Applying a Parallel Any–Time Control Algorithm to a Real–World Speech Understanding Problem

J. Fischer, H. Niemann

Lehrstuhl für Mustererkennung (Informatik 5)
Universität Erlangen-Nürnberg
Martensstr. 3, D – 91058 Erlangen
E-mail: fischerj@informatik.uni-erlangen.de

Abstract

In this paper we show how to manage speech understanding using a control algorithm which enables parallel processing and is based on iterative optimization.

The task specific knowledge is represented by a semantic network consisting of concepts representing objects, and links. An optimal interpretation of a spoken utterance is available if a best scored instance of a goal–concept can be computed. To enable an efficient parallel computation, the concept centered network is converted to an attribute centered task–graph. For computing an optimal instance of the goal–concept, the task–graph has to be computed bottom–up until a heuristic judgement function has been optimized. Several instances of concepts can be computed on parallel, e.g. on a local network of heterogeneous workstations. The any–time capability of the control algorithm is provided by the use of iterative optimization methods for the search of a best fitting instance of a goal concept.

1 INTRODUCTION

Automatic recognition of complex patterns, specifically spontaneous speech and motion images, will become increasingly relevant for a great number of applications in the future. Such applications include service robots to aid handicapped individuals, multi–modal telecooperation tasks, etc. In order to apply such systems to the real world, real–time as well as any–time capabilities are indispensable.

Parallel processing is a promising means to achieve the desired speed, iterative methods may provide an any–time behaviour. A variety of parallel algorithms for problems from data–driven processing have been developed, especially in image processing [1]. In contrast, parallel symbolic processing is much less investigated, although some major problems of the field, like e.g. parallel knowledge representation [2], are discussed in the literature.

Recently, an approach to knowledge–based pattern understanding based on iterative optimization methods and allowing an efficient parallel computation has been developed and successfully tested on a small image understanding task [3, 4]: recognition of streets from TV image sequences recorded in a moving car. Three image sequences, each consisting of 30 gray–level images, were used. The goal of the analysis was to obtain a description of the road and its markers.

In this paper, we propose the application of this control algorithm to a real–world speech understanding problem, using as a framework a dialog system which is able to answer inquiries about the German InterCity train time–table. In section 2 the basic idea of the algorithm will be shown. A survey of the knowledge representation formalism will also be given. In section 3 we will present the speech understanding task and how this algorithm has been applied to it, and in section 4 we will show first results on corpora of read and spontaneously spoken utterances. The paper ends with a Conclusion and Outlook in section 5.

2 A PARALLEL ANY–TIME CONTROL ALGORITHM

The control algorithm presented in this paper was developed for the use with knowledge represented in a semantic network. Semantic networks were introduced by the end of the sixties as a coarse

[1]This work was supported by the Real World Computing Partnership (RWCP).

model of the human mind [5]. Information about a general idea (object, event, etc.) is represented by nodes, relations between those ideas are modeled by links. The semantic network formalism which we use, ERNEST(**ER**langer **NE**tzwerk Sy**ST**em) [6], provides three types of nodes: *concepts*, representing a general idea, (e.g. Noun), *modified concepts*, which are concepts restricted by the value of some of its attributes (e.g. Noun; number: plural), and *instances*, representing a concrete realization of a concept (e.g. "train"). Relations between the nodes are established by three types of links:

- *part links*, which link a concept to the simpler concepts it consists of,
- *concrete links*, linking concepts of different levels of abstraction, and
- *specialization links*, which are used to establish inheritance of attributes, relations and links from more general concepts to more special ones.

The main components of a concept C are its *parts* P and its *concretes* K. Furthermore, a concept has a set of *attributes* A and *structural relations* S between these attributes, which are computed during analysis by referred functions F. Each concept refers also to a function which computes during analysis the score $\boldsymbol{G}$ of a modified concept or an instance of C. Since there may be different realizations of the same "object", *modalities* H_l were introduced into the knowledge representation formalism[2], each one defining a permissible combination of *obligatory* and *optional* parts and concretes of the object modeled by concept C. For example, the concept SY_NG representing a noun group may consist of a proper noun by its own (e.g. "Berlin") or of the combination of an article, an adjective, and a noun (e.g. "the next train"). SY_NG would, in this case, be defined by two modalities:

$H_1^{(SY\_NG)}$: {obl. *proper noun*} and

$H_2^{(SY\_NG)}$: {obl. *noun;* opt. *article, adjective*}.

