Contextual *Swarm*-Based Multi-layered Lattices: A New Architecture for Contextual Pattern Recognition

David G. Elliman and Sherin M. Youssef

School of Computer Science & IT, Nottingham University
dge@cs.nott.ac.uk, sherin@aast.edu

Abstract. This paper introduces a novel approach for dynamic structuring of contextual lattices. It is anticipated that the approach can be applied to improve the accuracy of word-segmentation patterns in autonomous text recognition systems. A multi-level hierarchical structure of lattices is used to implement the algorithm, and the approach can be applied in a generic manner to other pattern recognition problems. We apply a top-down structural model in parallel with a constrained probabilistic model and intelligent distributed searching paradigm. This paradigm is based on the integration between probabilistic bi-grams and adaptive intelligent *swarm*-based agent search to identify the most likely sentence structures. The searching paradigm allows the exploitation of positive feedback as a search mechanism and, consequently, makes the model amenable to parallel implementation. The distributed intelligence of the proposed approach enables the dynamic structuring of contextual lattices and has proved to scale well with large lattice sizes. Moreover, we believe that the proposed architecture solves the *ill*-conditioned nature of most pattern recognition problems that lies in the effect of noise in the segmentation phase. To verify the developed Swarm-based Intelligent Search Algorithm (SISA), a simulation study was conducted on a set of variable size scripts. The proposed paradigm proved to be efficient in identifying the most highly segmented patterns and also returned good decisions concerning lower probability segments enabling further re-segmentations and re-combinations to take place. The paper is the first to apply the intelligent swarm-based paradigm for the identification of optimal segmented patterns in contextual recognition models. The algorithm is compared with other algorithms for the same problem, and the computational results demonstrate that the proposed approach is very efficient and robust for large-scale statistical contextual-lattice structures.

Introduction

Sentence segmentation has become an integral part of intelligent text recognition and the optimization of large vocabulary models. However, sentence segmentation may include false segments (or sub-words) as a result of the misidentification of segmentation points. The existence of these false words leads to sub-optimal or inaccurate solutions, and severely reduces the recognition rate. Therefore effective identification and reconstruction of false paths (or sentence words) corresponding to false sub segments has the potential to improve the segmentation accuracy. Extensive search has been carried out on the problem of identifying false paths that arise in some applications, such as time analysis and circuit optimization in large digital IC designs. Although the complete identification of all such false paths in a circuit or network is an NP-complete problem, a number of heuristic or approximate methods have been pro-

S. Marinai and A. Dengel (Eds.): DAS 2004, LNCS 3163, pp. 496–507, 2004.

posed [1,2]. The search process is the most important and challenging part of our optimization strategy. The likelihood of different segmentation patterns is computed using scores on the feature model. The search paradigm has then to efficiently choose patterns with the highest likelihood. The number of possible hypotheses grows exponentially with the number of features and imposes heavy constraints on the computation and storage requirements. Therefore intuitively obvious techniques such as an exhaustive search are not at all practical and strategies that save on computation by modifying the search space are vital to achieve efficient and accurate performance.

In character recognition dictionary or n-gram constraints [3] will restrict the allowable combination of characters in forming words, while grammatical constraints will limit the assembly of words into sentences. Lack of success in some pattern recognition problems can be a result of the problem being ill conditioned [4]. An example of the ill-conditioned nature of pattern recognition lies in the segmentation stage. The noise removal stage may apply smoothing for example, which reduces noise, but may result in adjacent symbols touching one another. Where *white space* is used to segment putative characters this small change will result in an incorrect glyph that will fail to be recognized correctly. In practice no segmentation method is sufficiently reliable for handwritten document recognition, and some symbols will be merged while others are broken. These errors are difficult to recover from in a *pipeline* pattern recognition architecture. If many putative segmentations are passed forward in order to increase the likelihood that the correct one is amongst them, then a combinatorial explosion of possibilities is likely to render solution space intractable.

