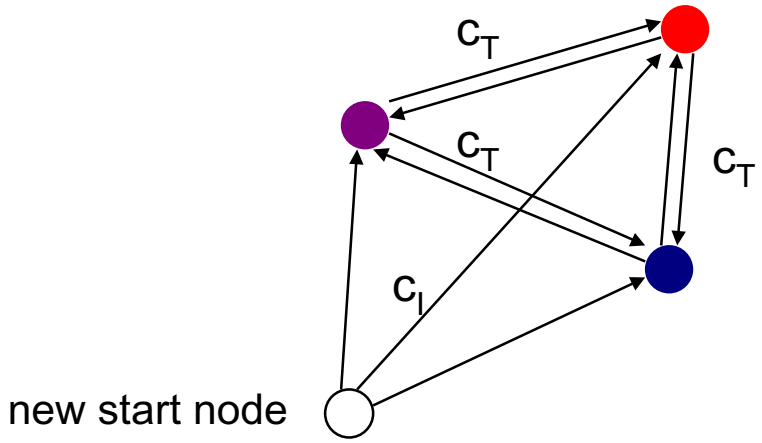
		COHERENCE: $\mathrm{Cb}(u_i) = \mathrm{Cb}(u_{i-1})$	COHERENCE*: $\mathrm{Cb}(u_i) \neq \mathrm{Cb}(u_{i-1})$
SALIENCE:	$\mathrm{Cb}(u_i) = \mathrm{Cp}(u_i)$	CONTINUE	SMOOTH-SHIFT
SALIENCE*:	$\mathrm{Cb}(u_i) \neq \mathrm{Cp}(u_i)$	RETAIN	ROUGH-SHIFT

	d	initial cost $c_I(u_1, \ldots, u_{d-1})$	transition cost $c_T(u_d \mid u_1, \ldots, u_{d-1})$
M.NOCB	2	0	$nocb_2$
M.KP	3	$nocb_2 + nocheap_2 + nosal_2$	$nocb_2 + nocheap_2 + nosal_2 + nocoh_3$
M.BFP	3	$(1 - nosal_2, nosal_2, 0, 0)$	$(cont_3, ret_3, ss_3, rs_3)$

Computational issues

- Number of permutations is too big for generate-and-test. Need 77 years for discourse of length 20.
- How hard is the problem really?
- If it is hard, can we solve it anyway?
  - Mellish et al. 98: Genetic Programming
  - Lapata 03: Approximative graph algorithm
  - No guarantees for quality of found solution

Discourse ordering as a graph problem



- each edge has a cost
- find cheapest path that visits all nodes

The Discourse Ordering Problem (with graphs)

- Input:
  - Graph G = (V,E)
  - start node s $\in$ V
  - d-place cost function c : $V^d \rightarrow \mathbf{R}$

- Output:
  - a simple directed path P = (s = v_0, v_1, ..., v_n) that visits all vertices, such that

$$\sum_{i=0}^{n-d+1} c(v_i, v_{i+1}, \ldots, v_{i+d-1})$$

    is minimal.

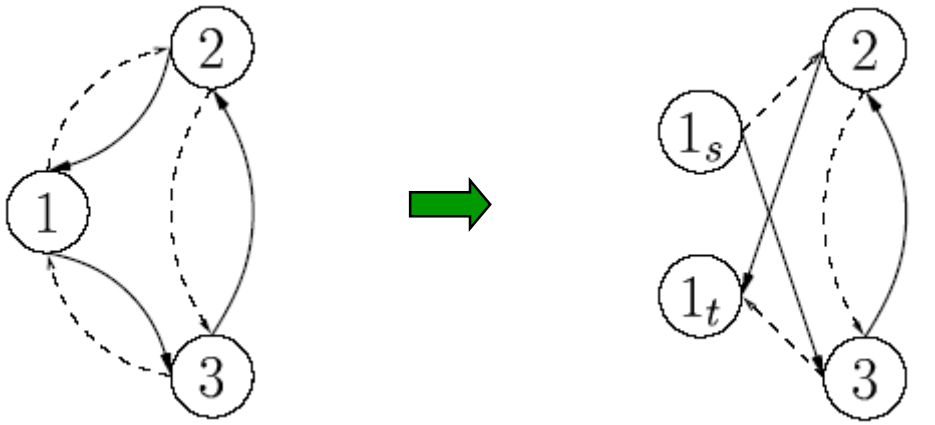
Travelling Salesman Problem

- Find cheapest round trip in a graph that visits every node exactly once.
- Classical NP-complete problem.
- Cannot be approximated: There is no general polynomial algorithm that guarantees good solutions (unless P = NP).
- Generalised asymmetric TSP (GATSP): Partition nodes into disjoint sets, tour must visit each set exactly once.

d-PDOP is equivalent to TSP

- Can reduce TSP to 2-PDOP:
  - hence NP-hard,
    no approximation algorithms
- Can reduce d-PDOP to GATSP:
  - hence, can apply algorithms for TSP
  - need GATSP to encode d-place cost function for d > 2.

