Parsing Natural Language using Guided Local Search

Michael Daum and **Wolfgang Menzel** [1]

Abstract. In this paper an application of guided local search (GLS) to the problem of natural language parsing is presented. The given parsing approach is situated in a constraint based parsing paradigm [10] that allows natural language processing in a robust and resource adaptive way [16]. Some extensions of GLS are introduced, most notably a multi-threaded search where a couple of agents cooperate with each other in parallel, showing synergetic effects. The resulting algorithm is compared to competing techniques within the framework of weighted constraint dependency grammars [21]. An experimental evaluation shows GLS being on par with similar approaches [7].

1 INTRODUCTION

In the framework of weighted constraint dependency grammars (WCDG) [21] grammatical knowledge about well-formed utterances and their structural descriptions is given as a set of possibly contradictory requirements, and the parsing problem is formulated as a partial constraint satisfaction problem. This allows to apply solution methods like constraint propagation [27], tree search [18], tabu search [8] or genetic methods [5] to mention only some. The work presented here adds the novel but powerful heuristics of guided local search to the canon of solution methods tried so far.

Within the WCDG framework parsing is viewed as an optimization problem which is meant to determine the best solution with respect to a weight–accumulating measure. Even in spite of erroneous input or contradictory evidence solutions can be brought forth. Therefore problem solving is robust right away. Information within the constraint solver is propagated immediately, i.e. information from different sources is amalgamated to a consistent whole. Grammatical knowledge is modelled in a declarative way expressed by violable constraints in WCDGs so that a finegrained distinction between gramatical and non-grammatical utterances is available. Optimality described by means of constraints is also a central issue in related linguistic work [19, 12, 4].

Nevertheless, advanced natural language processing imposes its own requirements and limits to the applicability of specific parsing techniques, namely realtime performance, anytime properties [1], resource adaptability [15] and bounded incremental parsing. From the perspective of these requirements guided local search exhibits some advantages for natural language parsing: as a transformation based heuristic it posesses strong anytime properties by steadily improving a solution found so far; it only requires $O(n)$ memory; search diversifies and terminates with respect to its own performance profile. This last property holds potential to effectively bound the increase of computation time during incremental left-to-right parsing, a claim which is subject of future research.

[1] Natural Language Systems, Department of Computer Science, University of Hamburg, (micha|wolfgang)@nats.informatik.uni-hamburg.de

GLS was introduced in [25] as a direct successor of GENET [3], a neuronal network architecture to solve constraint satisfaction problems (CSPs). Since then it was applied successfully to a series of problems [17, 13, 26]. Adapting the penalty based approach GLS offers a method to solve combinatorial optimization problems in general and partial constraint satisfaction problems (PCSPs) in particular. The application of guided local search to natural language processing was first described in [23].

2 FOUNDATIONS

2.1 Guided Local Search

GLS is a meta-heuristic, like tabu search or simulated annealing, which tries to guide a local search out of local optima while investigating a combinatorial optimization problem. Figure 1 illustrates how this heuristic works.

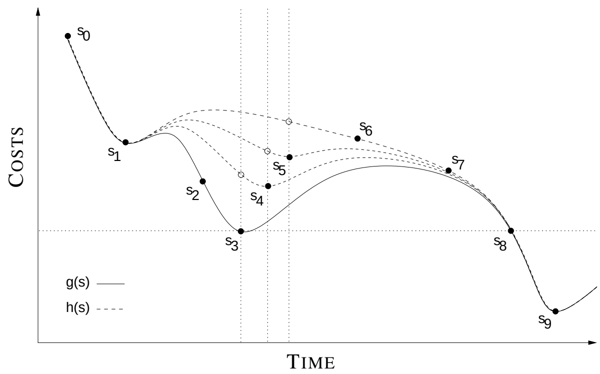

Figure 1. GLS escaping from a local minimum

By definition, local search only considers *neighbors* in a search space of feasible solutions. Two solutions s_i and s_j are considered *neighbors* if s_i can be transformed to s_j in one step and vice versa. Local search then carries out only those transformations which lead to an immediate improvement with respect to an objective function. Inevitably, local search terminates in local optima like s_3 in Figure 1. At that point local search hands over control to the meta-heuristic. GLS alters the objective function itself that is used to compute the costs incurred by this local optimum. This is done by penalizing a set of *solution features* of s_3, that is by learning the characteristics of local optima in the search space. We derive a function h by adding a penalty term to the objective function g. Now local search uses the *augmented function* h instead of g to escape from the local optimum. We repeat this procedure until an external termination criterion is met.

