A Discrete Particle Swarm Optimization Algorithm for Domain Independent Linear Text Segmentation

Ji-Wei Wu
Dept. of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
jwwu.cs97g@nctu.edu.tw

Judy C.R. Tseng
Dept. of Computer Science and Information Engineering
Chung Hua University
Hsinchu, Taiwan
judycrt@chu.edu.tw

Wen-Nung Tsai
Dept. of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
tsaiwn@cs.nctu.edu.tw

***Abstract*—Linear text segmentation has been used in several natural language processing tasks, such as information retrieval and text summarization. It has been proven that linear text segmentation is beneficial to these tasks. To improve the performance of linear text segmentation, a novel domain independent linear text segmentation algorithm, called *DPSO-SEG*, is proposed in this paper. *DPSO-SEG* applies the Discrete Particle Swarm Optimization (*DPSO*) algorithm to find the optimal topical segments. The performance of *DPSO-SEG* is compared with some state-of-the-art linear text segmentation algorithms. The experimental results show that *DPSO-SEG* is advantageous in its controllable time complexity and its promising accuracy, especially when the segment size is consistent.**

Keywords-Discrete Particle Swarm Optimization; Linear text segmentation

I. INTRODUCTION

Documents are usually composed by several subtopics, or different aspects of the same topic. The aim of linear text segmentation is to segment texts into thematic homogeneous units (i.e. to discover the topic boundaries), each of which corresponds to a relevant subject and only consists of contiguous paragraphs. Linear text segmentation is important to several natural language processing tasks, such as information retrieval and text summarization. The performance of the natural language processing tasks will be improved if the subtopic structure of text is identified [19]. In information retrieval, linear text segmentation enables the system to provide only the fragment(s) of text that the user is interested in, rather than the whole document. It will provide more accurate information to the user and reduce the user's burden to read the whole document to obtain the fragment(s) of his/her interest. In text summarization, linear text segmentation enables the system to summarize subtopics of a document instead of the whole document [9].

Many linear text segmentation algorithms have been proposed, such as *TextTiling* [8], *C99* [2] and dynamic programming algorithms [5, 20]. Although it has been proved that these algorithms have good performances on linear text segmentation, some problems still exist: some of the algorithms use only local clues; the thematic proximity of sentences belonging to the concerned area is computed solely [14]. It makes these algorithms suffer from lower accuracy. Although some of the algorithms use global clues, such as the similarities among all parts of a text, the segmentation is still performed by a local method (such as clustering) [5].

To cope with these problems, several global optimization approaches that employ global information representations have been proposed. Some of them attempt to optimize the internal segment cohesion by using Dynamic Programming (*DP*) [5, 9]. However, the consideration of segmentation cost function is still a problem, since some useful information does not appear treatable, such as the dissimilarity between adjacent segments [13]. Although some *DP* algorithms consider both internal cohesion and dissimilarity between adjacent segments, it suffers from the high computing complexity [20].

In recent years, some linear text segmentation algorithms based on Genetic Algorithm (*GA*) have been proposed. Thanks to the evolutionary nature of *GA*, both internal cohesion and dissimilarity between adjacent segments appear to be treatable. Moreover, the semantic boundaries are considered globally [13], rather than locally. Nevertheless, GA-based algorithms suffer from several well-known disadvantages, such as complex implementation and expensive computational cost. Moreover, many parameters are used in *GA*, such as the type of selection, the type of crossover, mutation probability and replacement strategy. It is difficult to determine the appropriate settings of these parameters.

In this paper, a novel domain independent linear text segmentation algorithm, called *DPSO-SEG*, is proposed.

Both the global information and global optimization approach are used to cope with the problems of previous approaches. *DPSO-SEG* is based on the most general statistical approach and uses the global information representation. Moreover, Discrete Particle Swarm Optimization (*DPSO*) [11] is used as the global optimization algorithm to find the optimal topical boundaries. *DPSO* is a stochastic and population based evolutionary algorithm that can be used to find an optimal solution for problem solving. Some *DPSO* algorithms have been developed and successfully applied to related problems. The results illustrated that the *DPSO* algorithm can achieve better performance.

