Experiments Using Stochastic Search for Text Planning

Chris Mellish, Alistair Knott, Jon Oberlander and Mick O'Donnell
Department of Artificial Intelligence and Human Communication Research Centre,
University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN
Email: {chrism,micko}@dai.ed.ac.uk, {alik,jon}@cogsci.ed.ac.uk

Abstract

Marcu has characterised an important and difficult problem in text planning: given a set of facts to convey and a set of rhetorical relations that can be used to link them together, how can one arrange this material so as to yield the best possible text? We describe experiments with a number of heuristic search methods for this task.

1 Introduction: Text Planning

1.1 The Task

This paper presents some initial experiments using stochastic search methods for aspects of text planning. The work was motivated by the needs of the ILEX system for generating descriptions of musum artefacts (in particular, 20th Century jewellery) [Mellish et al 98]. We present results on examples semi-automatically generated from datastructures that exist within ILEX.

Forming a set of facts about a piece of jewellery into a structure that yields a coherent text is a non-trivial problem. Rhetorical Structure Theory [Mann and Thompson 87] claims that a text is coherent just in case it can be analysed hierarchically in terms of *relations* between text spans. Much work in NLG makes the assumption that constructing something like an RS tree is a necessary step in the planning of a text. This work takes as its starting point Marcu's [Marcu 97] excellent formalisation of RST and the problem of building legal RST trees, and for the purposes of this paper the phrase "text planning" will generally denote the task characterised by him. In this task, one is given a set of facts *all* of which should be included in a text and a set of relations between facts, *some* of which can be included in the text. The task is to produce a legal RS tree using the facts and some relations (or the "best" such tree).

Following the original work on RST and assumptions that have been commonly made in subsequent work, we will assume that there is a fixed set of possible relations (we include "joint" as a second-class relation which can be applied to any two facts, but whose use is not preferred). Each relation has a *nucleus* and a *satellite* (we don't consider multiple nuclei or satellites here, apart from the case of "joint", which is essentially multinuclear). Each relation may be indicated by a distinctive "cue phrase", with the nucleus and satellite being realised in some fashion around it. Each relation has applicability conditions which can be tested between two atomic facts. For two complex text spans, a relation holds exactly when that relation holds between the nuclei of those spans. Relations can thus hold between text spans of arbitrary size.

Figure 1 shows an example of the form of the input that is used for the experiments reported here. Each primitive "fact" is represented in terms of a subject, verb and complement (as well as a unique identifier). The

```
fact('this item','is','a figurative jewel',f6).
fact(bleufort,'was','a french designer',f3).
fact(shiltredge,'was','a british designer',f7).
fact('this item','was made by',bleufort,f8).
fact(titanium,'is','a refractory metal',f4).

rel(contrast,f7,f3,[]).          mentions(F,O) :-
rel(elab,F1,F2,[]) :-               fact(O,_,_,F).
   mentions(F1,O),               mentions(F,O) :-
   mentions(F2,O),                  fact(_,_,O,F).
   \+ F1=F2.
```

Figure 1: Example Input

"subject" is assumed to be the entity that the fact is "about". The approaches reported here have not yet been linked to a realisation component, and so the entities are represented simply by canned phrases for readability (it is assumed that each entity in the domain has a fixed distinctive phrase that is always used for it). Relations are represented in terms of the relation name, the nucleus and satellite facts and a list (in this example, empty) of precondition facts which need to have been assimilated before the relation can be used (this represents an extension to Marcu's chcracterisation). This example uses the definition of (object-attribute) "elaboration" that we will be using consistently, namely that one fact can elaborate another if they have an entity in common (of course, there are other kinds of elaborations, but we would want to model them differently).

1.2 Controlling Search in Text Planning

There seem to be three main approaches to controlling the search for a good RS tree (or something similar). One is to restrict what relations can appear in the nucleus and satellite of others (for instance, using Hovy's [Hovy 90] idea of "growth points"). This is a step towards creating "schemas" for larger pieces of text. It can therefore be expected that it will produce very good results in restricted domains where limited text patterns are used, but that it will be hard to extend it to freer text types. The second idea is to use information about goals to limit possibilities. This is an element of Hovy's work but is more apparent in the planning work of Moore and Paris [Moore and Paris 93]. This second approach will work well if there are strong goals in the domain which really can influence textual decisions. This is not always the case. For instance, in our ILEX domain [Mellish et al 98] the system's goal is something very general like "say interesting things about item X, subject to length and coherence constraints".