Thus guided local search consists of the following four parts:

- a set $F = \{f_1, f_2, \ldots, f_n\}$ of features used to characterise local optima; let $f_i(s) = 1$ if feature f_i holds in solution s, and 0 otherwise.
- a set $C = \{c_1, c_2, \ldots, c_n\}$ of costs for every feature f_i
- a set of penalty weights $P = \{p_1, p_2, \ldots, p_n\}$ for every feature f_i
- a regularisation parameter λ

The augmented objective function g is defined as

$$h(s) = g(s) + \lambda \cdot \sum_{f_i \in F} p_i \cdot f_i(s) \,. \tag{1}$$

Penalty weights are increased when a repair seems most useful regarding its severity and the amount of previous work which was spent to avoid the corresponding features. Therefore the utility of a feature can be defined as

$$util(s, f_i) = f_i(s) \cdot \frac{c_i}{1 + p_i} \tag{2}$$

and we compute the set of features maximizing this function.

2.2 Dependency parsing using WCDG

The application of constraint satisfaction techniques for parsing has its roots in [14] and was later extended in [9]. Given the german sentence

ich denke wir treffen uns um zehn. (I think we meet at ten.) (3)

we are interested in finding a structural analysis as shown in Figure 2. The upper part represents a traditional surface syntactic structure

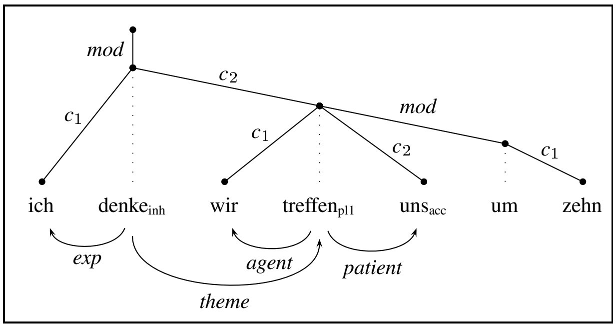

Figure 2. Dependency analysis of sentence (3)

consisting of dependencies between word forms labelled by the type of subordination. In addition a semantic functor–argument–structure is modelled on independent levels of the constraint optimization problem. These relationships are depicted as arrows below the surface structure. The different levels are mapped onto each other using constraints as well. Note that using a weighted constraint grammar (WCDG) we are not restricted to a specific dependency theory as long as it considers only relations between word forms. A WCDG is defined as a 4–tuple $\langle \Sigma, R, L, C \rangle$ where

- Σ is a lexicon of known words, like $\{\text{ich}, \text{denke}_{\text{appo}}, \text{denke}_{\text{inh}}, \text{treffen}_{\text{inf}}, \text{treffen}_{\text{pl1}}, \ldots\}$,
- R is a set of language levels, e.g. $\{\textit{syn}, \textit{agent}, \textit{exp}, \ldots\}$,
- L is a set of labels that indicate the type of a subordination, e.g. $\{c_1, c_2, \textit{mod}, \textit{link}, \ldots\}$
- and C is a set of constraints expressing wellformedness conditions; let $C_i \in C$ be a function where $C_i(s) \in [0.0..1.0[$ whenever s violates the constraint, and be 1.0 otherwise.