In this paper, we introduce a new architecture for contextual modeling using multi-layer lattices in parallel with an intelligent distributed searching paradigm for the optimization of the contextual model. A new adaptive intelligent swarm-based agent approach is proposed for the optimization of this lattice structure where good suggestions can be fed back to the segmentation layer to adjust the segmentation points. The constructed multi-layered lattice is dynamically stretched and/or shrunk according to improvements in the segmentation process until a more optimal structure is obtained. The proposed approach can be applied to find the best sub-lattice structure of *L*-best sentences from segmented entities formed from low-level processing. The proposed iterative intelligence, swarm-based agent model is a first applied to this class of problems. From experimental results we believe that our proposed approach holds promise in improving the segmentation process and is robust and dynamically adaptable to any changes in the contextual model. The paper is organized as follows. In the next section, we introduce the contextual model based on a multi-layer lattice and illustrate by describing the representation of the segmented words using attributed *word*-lattice graph. Next, in section 3, we introduce the proposed swarm-based intelligent searching paradigm. The dynamic re-structuring of contextual lattices is explained in section 4. Experiments and simulation results are presented in section 5 with an analysis of the performance of the proposed algorithm. Finally, a summary and conclusions are given in section 6.

Contextual Modeling Using a Multi-layer Lattice

In order to explain the model, a simple example is illustrated by a tiny subset of possible English sentences as follows: *(1) The cat sat on the mat*, *(2) The boy threw the ball*.

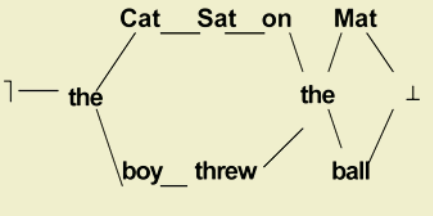

Fig. 1. A Word Lattice

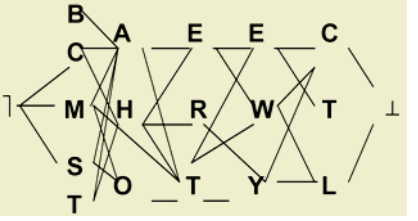

Fig. 2. A Letter Lattice

In this domain we have only ten words, three of which are the word "*the*". The a priori probability of "*the*" is therefore 0.3, while that of the other words is 0.1. A transition matrix can also be derived with the probability of any pair of words appearing contiguously. We introduce special tokens to mean before the start of, and after the end of a sentence, and include these in our bi-gram probabilities. These tokens might be termed *top* ($\top$) and *bottom* ($\bot$) in the context of forming a lattice as shown in figure 1. In general a lattice implements a partial ordering based on the $\leq$ relationship. Our lattice refers to the ordering or words in a sentence and is in fact a total ordering (that is on $<$) as it is not possible for two words to share the same position in a sentence.

A similar lattice may be constructed for letters as shown in figure 2. Consider an imperfect segmentation process that takes an image and divides it into a sequence of glyphs which it is hoped correspond to characters, and which also identifies word boundaries (Δ). Each of the glyphs is processed by a classifier, which returns a set of character classes, each with an associated probability. Every sentence of characters between word boundaries is applied to a character lattice of the type shown in figure 2. This prunes character combinations that could not form words in the model. If no valid word can be formed then the system backtracks to the segmentation stage. Where words can be formed that obey the lattice constraints, these are assigned a probability according to the recognition, letter and letter bi-gram probabilities. That is: $P_w = \prod_{l=1\ldots L}(p_l r_l) \prod_{l=1\ldots L-1}(d_l)$.

Where r_l is the recognition probability of a letter, p_l is the a priori probability of that letter, and d_l is the bi-gram probability of the letter in position l and the following letter. This expression is normalized to remove the effects of word length and scaled to a convenient magnitude. The system backtracks to the segmentation stage, and the process is repeated for the affected words. This process will usually result in groups of words that have a probability of being correct. These constitute our top-down cue, and we hypothesise words based on the most likely words in the model. This process will suggest segmentation and classification mistakes that can be corrected, and the process repeated to try to establish further results that are consistent with those sequences already found.

Representation of the Segmented Words Using Attributed Word-Lattice Graph

Word strings (or sub segments) which are resulted from different segmentation likelihood patterns, are represented as a bi-gram attributed *word*-lattice graph $\boldsymbol{L_G}$ ***(⊤, S, T, A, ⊥)***, where:

$\top$: token mean before the start of sentence.

S: the set of all states in the lattice, where a state represents a word (or sub-segment) string, $|S| = N$.