The third approach, most obviously exemplified by [Marcu 97], is to use some form of explicit search through possible trees, guided by heuristics about tree quality. Marcu first of all attempts to find the best ordering of the facts. For every relation that could be indicated, constraints are generated saying what the order of the two facts involved should be and that the facts should be adjacent. The constraints are weighted according to attributes of rhetorical relations that have been determined empirically. A standard constraint satisfaction algorithm is used to find the linear sequence such that the total weight of the satisfied constraints is maximal. Once the sequence of facts is known, a general algorithm [Marcu 96] is used to construct all possible RS trees based on those facts. It is not clear how the best such tree is selected, though clearly the adjacency and order constraints could in principle be reapplied in some way (possibly with other heuristics that Marcu has used in rhetorical parsing) to select a tree.

We are interested in further developing the ideas of Marcu, but seek to address the following problems:

1. It is not clear how scalable the approach is. Constraint satisfaction in general is intractable, and having weighted constraints seems to make matters worse. Enumerating all RS trees that can be built on a given sequence of facts also has combinatorical problems. Marcu's approach may not be much better than one that builds all possible trees. Yet if there are enough relations to link any pair of facts (which, given the existence of elaboration, may often be nearly the case), the number of trees whose top nucleus are a specified fact grows from 336 to 5040 to 95040 as the number of facts grows from 5 to 6 to 7. In our examples, we have more like 20-30 facts.

2. As Marcu points out, the constraints on linear order only indirectly reflect requirements on the tree (because related facts need not appear consecutively). Though in fact we will use the idea of planning via a linear sequence later, we would like to experiment using measures of quality that are applied directly to the trees. We also have a number of factors that we would like to take account of in the evaluation (see section 3 below).

2 Stochastic Search

Building a good RS tree is a search problem. Stochastic search methods are a form of heuristic search that use the following generic algorithm:

1. Construct a set of random candidate solutions.
2. Until some time limit is reached,
   Randomly pick one or more items from the set, in such a way as to prefer items with the best "scores".
   Use these to generate one or more new random variations.
   Add these to the set, possibly removing less preferred items in order to keep the size constant.

Examples of stochastic search approaches are stochastic hillclimbing, simulated annealing and evolutionary algorithms. The approaches differ according to factors like the size of the population of possible solutions that is maintained, the operations for generating new possibilities and any special mechanisms for avoiding local maxima. They are similar to one another (and different from constraint satisfaction and enumeration approaches) in that they are heuristic (not guaranteed to find optimal solutions) and they are "anytime". That is, such an algorithm can be stopped at any point and it will be able to yield at that point a result which is the best it has found so far. This is important for NLG applications where interface considerations mean that texts have to be produced within a limited time.

3 Evaluating RST trees

A key requirement for the use of any stochastic search approach is the ability to assess the quality of a possible solution. Thus we are forced to confront directly the task of evaluating RST trees.

We assign a candidate tree a score which is the sum of scores for particular features the tree may have. A positive score here indicates a good feature and a negative one indicates a bad one.

We cannot make any claims to have the best way of evaluating RS trees. The problem is far too complex and our knowledge of the issues involved so meagre that only a token gesture can be made at this point. We offer the following evaluation scheme merely so that the basis of our experiments is clear and because we believe that some of the ideas are starting in the right direction. Here are the features that we score for:

Topic and Interestingness We assume that the entity that the text is "about" is specified with the input. It is highly desirable that the "top nucleus" (most important nucleus) of the text be about this entity. Also we prefer texts that use interesting relations. We score as follows:

-10 for a top nucleus not mentioning the subject of the text
-30 for a joint relation
+21 for a relation other than joint and elaboration