To the best of our knowledge, *DPSO* has not yet been applied to linear text segmentation. It is expected that *DPSO-SEG* will do as well in linear text segmentation as in other research areas. We will prove this by some experiments in Section V. Furthermore, compared to other evolutionary algorithms, such as Genetic Algorithm (*GA*), *PSO* is easy to implement and with a lower complexity. Another reason why *PSO* is attractive is that *PSO* has fewer parameters to be tuned [4, 15]. Moreover, the time complexity of *PSO* can be controlled by adjusting the number of terminal iterations. That is, the controllable time complexity of *PSO* makes *DPSO-SEG* more scalable.

II. Previous Works

There are several classical approaches that have been proposed for linear text segmentation. Existing techniques can be divided into two categories: lexical cohesion methods and multi-source methods [2]. The lexical cohesion methods utilize word stem repetition, context vectors, entity repetition, semantic similarity, word distance model and word frequency model to detect cohesion. These methods include sliding window, lexical chains, dynamic programming, agglomerative clustering and divisive clustering [2, 16]. The multi-source methods combine lexical cohesion with other indicators of topic shift, such as cue phrase, prosodic features, reference, syntax and lexical attraction using decision tree and probabilistic methods [2].

In the literatures [5, 20], some problems of existing algorithms have been pointed out. Some algorithms, such as sliding window [7], use only local information to segment text. The thematic proximity of sentences belonging to the concerned area is computed solely, and the window size is difficult to determine [14]. Although some algorithms use global approaches, involves the similarity between all parts of a text [2, 17], however, they are not completely globalized methods: they still perform the actual segmentation by the local method – clustering [5]. For example, similarity matrix of the text sentences is used in *C99* [2], while the locations of the topic boundaries are determined by clustering. To cope with this problem, dynamic programming has been applied to text segmentation [5, 9, 20]. In [9], the text segmentation problem is converted into an image segmentation problem. An image enhancement technique, called anisotropic diffusion, was applied and the dynamic programming technique was adapted to find the optimal topical boundaries. Fragkou et al. [5] also used dynamic programming in optimizing the segmentation cost function, but a prior knowledge about the segment length is needed.

Above mentioned approaches attempt to optimize the internal segment cohesion more globally by using dynamic programming algorithms. However, some useful clues, such as the similarity between adjacent segments, are not considered. Ye et al. [20] proposed a novel dynamic programming approach which considers both within-segment lexical similarity and between-segment lexical similarity. Nevertheless, it suffers from its high computing complexity O(K^3), where *K* is the number of sentence in the text. In [13], a Genetic Algorithm (*GA*), called *SegGen*, is proposed. Both the internal cohesion of the formed segments and the similarity of the adjacent segments are considered. Furthermore, the creation of boundaries is performed simultaneously, instead of sequentially. However, some well-known disadvantages *of GA* have been pointed out in the literatures: First, the implementation of *GA* is complex. Second, large number of evaluations of the objective function is required, which is computationally expensive. Finally, there are many parameters to be adjusted. Appropriate parameters are difficult to determine, and poor parameter estimation can lead to abortive results.

In this paper, a novel linear text segmentation algorithm, *DPSO-SEG*, is proposed. The algorithm, *DPSO-SEG*, makes full use of all the merits of previous works, such as the global similarity measurement and the global optimization approach.

III. Particle Swarm Optimization

Particle Swarm Optimization (*PSO*) was originally proposed by Eberhart and Kennedy, inspired by the social behavior of bird flocking and fish schooling [10]. *PSO* is a swarm intelligence technique, each particle is an individual, and the swarm is composed of particles. In *PSO*, each particle is modeled by a position and a velocity. A particle's position in the search space (solution space) represents one solution for the problem. Particles share useful information and cooperate to find the best position (solution) in the search space. At every iteration, each particle adjusts its velocity and position according to its own best experience and the swarm's best experience:

$$\begin{aligned} v_{id}(t+1) = {} & w * v_{id}(t) \\ & + c_1 * rand_1 * (p_{id}(t) - x_{id}(t)) \\ & + c_2 * rand_2 * (p_{gd}(t) - x_{id}(t)) \end{aligned} \tag{1}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{2}$$

where $v_{id}(t)$ is the velocity of particle *i* at time *t* in the search space along dimension d; $v_{id}(t+1)$ is the new velocity of particle *i*; x_{id} is the position of particle *i*; p_{id} is the position of particle *i* that experience the best performance; p_{gd} is the position of the swarm that experience the best performance; *w* is the inertia weight factor; c_1 and c_2 are constant weight factors; $rand_1$ and $rand_2$ are random variables in the [0, 1] interval.