T: the set of bi-gram transition probability between each two words. $\eta = \{\eta_{ij} \; \forall \; i,j \in N\}$, where η_{ij} is the bi-gram probability of observing a word j given that it is preceded by the word i.

A: the set of states probability pairs , $A = \{a_1,a_2,a_3,\ldots,a_N\}$, and $a_i = \{\varphi_i(u), \vartheta_i(u)\}$, where:

$\varphi_i(u)$ is the recognition probability, and $\vartheta_i(u)$ is the a priori probability of that word.

$\bot$: token mean after the end of sentence.

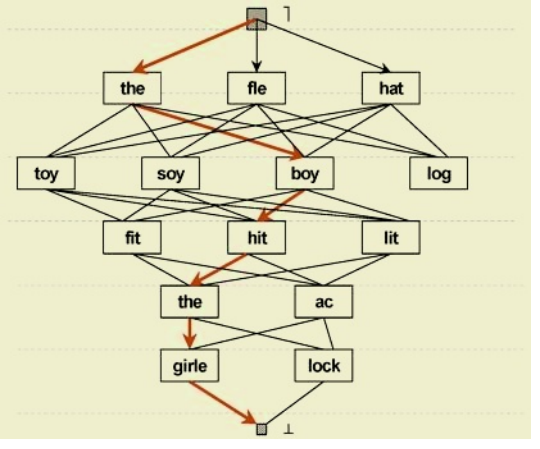

Fig. 3. An example of a fully-connected *Word*-lattice graph of 14 words

Swarm-Based Intelligent Search for the Optimization of Contextual Lattices (SISOCL)

Overview of Swarm Intelligence (Ant Colony Systems ACS)

Swarm intelligence originates from the work on the emergence of collective behaviors of real ants [5-7]. Ant Colony Systems (ACS) is a particular heuristic of ant colony optimization (ACO), one of the nature-inspired meta-heuristics to the solution of discrete optimization problems. The first ACS was introduced by Dorigo [8,9], which is termed the ant system (AS). It is the result of research on computational intelligence approaches to combinatorial optimization problems. By laying down different density trails of an attracting substance called a “pheromone”, ants become able to discriminate between food sources of different routes and qualities. Sensing of pheromone trails is used as a mechanism for the indirect communication among individuals regarding paths, and used to make routing decisions. In ant colony-based algorithms, a set of artificial agents moves on the graph which represents the instance

of the problem. While moving they build solutions and modify the problem representation by adding collected information. This process continues until almost every agent will eventually choose the same optimal path. Studies have shown that agents learn about the state of the environment continuously and very effectively, and can search for specific items of information in a large search space. In addition, the artificial ants are equipped with a local heuristic function to guide their search through the set of feasible solutions.

The Proposed Swarm Agent Model

The objective of our searching paradigm is to find the top *L*-best sentences in the contextual segmentation model and return good decisions for dynamic re-structuring of the contextual *word*-lattice. The main concept in the algorithm is based on generating populations of swarm agents able to navigate in the search space in a distributed manner and intelligently find the best sub-lattice structure representing the top *L*-best sentences. Two types of swarm-based agents are proposed in our model: the Forward agent (F-agent) and the Backward agent (B-agent). The Forward agent (F-agent) is modeled as a small moving object, handling a small stack memory and capable of exploring the search space based on the local knowledge of the neighbouring nodes, and which can apply a local updating process to the environment. At any time step, the F-agent at node *i* has to choose a neighboring node *j* to move to. It samples a random number *q*. If $q \le q_0$, then the best forward word-node is chosen (exploitation) according to equation (1), otherwise a word neighboring node is chosen according to equation (2) (bias exploration):

$$j = \begin{cases} \arg \max_{u \in S_k(i)} \left\{ [\tau(i,u)]^{\alpha} [\eta(i,u).\psi(u)]^{\beta} \right\} & \text{if } q \le q_0 \\ J & \text{otherwise} \end{cases} \quad (1)$$

where:

$\tau(i,u)$ is the pheromone trail of edge *(i,u)*,

$\eta(i,u)$ is the bi-gram probability from word node *i* to node *u*.

$S_k(i)$ is the set of nodes that remain to be visited by agent *k* positioned on node *i* (to make the solution feasible),

$\psi(u)$ is the *word*-node probability calculated as follows: $\psi(u) = \varphi(u) \,.\, \vartheta(u)$, where $\varphi(u)$ is the probability of recognition of candidate word *u*, and $\vartheta(u)$ is the probability of frequent use in the *BNC* dictionary.