Size of Substructures Scott and de Souza [Scott and de Souza 90] say that the greater the amount of intervening text between the propositions of a relation, the more difficult it will be to reconstruct its message. We score as follows:

-4 for each fact that will come textually between a satellite and its nucleus

Constraints on Information Ordering Our relations have preconditions which are facts that should be conveyed before them. We score as follows:

-20 for an unsatisfied precondition for a relation

Focus Movement We do not have a complex model of focus development through the text, though development of such a model would be worthwhile. As McKeown and others have done, we prefer certain transitions over others. If consecutive facts mention the same entities or verb, the prospects for aggregation are greater, and this is usually desirable. We score as follows:

-9 for a fact (apart from the first) not mentioning any previously mentioned entity
-3 for a fact not mentioning any entity in the previous fact, but whose subject is a previously mentioned entity
+3 for a fact retaining the subject of the last fact as its subject
+3 for a fact using the same verb as the previous one

Object Introduction When an entity is first introduced as the subject of a fact, it is usual for that to be a very general statement about the entity. Preferring this introduces a mild schema-like influence to the system. We score as follows:

+3 for the first fact with a given entity as subject having verb "is"

4 Using Stochastic Search for Text Planning

Using the above evaluation metric for RS trees, we have experimented with a range of stochastic search methods. Space does not permit us to discuss more than one initial experiment in this section. In the next section, we describe a couple of methods based on genetic algorithms which proved more productive.

4.1 Subtree Swapping

The subtree swapping approach produces new trees by swapping random subtrees in a candidate solution. It works as follows:

1. Initialise with a tree for each combination of interesting (non-elaboration) relations, with any fact only appearing in

one. Make into a complete tree by combining together these relations and any unused facts with "joint" relations (or better ones if available).

2. Repeatedly select a random tree and swap over two random subtrees, repairing all relations. Add the new tree to the population.

When two subtrees are swapped over in an RS tree, some of the relations indicated in the tree no longer apply (i.e. those higher relations that make use of the nuclei of the subtrees). These are "repaired" by in each case selecting the "best" valid relation that really relates the top nuclei (i.e. a non-elaboration relation is chosen if possible, otherwise an elaboration if that is valid, with "joint" as a last resort).

We investigated variations on this algorithm, including having initial random balanced trees (including the "best" relation at each point) and focussing the subtree swapping on subtrees that contributed to bad scores, but the above algorithm was the one that seemed most successful.

4.2 Initial Results

Figure 2 shows an example text generated by subtree swapping. Note that we have taken liberties in editing by hand the surface text (for instance, by introducing better referring expressions and aggregation). The ordering of the material and the use of rhetorical relations are the only things which are determined by the algorithm.

Results for subtree swapping are shown together with later results in Figure 5 (the example text shown for subtree swapping is for the item named `j-342540`). The most obvious feature of these results is the huge variability of the results, which suggests that there are many local maxima in the search space. Looking at the texts produced, we can see a number of problems. Unfortunately, if there is only one way smoothly to include a fact in the text, the chance of finding it by random subtree swapping is very low. The same goes for fixing other local problems in the text. The introduction of "the previous jewel" is an example of this. This entity can only be introduced elegantly through the fact that it, like the current item, is encrusted with jewels. The text is still suffering from material getting between a satellite and its nucleus. For instance, there is a relation (indicated by the colon) between "It is encrusted with jewels" and "it has silver links encrusted asymmetrically...", but this is weakened by the presence of "and is an Organic style jewel" in the middle).

The trouble is that subtree swapping needs incrementally to acquire all good features not present in whichever initial tree develops into the best solution. It can only acquire these features "accidentally" and the chances of stumbling on them are small. Different initial trees will contain different good fragments, and it seems desirable to be able to combine the good parts of different solutions. This motivates using some sort of *crossover* operation that can combine elements of two solutions into a new one [Goldberg 89]. But it is not immediately clear how crossover could work on two RS trees. In particular, two chosen trees will rarely have non-trivial subtrees with equal fringes. Their way of breaking up the material may be so different that it is hard to imagine how one could combine elements of both.