PSO was first applied to optimize continuous nonlinear problems. Later, *PSO* was applied to operate on discrete binary variables, and a Discrete *PSO* (*DPSO*) was developed [11]. In *DPSO*, each dimension of the particle's position is represented as 0 or 1 rather than a real number. Moreover, the particle's velocity v_{id} represents the probability of bit x_{id} taking the value 0 or 1. Therefore, the (2) that adjusts the particle position is replaced by the (3) as follows:

$$\begin{aligned}&if\,(\,rand() < S(v_{id}(t+1)))\; then\; x_{id}(t+1) = 1\\ &else\; x_{id}(t+1) = 0\end{aligned} \tag{3}$$

where *S(v)* is a sigmoid limiting transformation; *rand()* is a quasi-random number selected from a uniform distribution in [0, 1].

IV. The Proposed Discrete Particle Swarm Optimization Algorithm

At present, *PSO* is widely used to solve continuous problems, while the studies of applying *PSO* on discrete optimization problems are relatively few. Generally speaking, binary representation is a more efficient data representation and is more easily to be simulated and implemented in nature. While the linear text segmentation problem can be considered as a combinational problem, it is solved by the *DPSO* algorithm, called *DPSO-SEG*, in this paper.

The goal of *DPSO-SEG* is to identify the optimal topic boundaries of the text segments in a document. At first, the terms within each sentence are tokenized and stemmed. Then, generic stop words are removed. After the basic preprocessing, each sentence is represented as a term-frequency vector. Then, sentence-sentence similarity between a pair of sentences is computed by cosine similarity. A sentence similarity matrix of the text then constructed using the sentence-sentence similarity. Finally, the optimal boundaries are created by *DPSO* according to the sentence similarity matrix.

A. *Sentence Similarity Matrix*

Suppose that a text *T* contains *S* sentences and *W* distinct words in each sentence. After tokenization, stop words removing and stemming, a dictionary of words and the frequency of each word in each sentence is constructed as a vector S_i for each sentence. The similarity between a pair of sentence S_i and S_j is computed by cosine measure:

$$Sim(S_i, S_j) = \frac{\sum_k w_{ik} \times w_{jk}}{\sqrt{\sum_k w_{ik}^2}\sqrt{\sum_k w_{jk}^2}} \tag{4}$$

where w_{ik} denote the frequency of word *k* in sentence *i*; w_{jk} denote the frequency of word *k* in sentence *j*.

Next, the sentence similarities are illustrated by considering a theoretical matrix – sentence-sentence similarity matrix *D*. The sentence similarity matrix *D* is constructed by computing all pair-wise sentence similarities. The sentence similarity matrix is also quantized for better boundaries display with:

$$d_{ij} = \begin{cases} 0 & if\ d_{ij} > 0 \\ 1 & if\ d_{ij} = 0\ or\ i = j \end{cases} \tag{5}$$

Since every sentence has the highest similarity with itself (i.e. $Sim(S_i,S_i)$=1), it will lead the favor segments containing only one sentence. To cope with this problem, we can eliminate the self similarity relationships (i.e. d_{ii}=1).

B. *Encoding*

In the proposed *DPSO-SEG* algorithm, the multidimensional vector space is modeled as the problem space of *DPSO*. A particle in the swarm can be represented as a possible solution. Given a text with *n* sentences, the particle position is represented as a bit string of (*n* - 1) elements. An example of a particle is shown in Fig. 1, where each element b_i represents a boundary. If there is a boundary between sentence *i* and *i* + 1, then b_i = 1, otherwise b_i = 0. The particle velocity is generated in the same way with randomly chosen real value in [0, 1].

C. *Initial Particles Generation*

Generally speaking, the lengths of semantic segments of a document tend to be even. That is, it is usually the case that there is little variance on the numbers of sentences contains in the semantic segments of a document. Accordingly, the initial solutions of *DPSO-SEG* are created by assuming that a document contains at least 2 semantic segments with even number of sentences. Suppose a document *D* contains *L* sentences and the number of initial particles to be created is *n*. The *i*th initial particle represents the solution with length L_i of segments, where L_i is determined by $L_i = i \times L/(2 \times n)$. For example, suppose a text *T* contains 100 sentences and the predefined number of initial particles is 20. The first particle contains 3 (1×100/(2×20)) segments, the second particle contains 5 (2×100/(2×20)) segments, and so on.