β is a parameter which determines the relative importance of pheromone versus bi-gram cost ($\beta>0$),

q is a random number uniformly distributed in [0,1],

q_0 is a parameter ($0 \le q_0 \le 1$) which determines the relative importance of exploitation versus exploration,

J is a neighboring word-node selected according to the probability distribution, called a random-proportional rule, given in the following equation:

$$p_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^{\alpha}[\eta(i,j).\psi(j)]^{\beta}}{\sum_{u \in S_k(i)} [\tau(i,u)]^{\alpha}[\eta(i,u).\psi(u)]^{\beta}} & \text{if} \quad j \in S_k(i) \end{cases} \tag{2}$$

When building the agents tours, the chosen edges are guided by both heuristic information and pheromone information. The state transition rule resulting from Equations (1) and (2) favour the choice of nodes (word segments) connected by highly transition probability (highly correlated) words with a greater amount of pheromone. While constructing a tour, the swarm F-agent apply a local updating. It changes the pheromone level on its visited nodes (or edges) by applying the local updating rule as follows:

$$\tau(i,j)=(1-\rho)\tau(i,j)+\rho\tau_0 \tag{3}$$

where τ_0 is the initial pheromone level and $0 < \rho < 1$ is the pheromone evaporation parameter. The effect of the local updating rule is to make the desirability of edges change dynamically in order to shuffle the tour. If ants explore different paths, then there is a higher probability that one of them will find an improving solution than if they all search in the narrow neighbourhood of the previous best tour. Every time an ant constructs a path, the local updating rule will make its visited edges' pheromone diminish and become less attractive. Hence, the nodes in one ant's tour will be chosen with a lower probability in building other ant's tours. As a consequence, ants will favor the exploration of edges not yet visited and prevent the convergence to a common path. A Backward swarm agent (B-agent) is modeled in the same way; a moving object, handling a stack memory and which is able to apply a global updating. The global updating rule is performed after all swarm-agents in the population have completed their tours. In order to make the search more directed, global updating is intended to provide a greater amount of pheromone to highly likelihood sentences and reinforce them. Therefore, only the globally *L*-best swarm-agents that found the *L*-best solutions (the most highly probable sentences) up to the current iteration of the algorithm are permitted to deposit pheromone.

The B-agent (corresponding to each of the *L*-best tours) traverses the same rout of its corresponding F-agent in the opposite direction and the pheromone level is modified according to the global updating rule which is proportional to the fitness of the corresponding tour. The fitness Γ of a candidate solution Ψ is estimated as follows:

$$\Gamma(\Psi)=\gamma. \prod_{\forall i,u \in \Psi, i \to u} \eta(i,u) \cdot \prod_{\forall u \in \Psi, i \to u} \varphi(u) \cdot \vartheta(u) \tag{4}$$

where $\eta(i,u)$ is the bi-gram probability (transition) from node i to node *u*, $\varphi(u)$ is the probability of recognition of word string *u*, and $\vartheta(u)$ is the a priori probability of that word. γ is a scaling coefficient. The solution length l_ψ is represented as the number of segmented words in the solution ψ. This length l_ψ depends on the location of segmentation points in the proposed sentence. The main objective of the globally updating rule is to increase pheromone on word-nodes (edges) of the current *L*-best tours and decrease pheromone on other edges. The pheromone level is modified according to the following equation:

$$\Delta\tau = \begin{cases} \Gamma(\psi) & \text{if the tour belongs to the global } L-\text{best sentenc} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\tau(i,j)=(1-\sigma).\tau(i,j)+\sigma.\Delta\tau \tag{6}$$

where $\Gamma(\psi)$ is the fitness of the corresponding sentence belonging to the *L*-best global best sentences ψ_{gb} (tours) found up to the current iteration. σ ($0 < \sigma < 1$) is the pheromone evaporation rate.