The dependency tree in Figure 2 is thus encodeable as a list of edges

$$\begin{aligned} s \;=\; & \{\langle \textit{syn}, \text{ich}, c_1, \text{denke}_{\text{inh}}\rangle, \langle \textit{syn}, \text{denke}_{\text{inh}}, \textit{mod}, \text{nil}\rangle, \\ & \langle \textit{syn}, \text{wir}, c_1, \text{treffen}_{\text{pl1}}\rangle, \langle \textit{syn}, \text{treffen}_{\text{pl1}}, c_2, \text{denke}_{\text{inh}}\rangle, \\ & \langle \textit{syn}, \text{uns}_{\text{acc}}, c_2, \text{treffen}_{\text{pl1}}\rangle, \langle \textit{syn}, \text{um}, \textit{mod}, \text{treffen}_{\text{pl1}}\rangle, \\ & \langle \textit{syn}, \text{zehn}, c_1, \text{um}\rangle, \langle \textit{exp}, \text{denke}_{\text{inh}}, \textit{link}, \text{ich}\rangle, \\ & \langle \textit{theme}, \text{denke}_{\text{inh}}, \textit{link}, \text{treffen}_{\text{pl1}}\rangle, \\ & \langle \textit{agent}, \text{treffen}_{\text{pl1}}, \textit{link}, \text{wir}\rangle, \\ & \langle \textit{patient}, \text{treffen}_{\text{pl1}}, \textit{link}, \text{uns}_{\text{acc}}\rangle, \ldots\} \end{aligned} \tag{4}$$

where one edge $e \in s$ is described as a 4–tuple $e \in R \times \Sigma \times L \times \Sigma$.

Mapping a WCDG to a partial constraint satisfaction (PCSP) is straightforward now. Recall the definition of a PCSP as a 4–tuple $\langle Z, D, C, g \rangle$ where

- Z is a finite set of variables,
- D is a finite set of domains D_i holding all possible values assignable to variables $z_i \in Z$,
- C is the set of constraints on variable assignments
- and g is the objective function on variable assignments.

So we map the words on each level of a given input utterance to the variables of the PCSP. The object is to find a variable assignment which is optimal with regards to all constraints C where the objective function is defined as

$$g(s) = \prod_{C_i \in C} C_i(s) \,. \tag{5}$$

In general we never feed a complete solution s into one constraint but prefer to judge at most two edges of the dependency structure. Thus the set of constraints C only consists of unary constraints C_i^1 and binary constaints C_i^2. The resulting limitation of the expressiveness is payed off by computational advantages [22].

3 APPLYING GLS

When applying an algorithm design technique like GLS to dependency parsing one has to answer a series of questions:

- How can dependency structures be transformed into each other?
- What is the neighborhood of a dependency structure?
- How can we devise an efficient local search method?
- What are reasonable solution features of dependency structures?
- When should GLS be terminated?

In what follows we cover some of these questions while not discussing thoroughly every decision that we made.

3.1 Neighborhood and local transformations

The process of parsing an utterance in WCDG consists of two phases: a problem generation phase and a search phase. In the first phase we build up the PCSP so that we generate all values of all domains in D. Thus the space of feasible solutions is given by

$$S = D_1 \times D_2 \times \ldots \times D_n \qquad \text{where} \quad D_i \in D \,. \tag{6}$$

All values in the domains are checked locally by all unary constraints whereby only those values with $g(\{e\}) > 0.0$ $(e \in D_i)$ are incorporated.

Some obvious transformations are possible when we look at the representation (4) of dependency relations, namely

- exchanging a lexical reading by looking up lexical ambiguities
- subordinating a word under a different modifiee
- changing the label of a subordination

We have to take two points into account. First these transformations can result in an unfeasible solution $s \notin S$ as there might not exist a corresponding variable value in the domains computed before. Secondly, variable assignments are not independent from each other. Choosing an assignment that uses a different lexical reading entails transformations of those dependency relations that still use conflicting readings. In the worst case a local transformation might end up changing all variable assignments.

Therefore we want to transform dependency relations by exchanging variable assignments only without manipulating the values themselves. The neighborhood of a dependency structure is not computed exhaustively for obvious reasons. Instead we use a greedy strategy using those values which are locally best in a domain. Thus we replace local search by a dependency construction procedure that is called by the meta-heuristic.

3.2 Solution features

In a PCSP the constraints themselves might be taken as solution features. So we could simply attach penalty weights to each constraint which are then increased during the search. Alternatively, domain values are used as solution features. The corresponding constraint violations are gathered for each variable so that variable assignments with maximum costs are diagnosed as *hot spots*. In this work we prefer the latter approach as the set of weights is directly linked to the problem size. Thus we allocate penalty weights for each possible value of the PCSP.