In order to start with heterogeneous populations, the initial position of each boundary in a particle is decided by a randomly chosen integer, which is close to the position of evenly divided boundary (about $\pm$1 sentences). For example, in the previous example, the positions of the 2 boundaries in first particle, the position of the first boundary will be generated by a random number in the range from 32 to 34, and the second boundary will be generated by a random number in the range from 65 to 67.

Besides of the above way of initial particle generation, we also investigate an alternative way of initial particle generation. According to the experimental results in previous

b_1	b_2	…	b_i	…	b_{n-1}

Figure 1. The example of a particle.

works [3], the performance of *PSO* could be improved by seeding the initial swarm with the result of an efficient algorithm. Moreover, the higher fitness value of the initial particle will directly force the particle to start the optimal refining stage. For these reasons, we devise another algorithm, called *DPSO-SEG*$_{C99}$, which create its initial solutions by the well known algorithm *C99* [2] rather than by controlled random number generation.

D. Fitness Function

To evaluate each of the particles, four criteria are used: the first two criteria are the commonly used internal (lexical) cohesion and external (two consecutive segments) dissimilarity. The third criterion is the standard deviation of segment length in a document. The last criterion is the number of segments.

The fitness function can be represented as:

$$\begin{aligned} f(particle_i) = & \frac{1}{\#seg} \times \sum_{j=1}^{\#seg} Internal(j) \\ & - \frac{1}{\#seg-1} \times \sum_{k=1}^{\#seg-1} External(k,k+1) \\ & + STDEV(i) \times \alpha \\ & + \frac{\#seg}{\#sentence} \times \beta \end{aligned} \tag{6}$$

where α and β are weighting factors. In our experience, the weighting factors (α, β) are better set as (1.5, 0.5) when the variance of segment length is greater, and (3, 0) when the variance of segment length is fewer. In order to find a proper segmentation, the lower fitness value is preferred.

1) Internal (lexical) cohesion

In a good segmentation, sentences within a segment should cover the same subtopic. Hence, the goal is to find the segmentation with the minimum internal (lexical) cohesion. The internal cohesion can be represented as:

$$Internal(j) = \frac{\sum_{S_m \in seg_j} \sum_{S_n \in seg_j} Sim(S_m, S_n)}{|seg_j| \times |seg_j|} \tag{7}$$

Internal(j) represents the internal density of the segment *j*; seg_j denote segment *j* and $|seg_j|$ denote the number of sentences in seg_j; $\sum_{S_m \in seg_j} \sum_{S_n \in seg_j} Sim(S_m, S_n)$ is the total number of 1's in the sentence similarity matrix *D* corresponding to the seg_j and $|seg_j| \times |seg_j|$ is the area of the sub-matrix.

2) External (two consecutive segments) dissimilarity

In a good segmentation, sentences among different segments should belong to different subtopics. Hence, the goal is to find the segmentation with the maximum external (two consecutive segments) dissimilarity. The external dissimilarity is represented as:

$$External(k, k+1) = \frac{\sum_{S_m \in seg_k} \sum_{S_n \in seg_{k+1}} Sim(S_m, S_n)}{|seg_k| \times |seg_{k+1}|} \tag{8}$$

External(k, k+1) is the dissimilarity between segment *k* and segment *k* + *1*; $\sum_{S_m \in seg_k} \sum_{S_n \in seg_{k+1}} Sim(S_m, S_n)$ is the total number of 1's in the matrix D corresponding to the adjacent segments seg_k and seg_{k+1}; $|seg_k| \times |seg_{k+1}|$ is the area of the sub-matrix.

3) Standard deviation of the segment length

Generally speaking, it is usually the case that the number of sentences in each segment is similar. For this reason, the goal is to minimize the standard deviation of the segment length in a text. The standard deviation of the segment length is represented as:

$$STDEV(i) = \sqrt{\frac{\sum_{j=1}^{\#seg} (|seg_j| - \mu)^2}{\#seg}} \tag{9}$$

STDEV(i) is the standard deviation of the segment length in a text; μ is the mean length of the segments in particle *i*; *#seg* is the number of segments.

4) Number of segments

According to our observation, owing to the factor *STDEV(i)* is considered, the number of boundaries created by *DPSO-SEG* is more than the actual number when the variance of segment length is greater. To cope with this problem, the number of segments is considered in the fitness function. The number of segments is normalized as:

$$\frac{\#seg}{\#sentence} \tag{10}$$

To make the segmentation better, low number of segments is preferred.