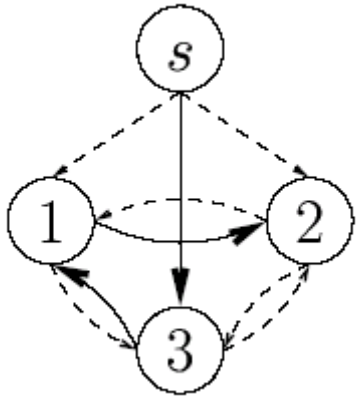
Reduction of TSP to 2-PDOP

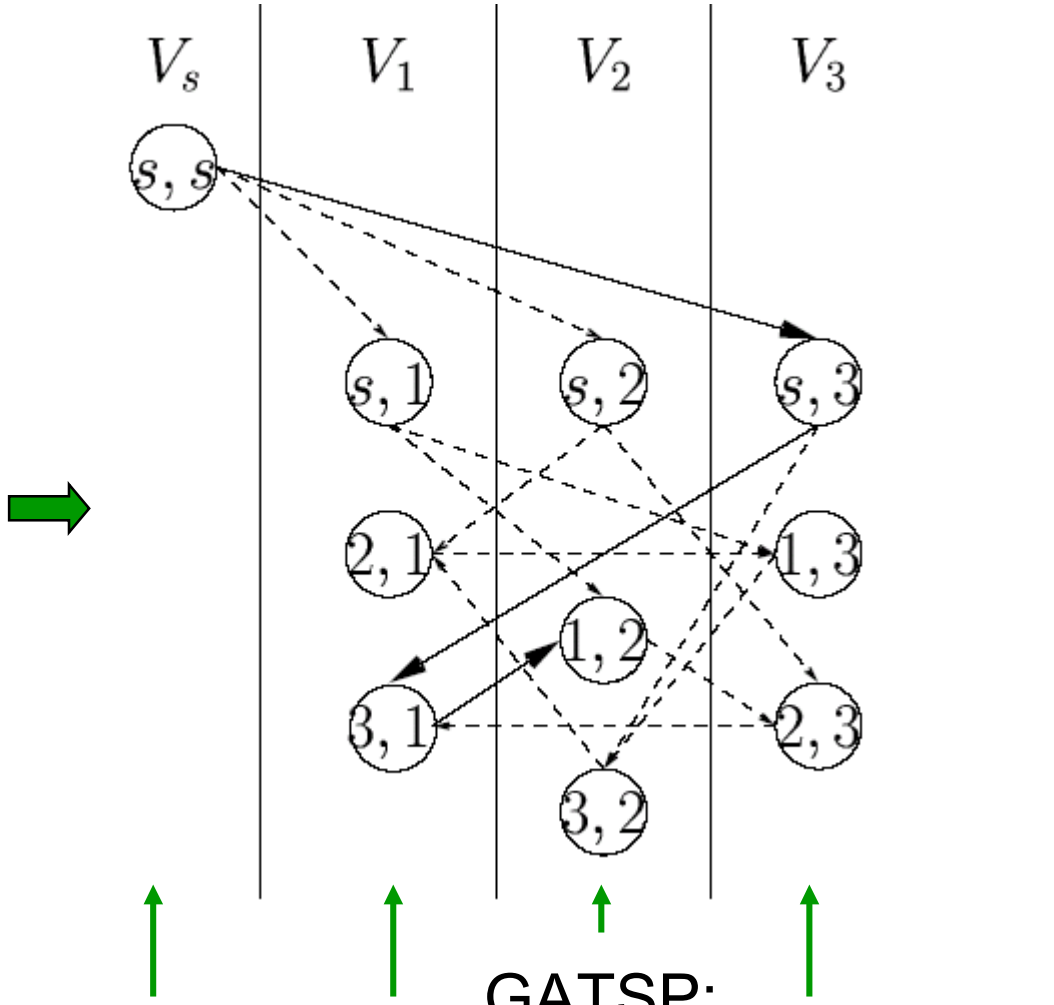


TSP

2-PDOP

Reduction of d-PDOP to GATSP





3-PDOP

GATSP:
Visit every column exactly once

Solving GATSP by Linear Programming

- A standard method for solving TSP and other combinatorial problems:
  Integer Linear Programming (ILP).
- Write problem as set of linear equations and inequalities.
- Find integer numbers that satisfy all (in)equations.