In a local optimum, when our greedy search terminates the set of hot spots is computed as described above and all values assigned in a hot spot are penalized. Note that a violated binary constraint C_i^2 distributes its costs to both involved variable assignments even if only one of the involved values is responsible for the violation. A closer look reveals that even unary constraints might be violated for quite different reasons, e.g. the label of an edge or a lexical reading was wrongly chosen. It can be assumed that this impreciseness is leveled in the long run after more knowledge about the search space has been gathered as penalty weights.

3.3 Performance profile and termination

In general GLS does not possess a built–in termination criterion besides the obvious ones like expiration of time (t_{max}) or the construction of a solution without constraint violations. So we observe the *performance profile* of the search, that is the quality improvement over time [20], in order to deduce an appropriate termination criterion. Figure 3 shows the measured performance profile of the sentence

mir wäre es dann lieber, wenn wir die ganze Sache auf den Mai verschieben. (I would prefer to defer the affair until may.) (7)

taken from a german corpus of appointment scheduling dialogues [6]. The two curves in this figure show the costs g and the augmented costs h as a function of time. We can see that those two curves diverge over time while penalty weights are increased. The dots in Figure 3 indicate the best solutions found so far. The conspicuous jump of h after eight seconds is caused by the solution found there, whose costs allow to prune the search space S locally, i.e. all variable assignments are omitted that are worse than the best solution. As a result, the set of penalty weights left to learn is reduced as well so that we get this characteristic performance profile.

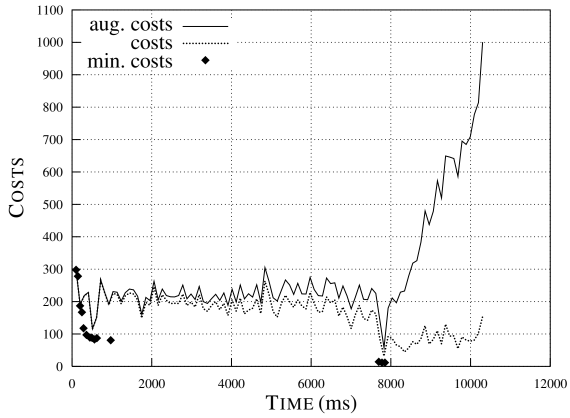

Figure 3. Costs during computation of (7)

The augmented costs in Figure 3 can be seen as a typical performance profile occuring in our experiments. Thus, we want to distinguish different *time phases* of the computation:

- The *initialization phase* at the beginning of a computation, where the improvement rate typically is very high.
- A *prolongation phase*, where penalty weights are learnt in order to further improve the best solution found in the initialization phase. Here, the improvement rate is rather low.
- The *termination phase* reached after a (near) optimal solution is found and the rest of the time is needed to assure that no better solutions are easily available.

Simple heuristics allow to determine the current phase of the analysis during an ongoing computation. This classification might be typical in transformational optimization processes and is inspired by a similar terminology from micro-biology [11].

The distinction of the tree phases provides us with a couple of advantages: (1) by limiting the augmented costs h in the termination phase to h_{max} a very effective termination criterion is available; (2) the communication overhead between two parallel searches as described in section 4 might be limited during the initialization phase when the agents are able to improve local solutions on their own; (3) furthermore, the regularization parameter λ might be chosen differently per phase: small values of λ could be of advantage in the initialization while greater values are used in the termination phase. The potential of such an approach needs to be investigated further.

4 EXTENSIONS

In addition to the already mentioned alternative termination criterion we implemented a couple of major and minor add-ons to the basic GLS, namely

- *reinforcement*: we define favoured features that reduce the costs instead of rising them in the augmented objective function (1)
- *local pruning*: domains D_i are sorted by augmented costs; a given percentage of highly expensive values is cut away regularly.
- *normalization*: we regulate the *importance* of a constraint besides its costs in order to manipulate the search focus.