The Dynamic Re-structuring of Contextual Lattices

The proposed paradigm returns good decisions concerning lower probability segments enabling further re-segmentations and re-combinations to take place. The re-segmentation request of the intelligent *SISOCL Layer* can take three forms:

- **Split(*p).** This feedback request ask for splitting a word-string *p* into two new word segments. An example is shown in figure 4.
- **Merge(*p1, *p2).** This request ask for the generation of a new word corresponds to merging two word strings *p1* and *p2 (figure 5).*
- **Adjust(*p1, *p2).** This request ask for the generation of two new word-nodes resulting from adjusting the segmentation point between two words strings *p1* and *p2*. An example is shown in figure 6.

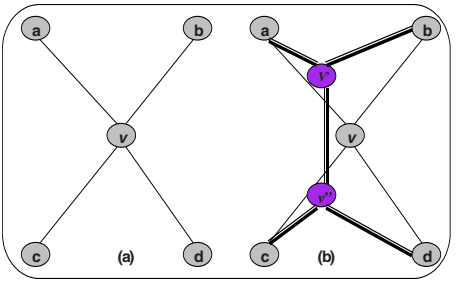

Fig. 4. (a) intermediate node ***v*** the ***Splitting*** of node ***v*** into ***v'***and ***v''***

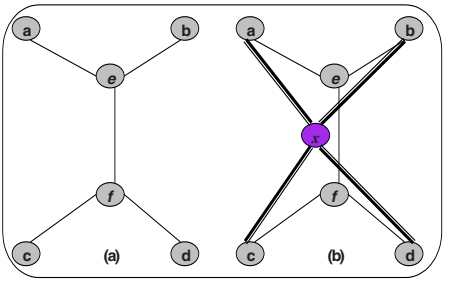

Fig. 5. (a) intermediate nodes ***e*** and ***f***, (b) node ***x*** resulting from ***Merging e*** & ***f***

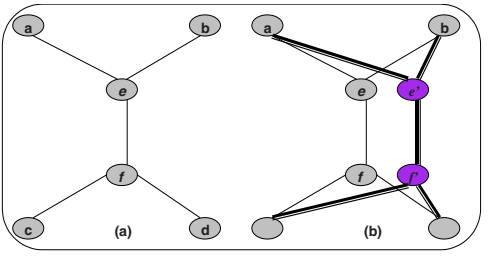

Fig. 6. (a) intermediate nodes ***e*** and ***f***, (b) nodes ***e'*** and ***f'*** are resulting from ***Adjusting*** the segmentation points between ***e*** & ***f***

Experiments and Results

In this section we present the results of our main experiments. We show that the proposed iterative method of jointly optimizing a lexicon, segmentation, and language model not only results in better sentence segmentation but also improves the reduction in the search space of the hypothesis model. The Word-Lattice models are trained using a dataset of 6,318 words with more than 800 occurrences in the whole 100M-word British National Code (BNC). Sets of electronic texts (e-texts), which are produced by Gutenberg Project (one of the Internet's oldest producer of free electronic e-texts) are used in the training process. The first part of our work, we have designed and produced a toolkit, which is a set of C++ software programs designed and implemented to facilitate textual document modeling work. Some of the designed tools are used to process general textual data into: word frequency lists and vocabularies, word bi-gram models, and bi-gram-related statistics. For the purpose of experimentation with the proposed searching paradigm we designed our own vocabulary language model and care has been taken to construct these test models so as to verify the performance of the search engine accurately. For the purpose of scoring, we used the a priori probabilities and the recognition probabilities of each segmented word (word segment) and the bi-gram transition probabilities that represent correlation between word segments [10]. Synthetic data is created by adding different hypothesis to each state in the model. To specify the improving in the quality of the solution, we used the probability factor g*, defined as: $g^* = q\,(\,q - q'\,)\,,$ where q is the probability of the best solution up-to time of consideration, and q' is the probability of the 2nd best solution. In figures 7 and 8, we compare the performance of our proposed algorithm with two randomized local search algorithms: Simulated Annealing (SA) and a hybrid two-Phase Optimization (2PO) [11].

Figure 7 tests the performance over the time. Both our proposed algorithm and the 2PO algorithm converge faster than SA. In figure 8, we studied how well the algorithms scale with the problem size. In order to simulate situations where there is a limited time for optimization, each algorithm was terminated after a constant execution time of 120 seconds even if it did not converge. For larger problems, SA is ineffective since it does not have enough time to freeze. Although 2PO does not converge either, it still manages to find good solutions even for big problem sizes. In figure 9, we compare the running time of the proposed *SISOCL* algorithm with the stack decoder A* algorithm [12], Viterbi algorithm [10,13], SA, and 2PO algorithms, for different lattice sizes.