E. DPSO-SEG Algorithm

After constructing the sentence similarity matrix, *DPSO-SEG* algorithm applies the principle of *DPSO* as follows:

1) Initialize a heterogeneous population of particles. Initial particles are created by the process mentioned in Section IV.C; each particle randomly chooses (n – 1) real values as its multi-dimensional velocity.

2) Evaluate the fitness value by (6).

3) Update the best position found by each particle itself so far.

4) Update the best position found by whole swarm so far.

5) Update the velocity and position of each particle according to (1) and (3).

6) Repeating step 2 until the predefined number of iterations is exceeded.

V. Experiments

The most commonly used and publicly available test corpus created by Choi [2] is used to evaluate the performance of *DPSO-SEG*. This corpus consists of 700 samples. A sample is a concatenation of ten text segments and each segment is the first *n* sentences of a randomly selected document from the Brown corpus [6]. Table I is the statistics of the test corpus.

The evaluation metric we used is the error probability metric P_k proposed by Beeferman et al. [1]. The probability of a randomly chosen pair of words a distance of k words apart inconsistently classified, P_k(ref, hyp), is calculated. This probability can be decomposed into the *miss* and *false alarm probabilities*:

$$\begin{aligned} &p(error \mid ref, hyp, k) \\ &\quad = p(miss \mid ref, hyp, diff, k)p(diff \mid ref, k) \\ &\quad + p(false\ alarm \mid ref, hyp, same, k)p(same \mid ref, k) \end{aligned} \tag{11}$$

lower error probability means higher accuracy.

Two experiments are conducted: the first investigates the convergence rates of *DPSO-SEG* and *DPSO-SEG*$_{C99}$ under various test sets; the second compares *DPSO-SEG* and *DPSO-SEG*$_{C99}$ with several well known methods without complex preprocessing (e.g. document dependent stop words removing and anisotropic diffusion [9]). The parameters in the *DPSO* algorithm employed are set as Table II. The number of particles is set to 20 as recommended by [11, 18]; the initial weight *w* is set to 1; the acceleration coefficient constants *c1* and *c2* are also set to 1.

A. *Experimental results of convergence*

Since *DPSO* is a nondeterministic algorithm, in order to observe the variance of error probability, *DPSO-SEG* is evaluated in various numbers of iterations from 50 to 2000 (the variation is 50 each time). The average error probability of 10 runs is computed for each number of iterations.

Fig. 2~5 show the convergence behaviors of error probabilities of *DPSO-SEG* and *PSO-SEG*$_{C99}$ under the four test sets. In Fig. 2 that represents the test set 3-11, the value of P_k is reduced sharply with fewer numbers of iterations, and smoothly after 350 iterations. It is converged at about 1500 iterations. Similar results are obtained in the following three sets.

TABLE I. STATISTICS OF THE TEST CORPUS.

Range of n	3-11	3-5	6-8	9-11
#samples	400	100	100	100

TABLE II. PARAMETER SETTINGS.

Parameter	**Value**
Number of particles	20
w	1
c1	1
c2	1

Fig. 2-5 also show that the performance of *DPSO-SEG*$_{C99}$ is better than *DPSO-SEG*.

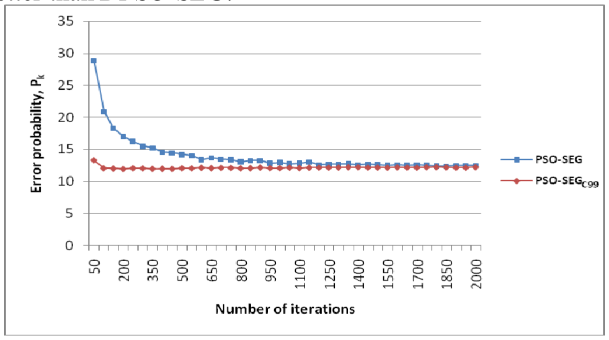

Figure 2. The convergence behaviors of PSO-SEG and PSO-SEG$_{C99}$ in Set 1 (range of 3-11).