Furthermore we investigated a parallel search algorithm called *multi-threaded guided local search* (MGLS) based on cooperation among several sequential search agents. This kind of parallelization differs from a data driven or functional parallelization where the search space is portioned. In a cooperative search the complete problem is made available to all the agents which can use different heuristics to explore the search space. One of the advantages of this kind of parallelization is that communication can be reduced to broadcasting best solutions found locally. We distinguish two modes of communication within MGLS in order to regulate how foreign information is used by an agent:

- *compliance*: a new best solution is broadcasted entirely to every agent; an agent adopts this solution and investigates it with his own parametrization
- *competition*: only the quality of a new best solution is broadcasted; agents might benefit from this information by locally pruning the search space

Both modes of communication are well motivated: in a compliant mode agents might escape from unpromising parts of the search space with the help of other agents. But this could also have the negative effect of being driven away from near optimal solutions. Such a distraction can be avoided in a competitive way of computation. So far, MGLS has only been simulated in a round-robin fashion on a single processor.

A similar approach of cooperative computation using tabu search was reported in [2]. A description of *systemic behavior*, that is the impact of cooperating agents on the optimization process, is given in [24].

5 EXPERIMENTS

In the experiments we used a weighted constraint grammar covering 222 sentences of the Verbmobil corpus [6]. All computations were carried out on a Pentium III 500 Mhz. Preliminary experiments were used to determine penalty regularizations $\lambda \in [3.0..3.2]$ and $h_{max} = 1000$. A limit of the execution time $t_{max} = 100$s was chosen for the given platform noticing only 7 sentences exceeding this threshold. All other computations terminate because of expiring augmented costs as shown in Figure 3.

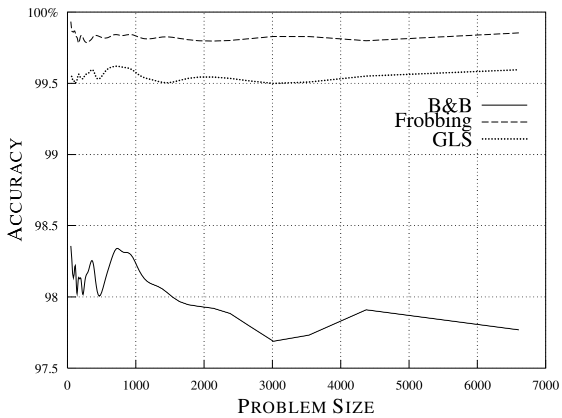

Figure 4. Accuracy of different solution methods

GLS has been compared to two other solution methods for WCDG, namely branch and bound, an agenda-based tree search, and

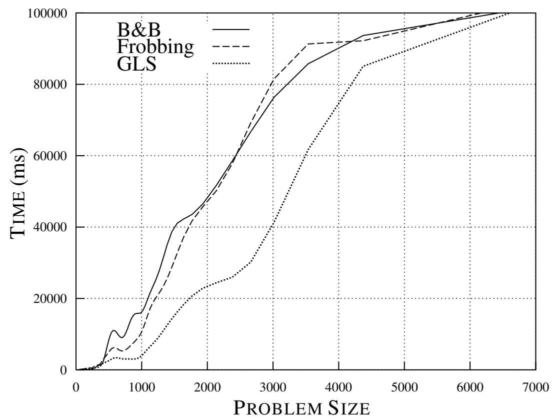

Figure 5. Performance of different solution methods

Frobbing [7], a transformation based algorithm related to tabu search. In order to yield comparable results, execution time for branch and bound and Frobbing was limited to 100s as well. The agenda size was limited to 3000 entries.

Figure 4 shows the accuracy of the final solutions depending on the problem size which is defined as $\sum_{D_i \in D} \mid D_i \mid$. All curves were smoothed indicating the mean accuracy. We see that GLS is able to provide a high accuracy which, however, is still outdone by Frobbing. On the other hand, GLS outperforms both other solution methods for all problem instances in terms of computation time (see Figure 5). Branch and bound did not show up well in our experiments which is due to its limited agenda size and execution time. Experiments in [21] show that a better accuracy is attained by a different agenda sorting strategy.