5 Restricting the Space of RST Trees

As a way of making a crossover operation conceivable, our first step has been to reduce the planning problem to that of planning the sequential order of the facts (in a way that echoes Marcu's approach to some extent). We have done this by making certain restrictions on the RS trees that we are prepared to build. In particular, we make the following assumptions:

This jewel is made from diamonds, yellow metal, pearls, oxidized white metal and opals.
It was made in 1976 and was made in London.
This jewel draws on natural themes for inspiration: it uses natural pearls.
It was made by Flockinger who is an English designer.
Flockinger lived in London which is a city.
This jewel is a necklace and is set with jewels.
It is encrusted with jewels and is an Organic style jewel: it has silver links encrusted asymetrically with pearls and diamonds.
Indeed, Organic style jewels are usually encrusted with jewels.
Organic style jewels usually draw on natural themes for inspiration and are made up of asymmetrical shapes.
Organic style jewels usually have a coarse texture.

This jewel is 72.0 cm long.

The previous jewel has little diamonds scattered around its edges and has an encrusted bezel. It is encrusted with jewels: it features diamonds encrusted on a natural shell.

Figure 2: Example Text from Subtree Swapping

1. The nucleus and satellite of a non-joint relation can never be separated.
2. "Joint" relations are used to connect unrelated paragraphs.

With these assumptions, an RS tree is characterised (almost) by the sequence of facts at its leaves. Indeed, we have an algorithm that almost deterministically builds a tree from a sequence of facts, according to these principles. (The algorithm is not completely deterministic, because there may be more than one non-elaboration relation that can be used with two given facts as nucleus and satellite – our evaluation function won't, of course, differentiate between these).

The algorithm for building a tree from a sequence essentially makes a tree that can be processed by a reader with negligable short-term memory. The tree will be right-branching and if the reader just remembers the last fact at any point, then they can follow the connection between the text so far and the next fact[1]. Interestingly, Marcu uses "right skew" to disambiguate between alternative trees produced in rhetorical *parsing*. Here we are setting it as a much harder constraint. The only exception is "joint" relations, which can join together texts of any size, but since there is no real relation involved in them there is no memory load in interpreting them.

The first two assumptions above make fundamental use of the order in which facts will appear in the text. For simplicity, we assume that every relation has a fixed order of nucleus and satellite (though this assumption could be relaxed). The approach is controversial in that it takes into account realisation order in

[1]In fact, there is local left-branching for relations whose nucleus is presented after the satellite, but such relations are often presented using embedded clauses in a way that signals the deviation from right-branching clearly to the reader. These structures cannot be nested.