We have seen above, how knowledge is represented by the semantic network formalism of ERNEST. For the knowledge to be used by a pattern understanding system, a control algorithm is necessary. The general task to be managed by the control algorithm is the computation of a symbolic description $\mathcal{B}$ of the observed data $\boldsymbol{f}(\boldsymbol{x})$, which

- *optimally* fits to the observed data and
- is *maximally* consistent with internally represented task–specific knowledge.

In ERNEST, the goal of analysis itself is represented by one or more concepts which are denoted *goal concepts* C_{g_i}. The description of the observed data is then represented by an instance $I(C_{g_i})$ of the goal concept. Since every concept refers to a function which computes a score $\boldsymbol{G}$ of an instance or a modified concept of it, there is also a score $\boldsymbol{G}(I(C_{g_i}))$. Now, for solving the analysis task, one can request the computation of an *optimal* instance $I^*(C_{g_i})$:

$$\begin{aligned} \mathcal{B}(\boldsymbol{f}(\boldsymbol{x})) &= I^*(C_{g_i}) \\ I^*(C_{g_i}) &= \underset{\{I(C_{g_i})\}}{\operatorname{argmax}} \{\boldsymbol{G}(I(C_{g_i}))|\mathcal{M},\mathcal{A}\} \end{aligned} \tag{1}$$

which means: *Generate an instance I^* for a goal concept C_{g_i} with maximal score $\boldsymbol{G}$ using the initial symbolic description $\mathcal{A}$ and the internal model $\mathcal{M}$, which is a network of concepts $\mathcal{M} = \langle C_k \rangle$.*

For computing an instance of a concept $I(C_k)$, it is necessary to compute instances of all of its obligatory parts and concretes for one modality $H_l^{(k)}$, values for all of its attributes and relations, and finally the score $\boldsymbol{G}(I(C_k))$. Facing both, the large amount of data and the limited processing time in most pattern understanding tasks, the exploitation of parallelism provides a promising way to compute an optimal instance $I^*(C_{g_i})$ in time with the sensory input.

In the approach proposed here, parallelism is exploited on two levels: on *knowledge level* and on *control level* (see below). To allow an efficient exploitation of parallelism, the concept–centered knowledge base is compiled into a *fine–grained task graph* on sub–conceptual level, and the pattern understanding problem is defined as a *combinatorial optimization* problem solved by means of *iterative* optimization methods, like e.g. *threshold acceptance* and *great deluge algorithm* [7], or *genetic algorithms* [8]. In each iteration step, a partial solution (i.e. an instance of a goal concept which approximately meets the requirements

[2] This was done in order to keep the knowledge base small. Another possibility would be to define one concept for each different realization of an object. This would, however, result in a very large and unwieldy knowledge base.

mentioned above) is computed. This results, per definition, in the *any–time* behaviour of the algorithm, since a coarse solution is obtained if less computation time is available, and a refined solution is obtained if more iteration steps can be performed.

Parallelism on knowledge level refers to the parallel computation of one instance of a goal concept. One possibility to exploit parallelism on this level is given by employing an isomorphic mapping between the processors of a parallel hardware and the nodes and links of a knowledge base (e.g. [9]). This turned out to be a feasible approach if both concepts and inferences are simple. In our approach, however, concepts may be complex and become a bottleneck in instantiation, since in the ERNEST formalism a concept may have an arbitrary number of attributes and relations.

To get around this problem, one compiles in a pre–analysis step the knowledge base into a fine–grained, attribute centered, and acyclic *task graph* $D = (V, E)$. Each node $v_i \in V$ represents an attribute, a relation, or the judgment of a concept. If node v_i is an argument of the procedure that computes the value for node v_j, a directed link $E_{ij} = (v_i, V_j) \in E$ is created to express that computation of node v_i has to be finished before computation of v_j may start. The acyclic graph may be mapped to a multiprocessor system for parallel processing (cf. [4]). Two steps are necessary to compute the task graph:

1. A recursive *top–down expansion* of the goal concepts $C_{g_1} \ldots C_{g_\kappa}$ by creating and linking modified concepts for all obligatory and optional parts and concretes (see Figure 1);

2. Creation of the task graph D by *splitting up* all concepts into their attributes, relations, judgments, etc. and linking them as described above.