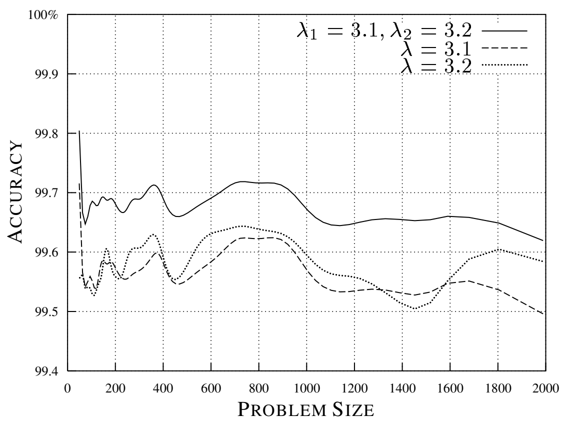

Figure 6. Accuracy of an MGLS compared to its parts

The last experiment discussed here is shown in Figures 6 and 7. We tested MGLS using two compliant agents parametrized with $\lambda_1 = 3.1$ and $\lambda_2 = 3.2$ and compared accuracy and performance to the agents runing in isolation. As noted before, MGLS only simulates concurrency. So it is not surprising to see no speedup in Figure 7. On the other hand, the execution time of the combined agents is strictly lower than the sum of the execution time of the isolated computations. This is accompanied by the observation shown in Figure 6 that the MGLS provides higher accuracy than its components in isolation. Even if we had coupled both agents by an ideal oracle choosing the

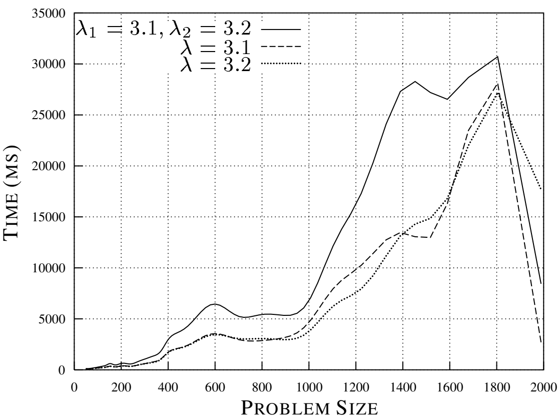

Figure 7. Performance of an MGLS compared to its parts

higher quality solution at the end of the computation, no comparable accuracy could have been obtained.

6 CONCLUSIONS

Recent advances in the field of combinatorial optimization have been carried over to natural language processing combining weighted constraint dependency grammars and guided local search in a fruitful way. We adapted the original framework using a greedy local search and improved the termination behavior based on observations about individual performance profiles where three characteristic phases of a transformational optimization process can be clearly distinguished.

Since GLS is able to improve feasible solutions over time it possesses strong anytime properties. Furthermore, cooperative search has shown to be a promising approach to concurrency. Our experiments encourage the assumption that a cooperative parallel search might not only speed up the computation but also contributes to a higher quality of solutions.

Further investigations will focus on the potential of GLS for left-to-right incremental parsing. In such a setting, new penalty weights that are added to a dynamically expanding problem are initially untrained and thus force the search to focus on them. Consequently, older dependency structures can be expected to fade away naturally so that an exponential increase of the computational effort can be avoided.

ACKNOWLEDGEMENTS

We like to thank all members of the project team, in particular Ingo Schröder, Kilian Foth and Horia F. Pop for their contributions, fruitful discussions and the stimulating atmosphere.

REFERENCES

[1] Mark Boddy and Thomas Dean, 'Solving time-dependent planning problems', in *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-89)*, (August 1989).

[2] Teodor Gabriel Crainic and Michel Gendreau, 'Towars an evolutionary method – cooperating multi-thread parallel tabu search hybrid', in *Processes of the 2nd International Conference on Meta-Heuristics*, (July 1997).

[3] Andrew J. Davenport, Edward P.K. Tsang, C. J. Wang, and K. Zhu, 'GENET: A connectionist architecture for solving constraint satisfaction problems by iterative improvement', in *Proceedings of the 12th National Conference on Artificial Intelligence*, volume 1, pp. 325–330, (1994).

[4] Denys Duchier, 'Axiomatizing dependency parsing using set constraints', in *Proceedings of the 6th Meeting on Mathematics of Language*, pp. 115–126, Orlando, FL, USA, (1999).

[5] Lawrence Davis (Ed.), *Genetic Algorithms and Simulated Annealing*, Research Notes in Artificial Intelligence, Morgan Kaufmann, Inc., Los Altos, CA, 1987.

[6] Wolfgang Wahlster (Ed.), *Verbmobil: Foundations of Speech-to-Speech Translation*, Springer-Verlag, Berlin, Heidelberg, 2000.