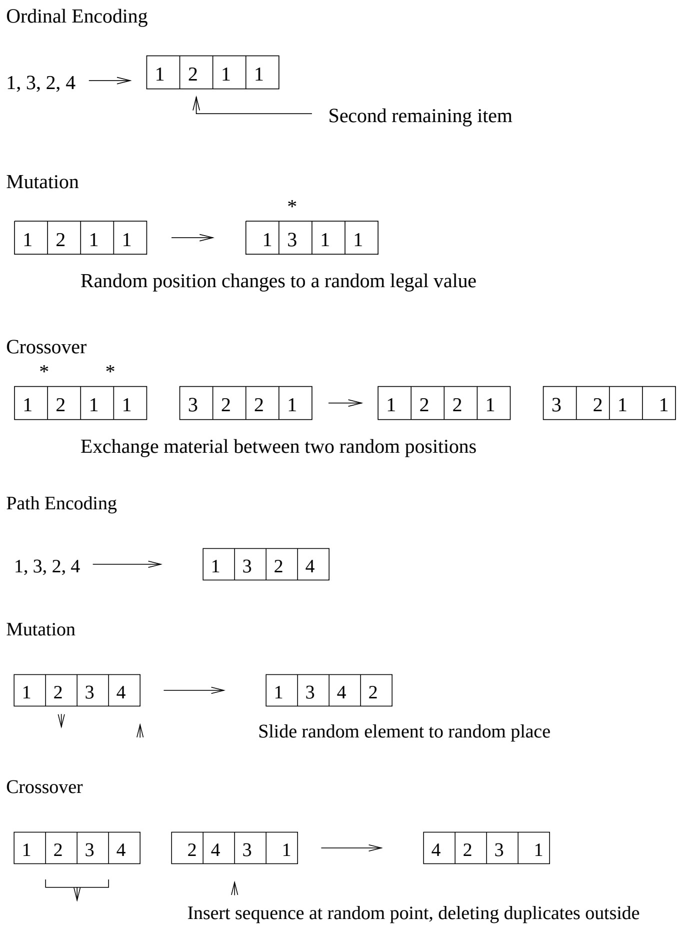

Figure 3: Ordinal and Path Representations

the criterion for a legal tree. It is likely that the above assumptions will not apply equally well to all types of text. Still, they mean that the planning problem can be reduced to that of planning a sequence. The next experiments were an attempt to evaluate this idea.

6 Using a Genetic Algorithm

The genetic algorithm we used takes the following form:

1. Enumerate a set of random initial sequences by loosely following sequences of facts where consecutive facts mention the same entity.
2. Evaluate sequences by evaluating the trees they give rise to.
3. Perform mutation and crossover on the sequences, with mutation having a relatively small probability.
4. When the "best" sequence has not changed for a time, invoke mutation repeatedly until it does.
5. Stop after a given number of iterations, and return the tree for the "best" sequence.

Notice that although the algorithm manipulates sequences, the evaluation is one that operates on *trees*. Mutation is a unary operation which, given one sequence, generates a new one. Crossover is binary in that it generates

This jewel is made from diamonds and yellow metals.
It was made by Flockinger, who was an English designer.
Flockinger lived in London, which is a city.
This jewel was made in London.
It is a necklace.
It is made from oxidized white metal, pearls and opals.
It is set with jewels.
This jewel is encrusted with jewels: it has silver links encrusted asymetrically with pearls and diamonds.
This jewel was made in 1976.
It is an Organic style jewel and is 72.0 cm long.
It draws on natural themes for inspiration: it uses natural pearls. Indeed, Organic style jewels usually draw on natural themes for inspiration.
Organic style jewels usually have a coarse texture, are usually made up of asymmetrical shapes and are usually encrusted with jewels.
The previous jewel is encrusted with jewels: it features diamonds encrusted on a natural shell.
It has little diamonds scattered around its edges and an encrusted bezel.

Figure 4: Text Planned by GA

new solution(s) based on *two* existing ones. The choice of mutation and crossover operations depends on how the sequences are internally represented and should facilitate the exchange of useful subparts of solutions. Two different representations have been tried so far. The relevant features are summarised in Figure 3.

6.1 Ordinal Representation

The ordinal representation [Michalewicz 92] assumes that there is an initial canonical sequence of facts (in the figure, this is assumed to be 1,2,3,4). A given sequence is represented by a sequence of numbers, where the ith element indicates the position of the ith element of the sequence in that canonical sequence with all previous elements deleted. So the ith element is always a number between 1 and $n+1-i$, where n is the length of the sequence. Mutation is implemented by a change of a random element to a random legal value. Crossover (here) is implemented by two-point crossover - the material between two random points of the sequences (the same points for both) is swapped over, yielding two new sequences. The ordinal representation has been used extensively for tasks such as the travelling salesman problem, and it has the advantage that the crossover operation is particularly simple.

6.2 Path Representation

In many ways, this is a more obvious encoding, though the operations are chosen to reflect the intuition that order and adjacency information should generally be maintained from old solution(s) to the new ones they give rise to. A sequence of facts is represented simply as that sequence. Mutation selects a random element, removes it from the sequence and then inserts it again in a random place. Crossover inserts a random subsequence of one solution into another, deleting duplicates