The expansion procedure in step 1 is necessary because each concept appears only once in the knowledge base. For example, in Figure 1 each single syntactic concept has as a concrete a concept representing a word–hypothesis (H_WHYP). When computing an instance for e.g. SY_PREP (representing a preposition), an instance for the concept H_WHYP as a concrete of SY_PREP must be computed.

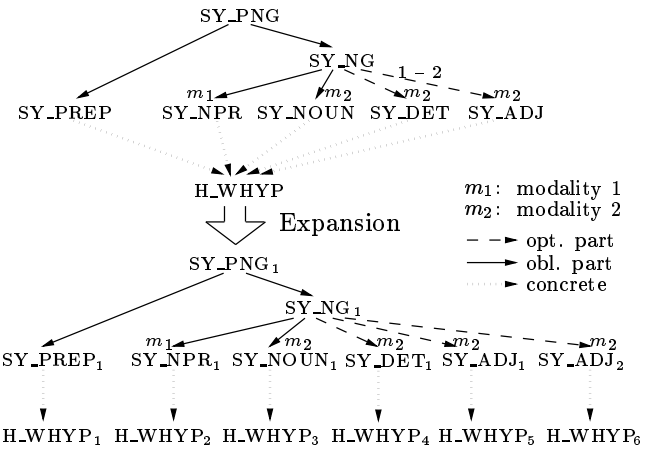

Figure 1: Example of the expansion mechanism for an excerpt of the knowledge base of EVAR (cf. section 3).

Nodes without predecessors represent (initial) attributes of concepts of lowest level of abstraction (e.g. concepts representing word hypotheses) and form the interface to the initial segmentation; nodes without successors represent the judgments of the goal concepts to be instantiated. By a bottom–up computation of all nodes of the task graph, an instance for each goal concept is computed.

Task of the control algorithm is, as mentioned before, to find the "best" interpretation of the segmentation objects, i.e. the instance of the goal concept with maximum score (in our speech understanding application, we so far only define one single goal concept, therefore we will refer in the following to "the goal concept C_g"). Competing instances arise from segmentation errors (e.g. errors in word recognition) and from ambiguous knowledge in the knowledge base, e.g. a concept C_k can be instantiated for each of its modalities $H_l^{(k)}$. So one can say that the instantiation of a goal concept C_g depends basically on:

- the assignment $(A_i, O_j), i = 1, \ldots, \mu$, of segmentation results O_j to some initial attributes A_i of primitive concepts, and

- the choice $(C_k, H_l^{(k)}), k = 1, \ldots, \lambda$, of a modality $H_l^{(k)}$ for each concept C_k that enables multiple definitions of an object.

In the approach presented here, the computation of a best scored instance is solved by combinatorial optimization. For that purpose, the cur-

rent *state of analysis* is summarized in a $(\mu + \lambda)$–dimensional vector

$$\boldsymbol{r} \quad = \quad ((A_i, O_j); (C_k, H_l^{(k)})) \qquad (2)$$

and the result of instantiation is rewritten as a function

$$\boldsymbol{g}(\boldsymbol{r}) \quad = \quad (\boldsymbol{G}(I(C_g))|\boldsymbol{r}), \qquad (3)$$

of the state vector $\boldsymbol{r}$.

For the reliable computation of a best scored instance $I^*(C_g)$ from the judgement vector in Equation (3) a cost function $\phi(\boldsymbol{r})$ is introduced (cf. [4]). The minimization of ϕ is done, as mentioned before, by means of iterative optimization. In each iteration step, exactly one state of analysis $\boldsymbol{r}$ is assigned to the task graph D. Then, by a bottom–up computation of D, the judgment $\boldsymbol{g}(\boldsymbol{r})$ and the costs $\phi(\boldsymbol{r})$ are computed. Iteration steps are performed until the cost function ϕ yields a value smaller then a chosen threshold T or as long as processing time is available. As a (partial) solution, the best scored instance $I_{best}(C_g)$ computed for $\boldsymbol{r}_{best}$ with minimal costs ϕ_{min} is reported.

Parallelism on control level can be exploited by a parallel computation of several competing instances (i.e. by a simultaneous evaluation of several states of analysis). This can be done by using e.g. the PVM (**P**arallel **V**irtual **M**achine, cf. [10]) on a local network of heterogeneous workstations. Each workstation computes the task graph D for a different state of analysis $\boldsymbol{r}$. In [3], results are reported using $p = 5$ workstations (HP 735) for the small image understanding task also mentioned in section 1.

