Computing Locally Coherent Discourses

Ernst Althaus  
LORIA  
Université Henri Poincaré  
Vandœuvre-lès-Nancy, France  
althaus@loria.fr

Nikiforos Karamanis  
School of Informatics  
University of Edinburgh  
Edinburgh, UK  
N.Karamanis@sms.ed.ac.uk

Alexander Koller  
Dept. of Computational Linguistics  
Saarland University  
Saarbrücken, Germany  
koller@coli.uni-sb.de

Abstract
We present the first algorithm that computes optimal orderings of sentences into a locally coherent discourse. The algorithm runs very efficiently on a variety of coherence measures from the literature. We also show that the discourse ordering problem is NP-complete and cannot be approximated.

1 Introduction
One central problem in discourse generation and summarisation is to structure the discourse in a way that maximises coherence. Coherence is the property of a good human-authored text that makes it easier to read and understand than a randomly-ordered collection of sentences.

Several papers in the recent literature (Mellish et al., 1998; Barzilay et al., 2002; Karamanis and Manurung, 2002; Lapata, 2003; Karamanis et al., 2004) have focused on defining local coherence, which evaluates the quality of sentence-to-sentence transitions. This is in contrast to theories of global coherence, which can consider relations between larger chunks of the discourse and e.g. structures them into a tree (Mann and Thompson, 1988; Marcu, 1997; Webber et al., 1999). Measures of local coherence specify which ordering of the sentences makes for the most coherent discourse, and can be based e.g. on Centering Theory (Walker et al., 1998; Brennan et al., 1987; Kibble and Power, 2000; Karamanis and Manurung, 2002) or on statistical models (Lapata, 2003).

But while formal models of local coherence have made substantial progress over the past few years, the question of how to efficiently compute an ordering of the sentences in a discourse that maximises local coherence is still largely unsolved. The fundamental problem is that any of the factorial number of permutations of the sentences could be the optimal discourse, which makes for a formidable search space for nontrivial discourses. Mellish et al. (1998) and Karamanis and Manurung (2002) present algorithms based on genetic programming, and Lapata (2003) uses a graph-based heuristic algorithm, but none of them can give any guarantees about the quality of the computed ordering.

This paper presents the first algorithm that computes optimal locally coherent discourses, and establishes the complexity of the discourse ordering problem. We first prove that the discourse ordering problem for local coherence measures is equivalent to the Travelling Salesman Problem (TSP). This means that discourse ordering is NP-complete, i.e. there are probably no polynomial algorithms for it. Worse, our result implies that the problem is not even approximable; any polynomial algorithm will compute arbitrarily bad solutions on unfortunate inputs. Note that all approximation algorithms for the TSP assume that the underlying cost function is a metric, which is not the case for the coherence measures we consider.

Despite this negative result, we show that by applying modern algorithms for TSP, the discourse ordering problem can be solved efficiently enough for practical applications. We define a branch-and-cut algorithm based on linear programming, and evaluate it on discourse ordering problems based on the GNOME corpus (Karamanis, 2003) and the BLLIP corpus (Lapata, 2003). If the local coherence measure depends only on the adjacent pairs of sentences in the discourse, we can order discourses of up to 50 sentences in under a second. If it is allowed to depend on the left-hand context of the sentence pair, computation is often still efficient, but can become expensive.

The structure of the paper is as follows. We will first formally define the discourse ordering problem and relate our definition to the literature on local coherence measures in Section 2. Then we will prove the equivalence of discourse ordering and TSP (Section 3), and present algorithms for solving it in Section 4. Section 5 evaluates our algorithms on examples from the literature. We compare our approach to various others in Section 6, and then conclude in Section 7.
2 The Discourse Ordering Problem

We will first give a formal definition of the problem of computing locally coherent discourses, and demonstrate how some local coherence measures from the literature fit into this framework.

2.1 Definitions

We assume that a discourse is made up of discourse units (depending on the underlying theory, these could be utterances, sentences, clauses, etc.), which must be ordered to achieve maximum local coherence. We call the problem of computing the optimal ordering the discourse ordering problem.

We formalise the problem by assigning a cost to each unit-to-unit transition, and a cost for the discourse to start with a certain unit. Transition costs may depend on the local context, i.e. a fixed number of discourse units to the left may influence the cost of a transition. The optimal ordering is the one which minimises the sum of the costs.