As observed from the figure, the swarm-based algorithm (*SISOCL)* is very robust for large graph sizes. Its running time grows with a smaller rate than the other compared algorithms. On the other hand, the running time of the 2PO algorithm grows with a smaller rate than SA. The ratio between the running time of SA and 2PO increases with the increase in the number of relations in the large lattice joins the word states. This can be explained by the fact that the 2PO starts with a local minimum. The systematic A* algorithm has a high run-time cost which limits its applicability for large problems. This can be explained by the fact that the A* algorithm works in a best-first search of the lattice looking for the highest probability path, and hence the highest probability sentence. So, as the number of states and relations increase, the running time grows exponentially, rendering exhaustive optimization inapplicable. Figure 10 shows the increase in the probability of the best and 2nd best solution corre-

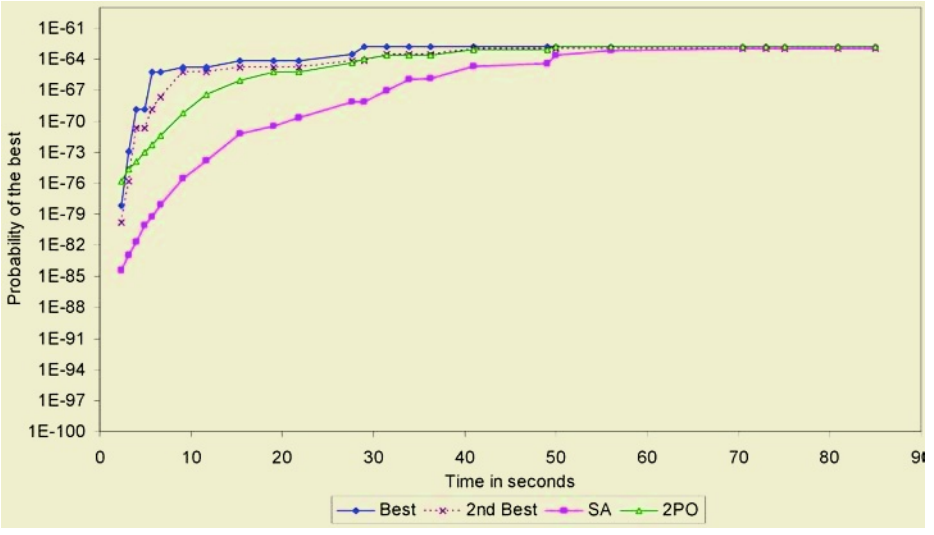

Fig. 7. The change in the likelihood of the best solution versus simulation time for different algorithms in comparison (swarm-based, SA and 2PO), for a *word*-lattice graph of size 35.

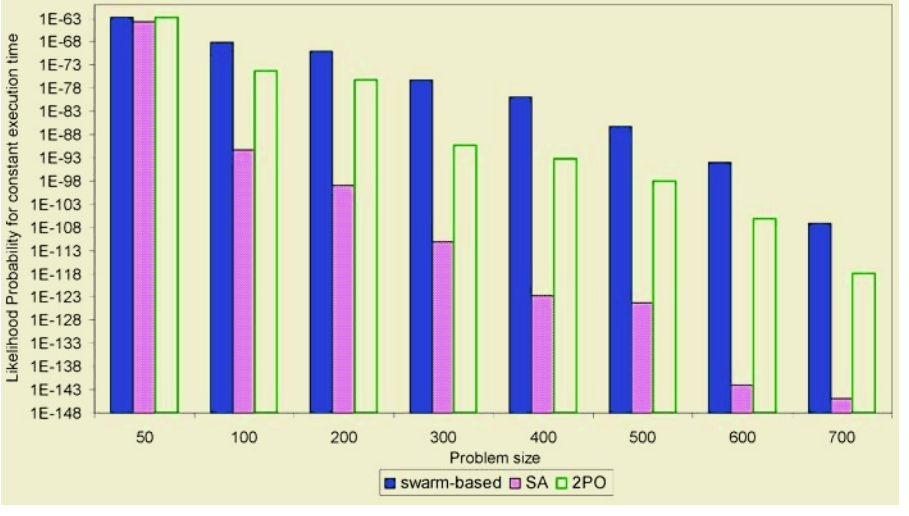

Fig. 8. Likelihood probability of the best solution versus problem size

corresponding to dynamic changes in the word-lattice model relative to re-segmentation processes. As shown in the chart, we obtained fast improvement. This means that the proposed approach is intelligent enough such that coming back with good decisions for the re-segmentation points, and hence achieve fast improvement in identifying the best segmentation pattern with maximum likelihood.