Figure 2 shows the scheme for the parallel control algorithm. On knowledge level, the task graph D adapted to a specific state of analysis (e.g. initial attribute A_i is connected to segmentation object O_m, modality $H_l^{(k)}$ has been assigned to all nodes belonging to concept C_k, etc.) may be computed on parallel (*parallel data–driven instantiation*). On control level, D is computed simultaneously on different workstations $(\mathrm{WS}_i, i = 1 \ldots p)$ for several states of analysis (*parallel search*).

3 APPLICATION TO SPEECH UNDERSTANDING

As a framework for the application of the above

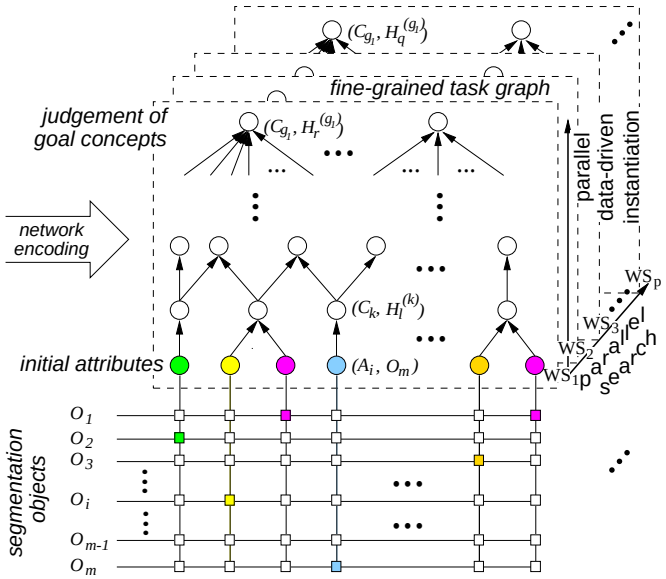

Figure 2: Scheme of the iterative parallel control algorithm. WS_i stands for the ith workstation in a network of heterogeneous workstations.

mentioned control algorithm to a real–world speech understanding problem, we use the dialog system EVAR (**E**rkennen "to recognize", **V**erstehen "to understand", **A**ntworten "to answer", **R**ückfragen "to ask back") [11], which is able to answer inquiries about the German train time–table. It is composed of two main modules: the acoustic processing (i.e. speech recognition) and the linguistic analysis (i.e. speech understanding) module.

The *acoustic processing* analyzes the speech signal and generates, by means of Hidden Markov Models and a stochastic grammar, a word hypotheses graph or best word chains. This word graph (or the best word chain) serves as input for the linguistic analysis component. For further details of the acoustic processing see [12].

The *linguistic analysis* of EVAR is built using the network formalism of ERNEST. The systems knowledge base is divided into procedural and declarative knowledge. The former consists of the procedures to calculate the values of attributes, relations, judgments, etc. The latter is built of concepts, which are arranged in the following *levels of abstraction*:

- *Word–hypotheses*: represents the interface between speech recognition and speech understanding; requests and verifies word hypotheses from the acoustic–phonetic front–

end.

- *Syntax*: represents syntactic constituents; generates syntactic constituents out of the set of word hypotheses.
- *Semantics*: models verb and noun frames with their deep cases; verifies the semantic consistence of the syntactic constituents, compounds them to larger ones, and performs task independent interpretation.
- *Pragmatics*: represents task dependent knowledge; interprets the constituents from the semantic module in the task–specific context.
- *Dialog*: models possible sequences of dialog acts; reacts in accordance to the interpreted intention of the spoken utterance.

In our approach, the original (sequential) control algorithm provided by ERNEST, which is based on a modified top–down/bottom–up A^*–algorithm, is being substituted by the parallel control algorithm described in section 2. This requires modifications of the systems knowledge base, e.g.:

- Since in the parallel control algorithm a strictly bottom–up instantiation is carried out, context has to be explicitly modeled in the knowledge base[3];
- Procedure interfaces have to be changed, since "search nodes" containing the actual information about the analysis do not exist anymore in the attribute–centered task–graph.

So far we have adapted the new control algorithm from word–hypotheses up to the dialog level, making it possible to instantiate a concept modeling an initial dialog step (i.e. a first request of the user), e.g.:

- "Good morning, I want to go to Munich tomorrow" or
- "Which train is leaving for Hamburg today after eight o'clock?"

[3]In the former A^* based control, context–information was propagated in a top–down step during analysis.