Definition 1. A d-place transition cost function for a set U of discourse units is a function \( c_T : U^d \rightarrow \mathbb{R} \). Intuitively, \( c_T(u_n|u_1, \ldots, u_{d-1}) \) is the cost of the transition \((u_{d-1}, u_d)\) given that the immediately preceding units were \(u_1, \ldots, u_{d-2}\).

A d-place initial cost function for U is a function \( c_I : U^d \rightarrow \mathbb{R} \). Intuitively, \( c_I(u_1, \ldots, u_d) \) is the cost for the fact that the discourse starts with the sequence \(u_1, \ldots, u_d\).

The d-place discourse ordering problem is defined as follows: Given a set \(U = \{u_1, \ldots, u_n\}\), a d-place transition cost function \(c_T\) and a \((d-1)\)-place initial cost function \(c_I\), compute a permutation \(\pi \) of \(1, \ldots, n\) such that

\[
\begin{align*}
&c_I(u_{\pi(1)}, \ldots, u_{\pi(d-1)}) \\
&+ \sum_{i=1}^{n-d+1} c_T(u_{\pi(i+d-1)}|u_{\pi(i)}, \ldots, u_{\pi(i+d-2)})
\end{align*}
\]

is minimal.

The notation for the cost functions is suggestive: The transition cost function has the character of a conditional probability, which specifies that the cost of continuing the discourse with the unit \(u_d\) depends on the local context \(u_1, \ldots, u_{d-1}\). This local context is not available for the first \(d-1\) units of the discourse, which is why their costs are summarily covered by the initial function.

2.2 Centering-Based Cost Functions

One popular class of coherence measures is based on Centering Theory (CT, (Walker et al., 1998)). We will briefly sketch its basic notions and then show how some CT-based coherence measures can be cast into our framework.

The standard formulation of CT e.g. in (Walker et al., 1998), calls the discourse units utterances, and assigns to each utterance \(u_i\) in the discourse a list \(\text{Cf}(u_i)\) of forward-looking centres. The members of \(\text{Cf}(u_i)\) correspond to the referents of the NPs in \(u_i\) and are ranked in order of prominence, the first element being the preferred centre \(\text{Cp}(u_i)\). The backward-looking centre \(\text{Cb}(u_i)\) is defined as the highest ranked element of \(\text{Cf}(u_i)\) which also appears in \(\text{Cf}(u_{i-1})\), and serves as the link between the two subsequent utterances \(u_{i-1}\) and \(u_i\). Each utterance has at most one \(\text{Cb}\). If \(u_i\) and \(u_{i-1}\) have no forward-looking centres in common, or if \(u_i\) is the first utterance in the discourse, then \(u_i\) does not have a \(\text{Cb}\) at all.

Based on these concepts, CT classifies the transitions between subsequent utterances into different types. Table 1 shows the most common classification into the four types CONTINUE, RETAIN, SMOOTH-SHIFT, and ROUGH-SHIFT, which are predicted to be less and less coherent in this order (Brennan et al., 1987). Kibble and Power (2000) define three further classes of transitions: COHERENCE and SALIENCE, which are both defined in Table 1 as well, and NOCB, the class of transitions for which \(\text{Cb}(u_i)\) is undefined. Finally, a transition is considered to satisfy the CHEAPNESS constraint (Strube and Hahn, 1999) if \(\text{Cb}(u_i) = \text{Cp}(u_{i-1})\).

Table 2 summarises some cost functions from the literature, in the reconstruction of Karamanis et al. (2004). Each line shows the name of the coherence measure, the arity \(d\) from Definition 1, and the initial and transition cost functions. To fit the definitions in one line, we use terms of the form \(f_k\), which abbreviate applications of \(f\) to the last \(k\) arguments of the cost functions, i.e. \(f(u_{d-k+1}, \ldots, u_d)\).

The most basic coherence measure, M.NOCB (Karamanis and Manurung, 2002), simply assigns to each NOCB transition the cost 1 and to every other transition the cost 0. The definition of \(c_T(u_2|u_1)\), which decodes to \(\text{nocb}(u_1, u_2)\), only looks at the two units in the transition, and no further context. The initial costs for this coherence measure are always zero.