[7] Kilian Foth, Wolfgang Menzel, and Ingo Schröder, 'A transformation-based parsing technique with anytime properties', in *Proceedings of the 4th International Workshop on Parsing Technologies*, pp. 89–100, Trento, Italy, (2000).

[8] F. Glover, 'Tabu search - part I', *ORSA Journal on Computing*, **1**(3), (1989).

[9] Mary P. Harper and Randall A. Helzerman, 'Extensions to constraint dependency parsing for spoken language processing', *Computer Speech and Language*, **9**(3), 187–234, (1995).

[10] Johannes Heinecke, Jürgen Kunze, Wolfgang Menzel, and Ingo Schröder, 'Eliminative parsing with graded constraints', in *Proceedings of the Joint Conference COLING/ACL '98*, Montréal, Canada, (1998).

[11] Eric R. Kandel, James H. Schwartz, and Thomas M. Jessell, *Essentials of Neural Science and Behavior*, Appleton & Lange, 1995.

[12] Fred Karlsson, 'Constraint grammar as a framework for parsing running text', in *Proceedings of the 13th International Conference on Computational Linguistics (COLING-90)*, pp. 168–173, Helsinki, Finland, (1990).

[13] P. Kibly, P. Prosser, and P. Shaw, 'Guided local search for the vehicle routing problem', in *Processes of the 2nd International Conference on Meta-Heuristics*, (July 1997).

[14] Hiroshi Maruyama, 'Structure disambiguation with constraint propagation', in *Proceedings of the 28th Annual Meeting of the ACL*, pp. 31–38, Pittsburgh, (1990).

[15] Wolfgang Menzel, 'Parsing of spoken language under time constraints', in *Proceedings of the 11th European Conference on Artificial Intelligence*, pp. 560–564, Amsterdam, (1994).

[16] Wolfgang Menzel and Ingo Schröder, 'Error diagnosis in a language tutoring system', *Special ReCALL publication: Language Procesing in CALL*, 20–30, (1999).

[17] Patrick Mills and Edward P.K. Tsang, 'Guided local search applied to the SAT problem', Technical report, Department of Computer Science, Universit of Essex, Colchester, UK, (1999).

[18] B. A. Nadel, *Tree search and arc consistency in constraint satisfaction problems*, Springer Verlag, New York, 1988. in L. Kanal und V. Kumar, Hrsg.

[19] Alan Prince and Paul Smolensky, 'Optimality theory: constraint interaction in generative grammar', Technical Report Report Nr. 2, Rutgers University Center of Cognitive Science, Piscataway, New York, (1993).

[20] Stuart J. Russell and Shlomo Zilberstein, 'Composing real-time systems', in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 91)*, pp. 212–217, Sydney, Australia, (1991).

[21] Ingo Schröder, *A Framework for Gradation in Natural Language Analysis Using Constraint Optimization*, Ph.D. dissertation, University of Hamburg, Department of Computer Science, forthcoming.

[22] Ingo Schröder, Wolfgang Menzel, Kilian Foth, and Michael Schulz, 'Modeling dependency grammar with restricted constraints', *International Journal Traitement Automatique des Langues: Grammaires de dépendance*, **41**(1), 97–126, (2000).

[23] Michael Schulz[2], *Parsen natürlicher Sprache mit gesteuerter lokaler Suche*, Diplomarbeit, University of Hamburg, Department of Computer Science, 2000.

[24] Michel Toulouse and Teodor Gabriel Crainic, 'An experimental study of systemic behavior of cooperative search algorithms', in *Processes of the 2nd International Conference on Meta-Heuristics*, (July 1997).

[25] Christos Voudouris, *Guided Local Search for Combinatorial Optimisation Problems*, Ph.D. dissertation, Department of Computer Science, University of Essex, Colchester, UK, 1997.

[26] Christos Voudouris and Edward P.K. Tsang, 'Function optimizing using guided local search', Technical Report CSM-249, Department of Computer Science, University of Essex, Colchester, UK, (September 1995).

[27] David Waltz, 'Understanding line drawings of scenes with shadows', in *The Psychology of Computer Vision*, ed., P. H. Winston, McGraw–Hill, New York, (1975).

[2] The author just married and is now known as Michael Daum.