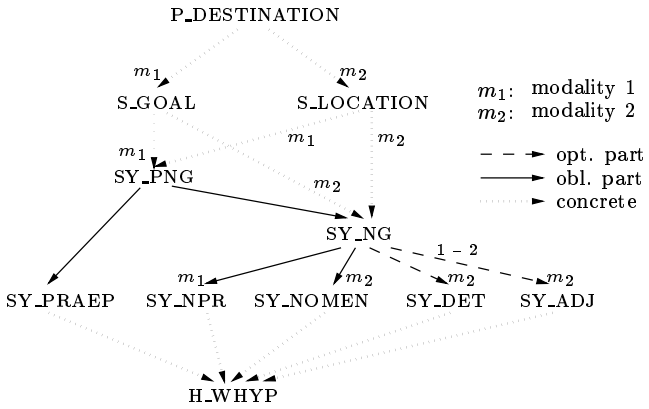

Figure 3: Excerpt of the knowledge base of EVAR.

Search space for our combinatorial optimization problem results in

$$|Z| = \prod_{i=1}^{\mu} \rho_i \cdot \prod_{k=1}^{\lambda} \theta_k \qquad (4)$$

ρ_i: Number of competing hypotheses for A_i

θ_k: Number of competing modalities for C_k.

In a real speech understanding problem, $|Z|$ is very large because of linguistic ambiguities and errors from the word recognition. A *reduction of search space* by a top–down/bottom–up propagation of intermediate results during analysis is not supported by our strictly bottom–up instantiation. Therefore, we developed and implemented a suited algorithm which makes a static top–down *propagation of linguistic constraints* before analysis. This algorithm is applied once to the task graph D during its generation, deleting nodes or restricting values of nodes.

The constraints which are propagated refer e.g. to modalities, values of attributes, etc. (e.g. an initial attribute node linked to a node on syntactic level representing a preposition will only be bound to a word which is a preposition). The deletion of nodes led to a reduction of the task graph by a factor of 6.5, reducing also the processing time for one iteration step by the same amount; furthermore, the modalities factor of search space was reduced from about 10^{75} to about 10^8.

Despite the propagation of linguistic constraints, search space is still very large. In order to improve the speed of convergence, a suited initialization of the state of analysis, which means to choose an initial state of analysis vector $\boldsymbol{r}_{init}$ which leads to costs $\phi(\boldsymbol{r}_{init})$ as close as possible

to the global minimum ϕ_{min}, seems to be promising. We therefore developed and implemented an initialization procedure based on heuristic rules and the incoming word chain or word hypotheses graph. This heuristic initialization was used for evaluating to what extend a well–aimed initialization influences the speed of convergence. Results of this experiments can be found in section 4.

4 EXPERIMENTAL RESULTS

Goal of the experiments was to evaluate the performance of the system with respect to its processing time and the amount of correctly analyzed pragmatic intentions[4], which are, for example:

We	want to go	to Hamburg	today .
TRAVELLER		DESTINATION	DEP_TIME

The environment of our experiments was the following:

- the **goal concept** of the task graph we used was a concept on dialog level modeling an initial users request;
- the task graph itself consisted of approximately **20 000 nodes**;
- the optimization method used was **stochastic relaxation** (cf. [3]);
- the optimization criterion was the **maximization of the number of covered words**;
- as input for the linguistic analysis we used the *spoken word chain.*

Stochastic relaxation was used since in [4] this method turned out to provide the best results (apart from genetic algorithms) for the examined image understanding task (cf. also section 1). Parallelization on control level was simulated for $p = 1 \ldots 5$ processors. The task graph was computed sequentially on each processor. Communication between processors occurred once after they had finished computation of a suboptimal instance for the goal concept (independent search). The test corpora used were:

- ASL–Süd test corpus of *read speech*, and
- EVAR–Spontan test corpus of *spontaneous speech.*

The ASL–Süd corpus was designed by different persons at different institutes and consists of first user requests of train time–table information. Out of this corpus, we chose 139 utterances which can be completely analyzed by the former EVAR system with the A^* based control algorithm. These 139 utterances contain a total number of 446 pragmatic intentions, for which instances of concepts have to be computed during analysis. Table 1 shows the number of correctly instantiated pragmatic concepts (in %) for this corpus of read speech. This results were achieved by using the heuristic initialization mentioned in section 3.

	correct pragmatic intentions (in %)				
n	p=1	p=2	p=3	p=4	p=5
1	89.0	92.6	92.8	94.2	94.4
5	89.0	92.8	93.0	95.2	96.2
10	86.8	94.6	95.7	97.1	97.5
25	90.4	95.9	97.5	98.7	98.9
50	94.4	97.5	98.7	98.9	99.3

Table 1: Percentage of correctly analyzed pragmatic intentions for n iterations and p processors; test corpus ASL–Süd .