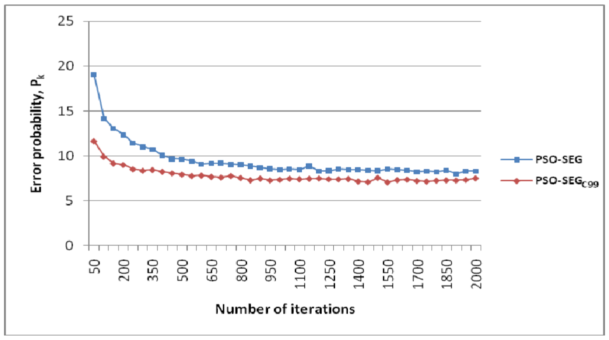

Figure 3. The convergence behaviors of PSO-SEG and PSO-SEG$_{C99}$ in Set 2 (range of 3-5).

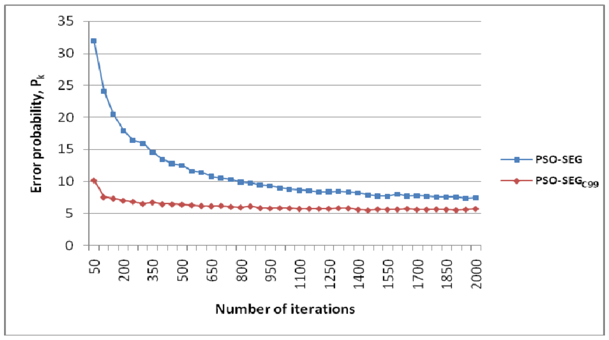

Figure 4. The convergence behaviors of PSO-SEG and PSO-SEG$_{C99}$ in Set 3 (range of 6-8).

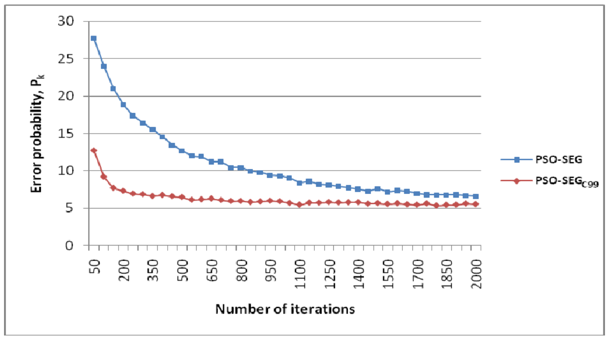

Figure 5. The convergence behaviors of PSO-SEG and PSO-SEG$_{C99}$ in Set 4 (range of 9-11).

Moreover, *DPSO-SEG$_{C99}$* also converges faster. In the first testing set, it is converge at about 750 iterations. Similar results are obtained in the following three sets.

B. Experimental results of comparisons

Table III compares *DPSO-SEG* and *DPSO-SEG$_{C99}$* with various linear text segmentation algorithms. The performance data of these state-of-the-art algorithms, *TextTiling*, *C99*, *U00*, and *TopSeg*, are based on the literatures [2, 12]. The performances of *DPSO-SEG* and *DPSO-SEG$_{C99}$* are shown as the mean values of P_k in their convergent states (i.e. 1550 to 2000 iterations for *DPSO-SEG* and 800 to 2000 iterations for *DPSO-SEG$_{C99}$* respectively).

According to Table III, *DPSO-SEG* is demonstrated to provide a comparatively promising accuracy. Moreover, Tables III shows that *DPSO-SEG$_{C99}$* is more accurate than the well known algorithms listed when the segment size is more consistent (that is, on Set 2, Set 3 and Set 4). When the segment size is inconsistent (such as Set 1), the error probabilities of *DPSO-SEG* and *DPSO-SEG$_{C99}$* are not so promising; yet they are still acceptable.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we presented a novel linear text segmentation algorithm, *PSO-SEG*, which makes full use of all the merits of global similarity measure and global optimization method. The experimental results show that *DPSO-SEG* is advantageous in its controllable time complexity and its promising accuracy, especially when the segment size is consistent. The experiment results also show that the performance of *DPSO-SEG* can be further improved by inserting a good initial particle as in *DPSO-SEG$_{C99}$*.

In the future, we will try to apply more efficient and more precise linear text segmentation algorithms, such *TSF* [12], to generate a better initial solution. It may not only improve the accuracy of linear text segmentation but also make the computation more efficient. Further verifications are also required to evaluate the time and cost effectiveness of *DPSO-SEG*.