The measure M.KP (Kibble and Power, 2000) sums the value of \(\text{nocb}\) and the values of three functions which evaluate to 0 if the transition is cheap, salient, or coherent, and 1 otherwise. This is an instance of the 3-place discourse ordering problem because \(\text{COHERENCE}\) depends on \(\text{Cb}(u_{i-1})\), which itself depends on \(\text{Cf}(u_{i-2})\); hence \(\text{nocoh}\) must take
Table 1: COHERENCE, SALIENCE and the table of standard transitions

| d   | initial cost $c_I(u_1, \ldots, u_{d-1})$ | transition cost $c_T(u_d|u_1, \ldots, u_{d-1})$ |
|-----|----------------------------------------|-----------------------------------------------|
| M.NOCB | 2 0                                    | nocb2                                        |
| M.KP     | 3 $nocb_2 + nocheap_2 + nasal_2$      | $nocb_2 + nocheap_2 + nasal_2 + noco_3$     |
| M.BFP    | 3 $(1 - nasal_2, nasal_2, 0, 0)$       | $(cont_3, ret_3, ss_3, rs_3)$                |
| M.LAPATA | 2 $-\log P(u_1)$                     | $-\log P(u_2|u_1)$                          |

Table 2: Some cost functions from the literature.

3 Equivalence of Discourse Ordering and TSP

Now we show that discourse ordering and the travelling salesman problem are equivalent. In order to do this, we first redefine discourse ordering as a graph problem.

$d$-place discourse ordering problem ($d$PDOP):

Given a directed graph $G = (V, E)$, a node $s \in V$ and a function $c : V^d \rightarrow \mathbb{R}$, compute a simple directed path $P = (s, v_0, v_1, \ldots, v_n)$ from $s$ through all vertices in $V$ which minimises $\sum_{i=0}^{n-d+1} c(v_i, v_{i+1}, \ldots, v_{i+d-1})$. We write instances of $d$PDOP as $(V, E, s, c)$.

The nodes $v_1, \ldots, v_n$ correspond to the discourse units. The cost function $c$ encodes both the initial and the transition cost functions from Section 2 by returning the initial cost if its first argument is the (new) start node $s$.

Now let’s define the version of the travelling salesman problem we will use below.

Generalised asymmetric TSP (GATSP): Given a directed graph $G = (V, E)$, edge weights $c : E \rightarrow \mathbb{R}$, and a partition $(V_1, \ldots, V_k)$ of the nodes $V$, compute the shortest directed cycle that visits exactly one node of each $V_i$. We call such a cycle a tour and write instances of GATSP as $((V_1, \ldots, V_k), E, c)$.

The usual definition of the TSP, in which every node must be visited exactly once, is the special case of GATSP where each $V_i$ contains exactly one node. We call this case asymmetric travelling salesman problem, ATSP.
We will show that ATSP can be reduced to 2PDOP, and that any dPDOP can be reduced to GATSP.

### 3.1 Reduction of ATSP to 2PDOP

First, we introduce the reduction of ATSP to 2PDOP, which establishes NP-completeness of dPDOP for all \( d > 1 \). The reduction is approximation preserving, i.e. if we can find a solution of 2PDOP that is worse than the optimum only by a factor of \( \epsilon \) (an \( \epsilon \)-approximation), it translates to a solution of ATSP that is also an \( \epsilon \)-approximation. Since it is known that there can be no polynomial algorithms that compute \( \epsilon \)-approximations for general ATSP, for any \( \epsilon \) (Cormen et al., 1990), this means that dPDOP cannot be approximated either (unless P=NP): Any polynomial algorithm for dPDOP will compute arbitrarily bad solutions on certain inputs.

The reduction works as follows. Let \( G = (V_1, \ldots, V_k, E, c) \) be an instance of ATSP, and \( V = V_1 \cup \ldots \cup V_k \). We choose an arbitrary node \( v \in V \) and split it into two nodes \( v_s \) and \( v_t \). We assign all edges with source node \( v \) to \( v_s \) and all edges with target node \( v \) to \( v_t \) (compare Figure 1). Finally we make \( v_s \) the source node of our 2PDOP instance \( G' \).

For every tour in \( G \), we have a path in \( G' \) starting at \( v_s \) visiting all other nodes (and ending in \( v_t \)) with the same cost by replacing the edge \((v, u)\) out of \( v \) by \((v_s, u)\) and the edge \((w, v)\) into \( v \) by \((w, v_t)\). Conversely, for every path starting at \( v_s \) visiting all nodes, we have an ATSP tour of the same cost, since all such paths will end in \( v_t \) (as \( v_t \) has no outgoing edges).

An example is shown in Fig. 1. The ATSP instance on the left has the tour \((1, 3, 2, 1)\), indicated by the solid edges. The node 1 is split into the two nodes \( 1_s \) and \( 1_t \), and the tour translates to the path \((1_s, 3, 2, 1_t)\) in the 2PDOP instance.