The any–time behaviour of the system is confirmed in Table 1 by the fact that the more iteration steps the system performed, the more pragmatic intentions were analyzed correctly. The system is able to react after each iteration step according to the partial solution that was found at this point. After $n = 1$ and $p = 5$, 97% of all destination places were correctly instantiated. Recall that each utterance contains an average of 3.2 pragmatic intentions. One can say that in at least 97% (practically all) of the utterances the system is able to keep a dialog with the user by confirming the departure place and asking for a pragmatic intention it has not yet found, e.g. the departure time.

The advantage of parallelization on control level is shown by the fact that we get better results by performing several iteration steps on p =1, ..., $p = 5$ processors, i.e. the more processors we use, the faster (concerning the number of iter-

[4] These are pragmatic units the system needs to know in order to react to the users request, and are represented in EVAR by pragmatic concepts, cf. section 3

ation steps) the analysis converges. Since the task graph is initialized by a different state of analysis[5] for each processor used, performing $n = 1$ iteration step on $p = 5$ processors yields better results (94%) than performing $n = 5$ iteration steps on $p = 1$ processor (89%). At the moment, *processing time* amounts to an average of 0.5 seconds per iteration step, independent of the size of the utterance.

Table 2 shows that by applying the heuristic initialization an error reduction of 86% could be achieved after the first iteration step. Because of the linguistic restrictions (cf. section 3), results are relatively good after the first iteration step even without an initialization.

$n=1$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
– Init.	56.7	60.9	60.9	60.9	60.9
+ Init.	89.0	92.6	92.8	94.2	94.4

Table 2: Percentage of correctly analyzed pragmatic intentions *with* and *without* heuristic initialization of the state of analysis; test corpus ASL–Süd .

Table 3 shows the percentage of correctly analyzed pragmatic intentions for the EVAR–Spontan test corpus. The EVAR–Spontan corpus consists of about 1 000 real, spontaneous dialogs collected by our current train time–table information system connected to the public telephone line. Out of this corpus, we selected 435 first users utterances from the dialogs of one recording phase (phase 08, cf. [13], Page 154). The selected utterances contained at least one pragmatic intention out of the application domain. Utterances, like for example

- "Hello, are you a computer?" or
- "I want to go to hell"

have been excluded from the evaluation. The number of pragmatic intentions in these 435 selected utterances was 988.

One can see in Table 3, that on a spontaneous test corpus performance concerning the number of correctly analyzed pragmatic intentions decreases comparing to the results on the read test corpus

[5]Please note that by the heuristic initialization not all parameters of the state of analysis vector are determined; there are still parameters which are randomly set.

	correct pragmatic intentions (in %)				
n	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
1	78.2	80.6	80.9	86.3	86.5
5	78.5	81.1	81.6	87.7	87.9
10	80.9	83.8	84.4	87.9	87.9
25	84.3	87.3	88.6	88.6	88.9
50	85.1	88.6	88.9	89.2	89.3

Table 3: Percentage of correctly analyzed pragmatic intentions for n iterations and p processors; test corpus EVAR–Spontan .

ASL–Süd . This is due to the fact that grammatical irregularities of spontaneous speech are not yet sufficiently considered in the linguistic knowledge base of EVAR. The results are, even though, quite good. Here too, one can state the any–time behaviour and the advantages of parallelization on control level. Processing time was the same as compared to processing time for the ASL–Süd utterances. After $n = 1$ and $p = 5$, 94% of all destination places were correctly instantiated and in 95% of the utterances, at least 1 pragmatic intention has been found.

If one compares Table 2 and Table 4, one can observe that results without initialization are close to each other for both corpora. On the spontaneous speech corpus, however, reduction of error rate by initialization was of "only" 69% compared to 86% for ASL–Süd . This seems to be plausible and even quite good, given that the heuristic rules for initialization were developed based on the read corpus. Since an initialization showed to be very efficient, we are at present working on a well–aimed initialization of the state of analysis by using different stochastic methods, e.g. neural networks, polygrams, and classification trees.