Moreover, we found there is always certain amounts of noise appear in sentence similarity matrix, and their existence tends to blur the boundaries of the segment regions. In the future, a deblurring noisy method, such as anisotropic diffusion [9], will also be added to *DPSO-SEG* to deal with such problem.

TABLE III. COMPARISONS OF P_K VALUES.

	3-11	3-5	6-8	9-11
TextTiling	46%	44%	43%	48%
C99	13%	18%	10%	10%
U00	11%	13%	6%	6%
TopSeg	10.74%	7.44%	7.95%	6.65%
DPSO-SEG	12.5%	8.3%	7.6%	6.9%
DPSO-SEG$_{C99}$	12.2%	7.3%	5.7%	5.7%

REFERENCES

[1] D. Beeferman, A. Berger, and J. Lafferty, "Statistical models for text segmentation", Machine learning, 1999, 34(1), pp. 177-210.

[2] F.Y.Y. Choi, "Advances in domain independent linear text segmentation", in Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference, 2000, pp. 26-33.

[3] X. Cui and T.E. Potok, Document Clustering Analysis Based on Hybrid PSO+K-means Algorithm, Journal of Computer Sciences, , Special issue on Efficient heuristics for information organization, 2005, pp. 27-33.

[4] A. El-Gallad, M. El-Hawary, A. Sallam and A. Kalas, "Enhancing the particle swarm optimizer via proper parameters selection", in Canadian Conference on Electrical and Computer Engineering, 2002, pp. 792-797.

[5] P. Fragkou, V. Petridis, and A. Kehagias, "A dynamic programming algorithm for linear text segmentation", Journal of Intelligent Information Systems, 2004, 23(2), pp. 179-197.

[6] W.N. Francis, and H. Kucera, "Frequency Analysis of English Usage: Lexicon and Grammar", Houghton Mifflin, Boston, Mass., 1982.

[7] M.A. Hearst, "Multi-paragraph segmentation of expository text", Proceedings of the 32nd annual meeting on Association for Computational Linguistics, 1994, pp. 9-16.

[8] M.A. Hearst, "TextTiling: Segmenting text into multi-paragraph subtopic passages", Computational linguistics, 1997, 23(1), pp. 33-64.

[9] X. Ji, and H. Zha, "Domain-independent text segmentation using anisotropic diffusion and dynamic programming", in Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, 2003, pp. 322-329.

[10] J. Kennedy, and R.C. Eberhart, "Particle swarm optimization", in Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942-1948.

[11] J. Kennedy, and R.C. Eberhart, "A discrete binary version of the particle swarm algorithm", in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 1997, pp. 4104-4108.

[12] R. Kern, and M. Granitzer, "Efficient linear text segmentation based on information retrieval techniques", in Proceedings of the International Conference on Management of Emergent Digital EcoSystems, 2009, pp. 167-171.

[13] S. Lamprier, T. Amghar, B. Levrat and F. Saubion, "Seggen: A genetic algorithm for linear text segmentation", in International Joint Conference on Artificial Intelligence (IJCAI'07), 2007, pp. 1647-1652.

[14] S. Lamprier, T. Amghar, B. Levrat and F. Saubion, "Toward a More Global and Coherent Segmentation of Texts", Applied Artificial Intelligence, 2008, 22(3), pp. 208-234.

[15] Y. Mehdad, and B. Magnini, "Optimizing Textual Entailment Recognition Using Particle Swarm Optimization", in Proceedings of the 2009 Workshop on Applied Textual Inference, 2009, p. 36-43.

[16] A. Mihaila, A. Mihis, and C. Mihaila, "A genetic algorithm for logical topic text segmentation", International Conference on ICDIM 2008, 2008, pp. 500-505.

[17] J.C. Reynar, "An automatic method of finding topic boundaries", Association for Computational Linguistics Morristown, 1994, pp. 331-333.

[18] Y. Shi, and R. Eberhart, "Parameter selection in particle swarm optimization", 7th International Conference on Evolutionary Programming VII, 1998, pp. 591-600.

[19] M. Utiyama, and H. Isahara, "A statistical model for domain-independent text segmentation", in Proceedings of the 39th Annual Meeting on Association for Computational Linguistics, 2001, pp. 499-506.

[20] N. Ye, J. Zhu, Y. Zheng, M.Y. Ma, H. Wang and B. Zhang, "A Dynamic Programming Model for Text Segmentation Based on Min-Max Similarity", Information Retrieval Technology, 2008, pp. 141-152.