### 3.2 Reduction of dPDOP to GATSP

Conversely, we can encode an instance \( G = (V, E, s, c) \) of dPDOP as an instance \( G' = ((V'_u)_{u \in V}, E', c') \) of GATSP, in such a way that the optimal solutions correspond. The cost of traversing an edge in dPDOP depends on the previous \( d - 1 \) nodes; we compress these costs into ordinary costs of single edges in the reduction to GATSP.

The GATSP instance has a node \([u_1, \ldots, u_{d-1}]\) for every \( d - 1 \)-tuple of nodes of \( V \). It has an edge from \([u_1, \ldots, u_{d-1}]\) to \([u_2, \ldots, u_{d-1}, u_d]\) iff there is an edge from \( u_{d-1} \) to \( u_d \) in \( G \), and it has an edge from each node into \([s, \ldots, s]\). The idea is to encode a path \( P = (s = u_0, u_1, \ldots, u_n) \) in \( G \) as a tour \( T_P \) in \( G' \) that successively visits the nodes \([u_i, \ldots, u_{i+d-1}]\), \( i = 0, \ldots, n \), where we assume that \( u_j = s \) for all \( j \leq 0 \) (compare Figure 2).

The cost of \( T_P \) can be made equal to the cost of \( P \) by making the cost of the edge from \([u_1, \ldots, u_{d-1}]\) to \([u_2, \ldots, u_d]\) equal to \( c(u_1, \ldots, u_d) \). (We set \( c'(c) \) to 0 for all edges \( e \) between nodes with first component \( s \) and for the edges \( e \) with target node \([s, s, \ldots, s]\).)

Finally, we define \( V'_u \) to be the set of all nodes in \( G' \) with last component \( u \). It is not hard to see that for any simple path of length \( n \) in \( G \), we find a tour \( T_P \) in \( G' \) with the same cost. Conversely, we can find for every tour in \( G' \) a simple path of length \( n \) in \( G \) with the same cost.

Note that the encoding \( G' \) will contain many unnecessary nodes and edges. For instance, all nodes that have no incoming edges can never be used in a tour, and can be deleted. We can safely delete such unnecessary nodes in a post-processing step.

An example is shown in Fig. 2. The 3PDOP instance on the left has a path \((s, 3, 1, 2)\), which translates to the path \(([s, s], [s, 3], [3, 1], [1, 2])\) in the GATSP instance shown on the right. This path can be completed by a tour by adding the edge
The equivalence of dPDOP and GATSP implies that we can now bring algorithms from the vast literature on TSP to bear on the discourse ordering problem. One straightforward method is to reduce the GATSP further to ATSP (Noon and Bean, 1993); for the case \( d = 2 \), nothing has to be done. Then one can solve the reduced ATSP instance; see (Fischetti et al., 2001; Fischetti et al., 2002) for a recent survey of exact methods.

We choose the alternative of developing a new algorithm for solving GATSP directly, which uses standard techniques from combinatorial optimisation, gives us a better handle on optimising the algorithm for our problem instances, and runs more efficiently in practice. Our algorithm translates the GATSP instance into an integer linear program (ILP) and uses the branch-and-cut method (Nemhauser and Wolsey, 1988) to solve it. Integer linear programs consist of a set of linear equations and inequalities, and are solved by integer variable assignments which maximise or minimise a goal function while satisfying the other conditions.

Let \( G = (V, E) \) be a directed graph and \( S \subseteq V \). We define \( \delta^+(S) = \{(u, v) \in E \mid u \in S \text{ and } v \notin S\} \) and \( \delta^-(S) = \{(u, v) \in E \mid u \notin S \text{ and } v \in S\} \), i.e. \( \delta^+(S) \) and \( \delta^-(S) \) are the sets of all incoming and outgoing edges of \( S \), respectively. We assume that the graph \( G \) has no edges within one partition \( V_u \), since such edges cannot be used by any solution. With this assumption, GATSP can be phrased as an ILP as follows (this formulation is similar to the one proposed by Laporte et al. (1987)):

\[
\begin{align*}
\min \quad & \sum_{e \in E} c_e x_e \\
\text{s.t.} \quad & \sum_{e \in \delta^+(v)} x_e = \sum_{e \in \delta^-(v)} x_e \quad \forall v \in V \quad (1) \\
& \sum_{e \in \delta^+(V_i)} x_e = 1 \quad 1 \leq i \leq n \quad (2) \\
& \sum_{e \in \delta^+(\cup_{j \in I} V_i)} x_e \geq 1 \quad I \subset \{1, \ldots, n\} \quad (3) \\
& x_e \in \{0, 1\}
\end{align*}
\]