$n=1$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
– Init.	54.9	55.8	56.1	56.3	56.3
+ Init.	78.2	80.6	80.9	86.3	86.5

Table 4: Percentage of correctly analyzed pragmatic intentions *with* and *without* heuristic initialization of the state of analysis; test corpus EVAR–Spontan .

Besides the initialization, which leads to a high percentage of correctly analyzed pragmatic intentions after the first iteration step, convergence speed can be further improved by implementing

an optimal procedure for choosing a new state of analysis after each iteration. Change of the state of analysis is done at the moment basically at random, and resulted in an average error decrease of 70% for ASL–Süd and 30% for EVAR–Spontan after 50 iteration steps (cf. tables 1 and 3). The large difference between the two corpora results, as mentioned before, from the fact that the spontaneously spoken corpus contained pragmatic intentions which are not yet modeled by the linguistic knowledge base of EVAR. It will be evaluated in the near future, to what extend the spontaneous utterances we used for evaluation of the parallel control algorithm can be analyzed using the former sequential A^* based control algorithm of EVAR.

5 CONCLUSION AND OUTLOOK

In this paper we proposed the application of a parallel any–time control algorithm for semantic network based pattern understanding to a real–world speech understanding problem. First experimental results showed the feasibility of the approach. Future work will concentrate on the extension of the control algorithm to all dialog steps, on the processing of word hypotheses graphs, and on the further improvement of processing time and convergence speed (cf. section 4). Moreover, an incremental processing of word hypotheses should be investigated for supporting a real–time performance and for a further improvement of the any–time behaviour.

References

[1] I. Pitas, editor. *Parallel Algorithms for Digital Image Processing, Computer Vision and Neural Networks,* Wiley Series in Parallel Computing. John Wiley & Sons, 1993.

[2] M. Evett, J. Hendler, and L. Spector. Parallel Knowledge Representation on the Connection Machine. In *Journal of Parallel and Distributed Computing*, pages 22(2): 168–184. 1994.

[3] H. Niemann, V. Fischer, D. Paulus, and J. Fischer. Knowledge Based Image Understanding by Iterative Optimization. In *KI–96: Advances in Artificial Intelligence. Proc. of the 20th Annual German Conference on Artificial Intelligence,* Dresden, 1996.

[4] Volker Fischer. *Parallelverarbeitung in einem semantischen Netzwerk für die wissensbasierte Musteranalyse.* Infix, Sankt Augustin, 1995.

[5] M.R. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing.* MIT Press, Cambridge, MA, 1968.

[6] H. Niemann, G. Sagerer, S. Schröder, and F. Kummert. ERNEST: A Semantic Network System for Pattern Understanding. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(9):883–905, 1990.

[7] G. Dueck. New Optimization Heuristics: The Great Deluge Algorithm and the Record–to–Record–Travel. *Journal of Computational Physics*, 104(1):86–92, 1993.

[8] D. Goldberg. *Genetic Algorithms: Search, Optimization and Machine Learning.* Addison–Wesley Publ. Co., Reading, Mass., 1989.

[9] Lokendra Shastri. *Semantic Networks: An Evidential Formalization and its Connectionist Realization.* Morgan Kaufmann Publishers, Pitman, London, 1988.

[10] A. Geist and V. Sunderam. The PVM System: Supercomputing Level Concurrent Computations on a Heterogeneous Network of Workstations. In *Proc. of the 6th IEEE Conference on Distributed Memory Computing* Portland, Oreg., pages 258–261, 1991.

[11] M. Mast, F. Kummert, U. Ehrlich, G. Fink, T. Kuhn, H. Niemann, and G. Sagerer. A Speech Understanding and Dialog System with a Homogeneous Linguistic Knowledge Base. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(2):179–194, 1994.

[12] H. Niemann, E. Nöth, E.G. Schukat-Talamazzini, A. Kießling, R. Kompe, T. Kuhn, K. Ott, and S. Rieck. Statistical Modeling of Segmental and Suprasegmental Information. In A.J. Rubio Ayuso and J.M. López Soler, editors, *Speech Recognition and Coding. New Advances and Trends*, volume 147 of *NATO ASI Series F*, pages 192–209. Springer, Berlin, 1995.

[13] Wieland Eckert. *Gesprochener Mensch–Maschine–Dialog.* Shaker, Aachen, 1996.