Figure11 illustrates the changes in the relative probability versus the number of re-segmentations for different word-lattice sizes. As observed from the figure, the pro-

posed searching paradigm first converges to the most likelihood solution. Then, decisions for re-segmentations take place. This is represented as a rapid increase in the relative probability, rapidly reaching values close to 1. So, the relative error decreases dramatically according to the dynamic re-configuration of the word-lattice structure. Good re-segmentation decisions are taken such that improvements of solutions takes place very fast even for large graph sizes.

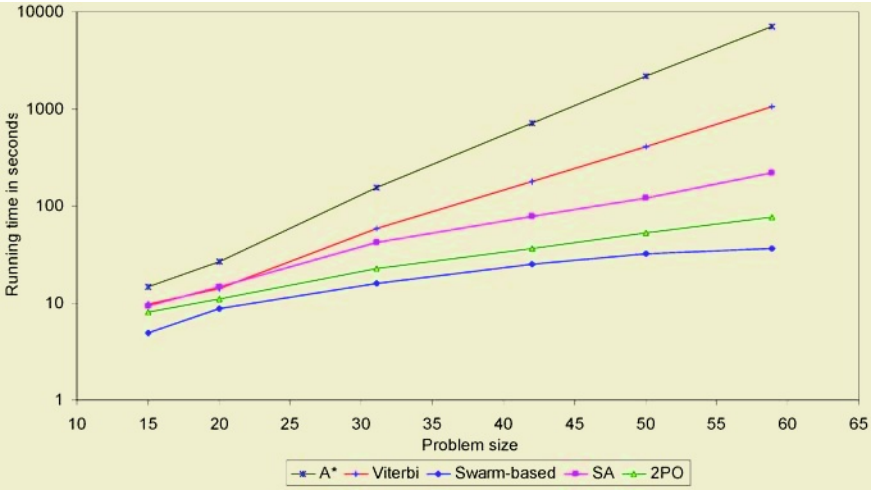

Fig. 9. Convergence time for different *Word*-lattice sizes

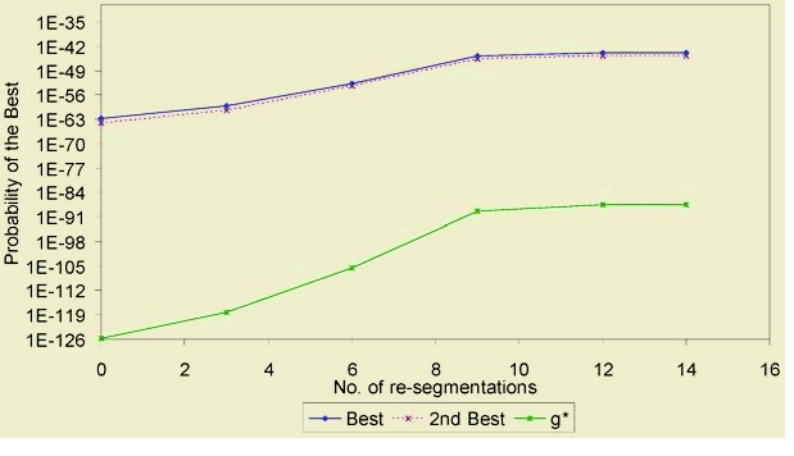

Fig. 10. Improving the probability of the Best segmentation pattern versus number of re-segmentation

Summary and Conclusions

A pattern recognition architecture has been proposed using hierarchical constraint lattices, backtracking, and a swarm-based search strategy. This has been implemented

and found to be highly effective and to scale well with large attributed lattice structures. If the lattice consists of a limited number of states, application of systematic algorithms, like A*, is efficient. As the number of states increases, however, the running time of these systematic methods grow exponentially, rendering exhaustive optimization inapplicable. The main strengths of the proposed algorithm are its robustness and the intelligent nature of the agents in exploring new search areas. Another potential advantage of such a paradigm is the ability to explore new good solutions in large size graphs and dynamically adapt to the changes in the hypothesized model. The simulation study for different word-lattice graphs demonstrates that the proposed algorithm is highly robust and very efficient in the sense of yielding fast and high-quality solutions. The results show that the method can easily remove a large set of specific false sub-paths with excellent run time performance and that highly probable sentences are computed early in the search.