We have a binary variable \( x_e \) for each edge \( e \) of the graph. The intention is that \( x_e \) has value 1 if \( e \) is used in the tour, and 0 otherwise. Thus the cost of the tour can be written as \( \sum_{e \in E} c_e x_e \). The three conditions enforce the variable assignment to encode a valid GATSP tour. (1) ensures that all integer solutions encode a set of cycles. (2) guarantees that every partition \( V_i \) is visited by exactly one cycle. The inequalities (3) say that every subset of the partitions has an outgoing edge; this makes sure a solution encodes one cycle, rather than a set of multiple cycles.

To solve such an ILP using the branch-and-cut method, we drop the integrality constraints (i.e. we replace \( x_e \in \{0, 1\} \) by \( 0 \leq x_e \leq 1 \)) and solve the corresponding linear programming (LP) relaxation. If the solution of the LP is integral, we found the optimal solution. Otherwise we pick a variable with a fractional value and split the problem into two subproblems by setting the variable to 0 and 1, respectively. We solve the subproblems recursively and disregard a subproblem if its LP bound is worse than the best known solution.

Since our ILP contains an exponential number of inequalities of type (3), solving the complete LPs directly would be too expensive. Instead, we start with a small subset of these inequalities, and test (efficiently) whether a solution of the smaller LP violates an inequality which is not in the current LP. If so, we add the inequality to the LP, resolve it, and iterate. Otherwise we found the solution of the LP with the exponential number of inequalities. The inequalities we add by need are called cutting planes; algorithms that find violated cutting planes are called separation algorithms.

To keep the size of the branch-and-cut tree small, our algorithm employs some heuristics to find further upper bounds. In addition, we improve lower bound from the LP relaxations by adding further inequalities to the LP that are valid for all integral solutions, but can be violated for optimal solutions of the LP. One major challenge here was to find separation algorithms for these inequalities. We cannot go into these details for lack of space, but will discuss them in a separate paper.

## 5 Evaluation

We implemented the algorithm and ran it on some examples to evaluate its practical efficiency. The runtimes are shown in Tables 3 and 4 for an implementation using a branch-and-cut ILP solver which is free for all academic purposes (ILP-FS) and a commercial branch-and-cut ILP solver (ILP-CS).

Our implementations are based on LEDA 4.4.1
The first is a family of algorithms for discourse ordering based on genetic programming (Mellish et al., 1998; Karamanis and Manurung, 2002). This is a very flexible and powerful approach, which can be applied to measures of local coherence that do not seem to fit in our framework trivially. For example, the measure from (Mellish et al., 1998) looks at the entire discourse up to the current transition for some of their cost factors. However, our algorithm is several orders of magnitude faster where a direct comparison is possible (Manurung, p.c.), and it is guaranteed to find an optimal ordering. The non-approximability result for TSP means that a genetic (or any other) algorithm which is restricted to polynomial runtime could theoretically deliver arbitrarily bad solutions.

Second, the discourse ordering problem we have discussed in this paper looks very similar to the Majority Ordering problem that arises in the context of multi-document summarisation (Barzilay et al., 2004).
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].

Figure 3: An example output based on M.NOCB.

2002). The difference between the two problems is that Barzilay et al. minimise the sum of all costs $C_{ij}$ for any pair $i, j$ of discourse units with $i < j$, whereas we only sum over the $C_{ij}$ for $i = j - 1$. This makes their problem amenable to the approximation algorithm by Cohen et al. (1999), which allows them to compute a solution that is at least half as good as the optimum, in polynomial time; i.e. this problem is strictly easier than TSP or discourse ordering. However, a Majority Ordering algorithm is not guaranteed to compute good solutions to the discourse ordering problem, as Lapata (2003) assumes.

7 Conclusion

We have shown that the problem of ordering clauses into a discourse that maximises local coherence is equivalent to the travelling salesman problem: Even the two-place discourse ordering problem can encode ATSP. This means that the problem is NP-complete and doesn’t even admit polynomial approximation algorithms (unless P=NP).