A Linear Program for GATSP

$$\min \quad \sum_{e \in E} c_e x_e$$

$$s.t. \quad \sum_{e \in \delta^+(v)} x_e = \sum_{e \in \delta^-(v)} x_e \quad \forall\, v \in V \qquad (1)$$

$$\sum_{e \in \delta^-(V_i)} x_e = 1 \qquad 1 \leq i \leq n \qquad (2)$$

$$\sum_{e \in \delta^+(\cup_{i \in I} V_i)} x_e \geq 1 \qquad I \subset \{1, \ldots, n\} \qquad (3)$$

$$x_e \in \{0, 1\}$$

Solving the Linear Program

- Branch-and-cut technique
- Solve linear (in)equations for arbitrary (not necessarily integer) values.
- Pick variable x with fractional value and solve subproblems with $x = 1$ and $x = 0$.
- Continue until all variables have integer values.
- Use heuristics to pick good variable and keep search space small.

Further Optimisations

- Exponential number of inequalities of Type 3:
  - start with a small subset of inequalities
  - check preliminary solutions for violations of Type 3 inequalities
  - add violated inequalities by need
- Add redundant inequalities that can be violated by non-integer solutions.

Evaluation

- Four cost functions:
  - M.LAPATA: d = 2, statistics-based
    (evaluated on discourses from BLLIP corpus)
  - Three centering-based cost functions
    (evaluated on discourses from GNOME corpus)
- Random graphs of different sizes
- Two ILP solvers:
  - CPLEX 9.0 (commercial)
  - SCIP 0.6.3 / SOPLEX 1.2.2a (free)

Evaluation: d = 2

Instance	Size	ILP-FS	ILP-CS
lapata-10	13	0.05	0.05
coffers1 M.NOCB	10	0.04	0.02
cabinet1 M.NOCB	15	0.07	0.01
random (avg)	20	0.09	0.07
random (avg)	40	0.28	0.17
random (avg)	60	1.39	0.40
random (avg)	100	6.17	1.97

(seconds CPU time, Pentium 4 at 3 GHz)

Evaluation: d = 3

Instance	Size	ILP-FS	ILP-CS
coffers1 M.KP	10	0.05	0.05
coffers1 M.BFP	10	0.08	0.06
cabinet1 M.KP	15	0.40	1.12
cabinet1 M.BFP	15	0.39	0.28
random (avg)	10	1.00	0.42
random (avg)	15	35.1	5.79
random (avg)	20	-	115.8

(seconds CPU time, Pentium 4 at 3 GHz)

An example output

Both cabinets probably entered England in the early nineteenth century / after the French Revolution caused the dispersal of so many French collections. / The pair to [this monumental cabinet] still exists in Scotland. / The fleurs-de-lis on the top two drawers indicate that [the cabinet] was made for the French King Louis XIV. / [It] may have served as a royal gift, / as [it] does not appear in inventories of [his] possessions. / Another medallion inside shows [him] a few years later. / The bronze medallion above [the central door] was cast from a medal struck in 1661 which shows [the king] at the age of twenty-one. / A panel of marquetry showing the cockerel of [France] standing triumphant over both the eagle of the Holy Roman Empire and the lion of Spain and the Spanish Netherlands decorates [the central door]. / In [the Dutch Wars] of 1672 - 1678, [France] fought simultaneously against the Dutch, Spanish, and Imperial armies, defeating them all. / [The cabinet] celebrates the Treaty of Nijmegen, which concluded [the war]. / The Sun King's portrait appears twice on [this work]. / Two large figures from Greek mythology, Hercules and Hippolyta, Queen of the Amazons, representatives of strength and bravery in war appear to support [the cabinet]. / The decoration on [the cabinet] refers to [Louis XIV's] military victories. / On the drawer above the door, gilt-bronze military trophies flank a medallion portrait of [the king].

(cabinet1, M.NOCB; cost = 2)

Evaluation: Interpretation

- Runtimes for d = 2 are fast.
- For d = 3, still quite fast for real-life examples.
- Coherence measures seem to be easier than random graphs.

Conclusion

- Discourse Ordering problem is equivalent to Travelling Salesman.
- Can use heuristics to compute optimal solution very efficiently (up to 50 discourse units per second).
- Applications:
  - real systems (generation and summarisation)
  - experimentation

Future Work

- Optimise GATSP algorithm.
- What makes real-life instances easier than random graphs?
- Global coherence measures (i.e., arrange units in discourse tree structure)?