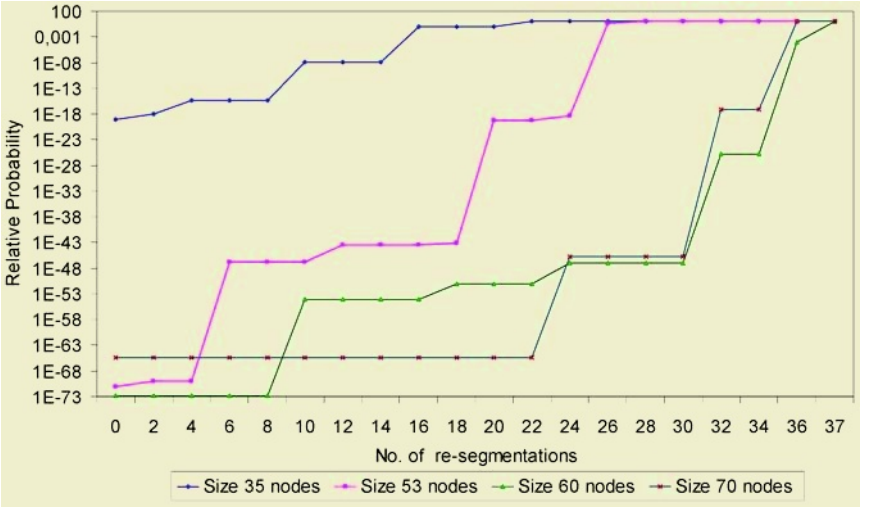

Fig. 11. Relative probability versus no. of re-segmentations for different *word*-lattice sizes

References

1. H., D., Du, C. et. al.: On The General False Path Problem in Timing Analysis. DAC (1989) 555-560
2. McGeer, P.C. et.al.: Efficient Algorithms for Computing the Longest Viable Path in a Combinatorial Networks. DAC (1989) 561-567
3. Elliman, D. G., Lancaster, I.: A review of Segmentation and Contextual Analysis Techniques for Text Recognition. Pattern Recognition, 23 (1990) 337-346
4. Tikhonov, A. N., arsenin V.: Solutions of *Ill*-posed Problems. W.H. Winston, Washington (1977)
5. Di Caro, Gianni, Dorigo, Marco: AntNet: A Mobile Agents Approach to Adaptive Routing. IRIDIA, University Libre de Bruxelles, av. F. Roosevelt, CP 194/6, 1050 – Brussels, Belgium, Technical Report IRIDIA 97-12 (1997)
6. Shi, Y., Eberhart, R.C.: Empirical Study of Particle Swarm Optimization. In: Proceedings of Congress on Evolutionary Computation (1999) 1945-1950

7. White, T.: Routing With Swarm Intelligence. System and Computer Engineering (SCE) department, Technical Report SCE-97-15, Carleton University (1997)
8. Dorigo, M.: Optimization, Learning and Natural algorithm. PhD thesis, DEL, Politecnico di Milano, Italy (1992).
9. Di Caro, M. G., Gambardella, M.: Ant Algorithms for discrete optimization. Artificial Life. 5 (1999) 137-172
10. Amtrup, J.W.: Incremental Speech Translation. Lecture Notes in Artificial Intelligence 1735, Springer-Verlag, Berlin Heidelberg New York (1999).
11. Kalnis, P. et al.: View Selection Using Randomized Search. Data & Knowledge Engineering (2002) 89-111.
12. Jurafsky, D., Martin, J. H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice-Hall, Inc., New Jersey (2000).
13. Forney, G. David: A Viterbi Algorithm. In: Proceedings Of The IEEE. 61:3 (1973), 268-278