Subtree Swapping				2000 Iterations		4000 Iterations	
Item	facts	elabs	rels	μ	σ	μ	σ
j-342540	28	298	13	-38.9	27.7	-15.0	39.3
j-990302	25	297	13	18.5	32.6	31.6	27.9
j-990811	24	274	6	-50.7	33.6	-2.2	27.6
Ordinal Representation				2000 Iterations		4000 Iterations	
Item	facts	elabs	rels	μ	σ	μ	σ
j-342540	28	298	13	110.2	25.6	127.3	26.1
j-990302	25	297	13	109.2	13.6	115.0	18.7
j-990811	24	274	6	57.0	17.6	66.7	17.8
Path Representation				2000 Iterations		4000 Iterations	
Item	facts	elabs	rels	μ	σ	μ	σ
j-342540	28	298	13	158.4	22.7	171.3	20.1
j-990302	25	297	13	175.0	19.3	192.9	13.7
j-990811	24	274	6	90.7	11.4	104.0	17.3

Figure 5: Results for 3 Algorithms

that occur outside the inserted subsequence.

6.3 Results

Figure 4 shows an example text produced using the path encoding operations (for `j-342540`, after 2000 iterations, just under 2 minutes, score 180). The same remarks about hand editing apply as before.

Figure 5 summarises the results for subtree swapping and the two genetic algorithms on a set of examples. These results summarise the mean and standard deviations of the scores of the system run 10 times. The system was tried with a limit of 2000 and 4000 iterations around the main loop of the algorithm. These took about 2 and 4 minutes respectively. With each example problem we have specified the number of facts, the number of elaboration relations and the number of non-elaboration relations. Note that there is not a very clear basis for comparison between algorithms, since each algorithm performs different operations during an "iteration". Nevertheless, since iterations take roughly the same amount of time one can get a rough idea of the relative performance.

The example text is now in a single paragraph, with a clear link from each sentence to the previous ones. From the numerical results, one can see that there is much less variability than before. This is mainly because the rigid tree-building constraints prevent really bad trees being built and so the worst results are less bad. The results are also significantly better than for subtree swapping, with the edge-sensitive representation clearly winning.

7 Discussion

It is necessary to be careful in evaluating these results. The results are only as good as the evaluation function, which is certainly flawed in major ways. The texts are of a specific type, there are only three of them and we have not used all rhetorical relations. The evaluations are especially limited by the fact that there is no account taken of the possibilities for aggregation, embedding etc. in the trees that are produced. Nevertheless the approach looks promising enough that it is a real candidate to be used with the ILEX system. Future work needs to look at improving the characterisation of good trees and if possible introducing more natural crossover/mutation operations.

8 Acknowledgements

The ILEX project is supported by EPSRC grant GR/K53321. We acknowledge the valuable assistence of the National Museums of Scotland and the useful advice of Andrew Tuson.

References

[Goldberg 89] Goldberg, D., *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison-Wesley, 1989.

[Hovy 90] Hovy, E., "Unresolved Issues in Paragraph Planning", in Dale, R., Mellish, C. and Zock, M., *Current Research in Natural Language Generation*, Academic Press, 1990.

[Mann and Thompson 87] Mann, W. and Thompson, S., "Rhetorical Structure Theory: Description and Construction of Text Structures", in Kempen, G., Ed., *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics*, Dordrecht: Nijhoff, 1987.

[Marcu 96] Marcu, D., "Building up Rhetorical Struicture Trees", *Proceedings of AAAI-96*, American Association for Artificial Intelligence, 1996.

[Marcu 97] Marcu, D., "From Local to Global Coherence: A Bottom-up Approach to Text Planning", *Proceedings of AAAI-97*, American Association for Artificial Intelligence, 1997.

[Mellish et al 98] Mellish, C., O'Donnell, M., Oberlander, J. and Knott, A., "An Architecture for Opportunistic Text Generation", paper submitted to INLGW-98.

[Michalewicz 92] Michalewicz, Z., *Genetic Algorithm + Data Structures = Evolution Programs*, Springer Verlag, 1992.

[Moore and Paris 93] Moore, J. and Paris, C., "Planning Texts for Advisory Dialogues: Capturing Intentional and Rhetorical Information", *Computational Linguistics* Vol 19, No 4, 1993.

[Scott and de Souza 90] Scott, D. and de Souza, C., "Getting the Message Across in RST-Based Text Generation", in Dale, R., Mellish, C. and Zock, M., Eds., *Current Research in Natural Language Generation*, Academic Press, 1990.