On the other hand, we have shown how to encode the discourse ordering problems of arbitrary arity $d$ into GATSP. We have demonstrated that modern branch-and-cut algorithms for GATSP can easily solve practical discourse ordering problems if $d = 2$, and are still usable for many instances with $d = 3$. As far as we are aware, this is the first algorithm for discourse ordering that can make any guarantees about the solution it computes.

Our efficient implementation can benefit generation and summarisation research in at least two respects. First, we show that computing locally coherent orderings of clauses is feasible in practice, as such coherence measures will probably be applied on sentences within the same paragraph, i.e. on problem instances of limited size. Second, our system should be a useful experimentation tool in developing new measures of local coherence.

We have focused on local coherence in this paper, but it seems clear that notions of global coherence, which go beyond the level of sentence-to-sentence transitions, capture important aspects of coherence that a purely local model cannot. However, our algorithm can still be useful as a subroutine in a more complex system that deals with global coherence (Marcu, 1997; Mellish et al., 1998). Whether our methods can be directly applied to the tree structures that come up in theories of global coherence is an interesting question for future research.

Acknowledgments. We would like to thank Mirella Lapata for providing the experimental data and Andrea Lodi for providing an efficiency baseline by running his ATSP solver on our inputs. We are grateful to Malte Gabsdil, Ruli Manurung, Chris Mellish, Kristina Striegnitz, and our reviewers for helpful comments and discussions.

References

R. Barzilay, N. Elhadad, and K. R. McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. Journal of Artificial Intelligence Research, 17:35–55.

S. Brennan, M. Walker Friedman, and C. Pollard. 1987. A centering approach to pronouns. In Proc. 25th ACL, pages 155–162, Stanford.

W. Cohen, R. Schapire, and Y. Singer. 1999. Learning to order things. Journal of Artificial Intelligence Research, 10:243–270.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. 1990. Introduction to Algorithms. MIT Press, Cambridge.

M. Fischetti, A. Lodi, and P. Toth. 2001. Solv-
ing real-world ATSP instances by branch-and-cut. *Combinatorial Optimization.*

M. Fischetti, A. Lodi, and P. Toth. 2002. Exact methods for the asymmetric traveling salesman problem. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations.* Kluwer.

N. Karamanis and H. M. Manurung. 2002. Stochastic text structuring using the principle of continuity. In *Proceedings of INLG-02,* pages 81–88, New York.

N. Karamanis, M. Poesio, C. Mellish, and J. Oberlander. 2004. Evaluating centering-based metrics of coherence for text structuring using a reliably annotated corpus. In *Proceedings of the 42nd ACL,* Barcelona.

N. Karamanis. 2003. *Entity Coherence for Descriptive Text Structuring.* Ph.D. thesis, Division of Informatics, University of Edinburgh.

R. Kibble and R. Power. 2000. An integrated framework for text planning and pronominalisation. In *Proc. INLG 2000,* pages 77–84, Mitzpe Ramon.

M. Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proc. 41st ACL,* pages 545–552, Sapporo, Japan.

G. Laporte, H. Mercure, and Y. Nobert. 1987. Generalized travelling salesman problem through n sets of nodes: the asymmetrical case. *Discrete Applied Mathematics, 18:*185–197.

W. Mann and S. Thompson. 1988. Rhetorical structure theory: A theory of text organization. *Text,* 8(3):243–281.

D. Marcu. 1997. From local to global coherence: A bottom-up approach to text planning. In *Proceedings of the 14th AAAI,* pages 629–635.

C. Mellish, A. Knott, J. Oberlander, and M. O’Donnell. 1998. Experiments using stochastic search for text planning. In *Proc. 9th INLG,* pages 98–107, Niagara-on-the-Lake.

G.L. Nemhauser and L.A. Wolsey. 1988. *Integer and Combinatorial Optimization.* John Wiley & Sons.

C.E. Noon and J.C. Bean. 1993. An efficient transformation of the generalized traveling salesman problem. *Information Systems and Operational Research, 31*(1).

M. Strube and U. Hahn. 1999. Functional centering: Grounding referential coherence in information structure. *Computational Linguistics, 25*(3).

M. Walker, A. Joshi, and E. Prince. 1998. Centering in naturally occurring discourse: An overview. In M. Walker, A. Joshi, and E. Prince, editors, *Centering Theory in Discourse,* pages 1–30. Clarendon Press, Oxford.

B. Webber, A. Knott, M. Stone, and A. Joshi. 1999. What are little trees made of: A structural and presuppositional account using Lexicalized TAG. In *Proc. 36th ACL,* pages 151–156, College Park.