IPNWPSR: Iterated Partial Neighbour Word Plane Sweep Replicated, Evolution of Searching in Partial Tandem Replicated Sequence on Plane Sweep Algorithm

Arash Ghorbannia Delavar
Department of computer engineering, Payame Noor University, Po Box 19395-3697 Tehran, Iran

Elahe Moghimi Hanjani
Department of computer engineering, Payame Noor University, Po Box 19395-3697 Tehran, Iran

Vahe Aghazarian
Islamic Azad University ,Central Tehran Branch, Tehran, Iran

ABSTRACT

This paper describes a method to optimize the plane sweep algorithm. The goal of this paper is to develop a method that reduces the comparison through removing the tandem replicated word comparison and also using partial search technique in a document for escaping from the keywords that are ineffective. The approach introduces Plane Sweep algorithm that is the base algorithm used to search for keywords. Reducing the search area, change the number of keyword's comparisons in a document and speed up our search algorithm. So searching operation is done in a smaller space and we don’t need to search all the keywords in a document. In this algorithm, we make a new technique to create the algorithm that detect the number of tandem replicated words in a document and also searching on a target part, thus reducing the number of keywords in a document speed up our search algorithm.

In proposed algorithm time complexity with lower order has been created than the basic algorithm. Searching for results occurs in a reduced space and it has led to a better performance without comparing all the keywords in the list. The algorithm is robust, and highly effective especially in a high volume of data.

Keywords

Plane sweep algorithm, Replicated data, String matching, Optimized algorithm, partial search, Text retrieval, Proximity search.

1. INTRODUCTION

It is difficult to get documents which are mostly related to the query. Finding this communication process is offline and the word should be searched, so we're looking for a good way. The issue for the search engine is to find the relevant documents and show the relevant ones first.

The approach that plane sweep algorithm is taken to the problem is that we consider the relation of the keywords which are in the neighborhood in a document, so we use the position of keyword instead of word itself. We use proximity search which means finding parts containing a specified collection of keywords [2, 4, 5, 18, 19]. In plane sweep algorithm we have to store the position of keywords, one of the most frequent approaches is inverted files. We have attempted in this paper to reduce the number of checked condition through saving frequency of the tandem replicated words, and also using partial search so that IPNWPSR will be performed in less time at high data storage. The basic reason why replicated data show up in string matching is that it is likely to slow down the basic plane sweep algorithm in a case which a word overlap itself repeatedly, We count the number of ordered pairs of symbols that are adjacent in the document. The essential idea of iterated partial Search in a document lies in focusing the search not on the full space of solutions but on a smaller subspace defined by the solutions. We repeatedly pick a range and flipped unsatisfied keywords. In this approach document is separated into several parts.

Moving iteratively builds a sequence of solutions generated by the IPNWPSR algorithm. It requires finding all the partial range for a given offset list using the technique to find the keyword with minimum repeat.

Running time of IPNWPSR Algorithm can be achieved in time $\mathrm{o}((n-\alpha)\log k)$, where n is the frequency of keywords occurrence in a document, α is the frequency of tandem replicated data and k is the number of query terms in a query.

2. RELATED WORK

Keyword proximity searching in a document is the method to find the relevant document that all the terms in a query appear with in a relatively small fixed-size window. In string matching, there are some results on finding *k* keywords within a given maximum distance *d*.Gonnet etal. proposed an algorithm for finding two keywords *P*1 and *P*2 within distance *d* in $\mathrm{o}(m_1+m_2)$ time, where $m_1 < m_2$ are the numbers of occurrences of the two keywords. Baeza-Yates and Cunto proposed the abstract data type Proximity and an $\mathrm{o}(\log n)$ time algorithm, but the construction takes $\mathrm{o}(n^2)$ time. Manber and Baeza-Yates also proposed an $\mathrm{o}(n\log n)$ time algorithm, but it takes $\mathrm{o}(dn)$ space [3,4].

They assume that the maximum distance *d* is known in advance. Sadakane and Imai proposed an $\mathrm{o}(n\log k)$ time algorithm for a restricted version of the problem. Their version of the problem corresponds to the basic proximity score. As far as we know, this is the only result for the *k*-keyword proximity problem. Plane sweep algorithm achieves the same time complexity while dealing with a generalized version of the problem [4, 5].

In our algorithm, we use the repetitive structures in which many copies of a word appear consecutively and also limit the searched area to a minimum with escaping from some ineffective keywords, which reduce running time in plane sweep algorithm, so that we present an optimal algorithm.

3. PROPOSED IPNWPSR ALGORITHM

We try to have a relationship based on query and keywords that we find in the search document, and also we need to reduce the number of comparisons so that search operation performs faster. The proposed algorithm reduces the searched area to a minimum and relies on an optimized search algorithm for effectively pruning the search space.

The most exceptional search engine would not provide good quality results if the original keywords selected by the user were not suitable. Therefore, the proposed algorithm aims at searching on a set of alternative keywords generated based on a user's original keywords to help a user in his/her subsequent search activities and reduce the time with using the relative range in searching.

Plane sweep algorithm considers that keywords which appear in the neighborhood in a document are related. Therefore we use position of keywords in a document as the unit of queries. By considering keyword positions we can find a paragraph or a sentence in a document which describes what we want to know. We define ranks of regions in documents which contain all specified keywords in order of their sizes. This is called proximity search. [5]

Definition1. Given k keywords $W_1, W_2, ..., W_k$, a set of lists $K = \{K_1, K_2, ..., K_k\}$ where the i-th keyword W_i, and positive integers $f_1, f_2, ..., f_k$, and $k(\leq k')$, the generalized *k* keyword proximity problem is to find the smallest range that contains *k* _ distinct keywords $W_i (1 \leq i \leq k)$ appear at least f_i times each in the range.

Note that the problem becomes the basic plane sweep algorithm when all $f_i = 1$, for $1 \leq i \leq k$.

Definition2. Let (X, d) be a metric space and $R^* \in D$ the set of offsets, a range query $ffset[Min(Q)] = q$, $(q,r), q \in D, r \in R^+$, reports all keywords that are within distance *r* to *q*, that is $(q,r) = \{u \in R^*, d(u,q) \leq r\}$. The volume defined by (q,r) is the range space, and all the keywords from R^* are reported.

The proposed algorithm can be implemented using range queries.

Definition3. A range space is a set $\Sigma = (D, R^*)$, where D is the search space and R^* is the family of subset of D. The elements of R^* are ranges of Σ, where Σ is a finite range. In optimization queries, we want to return an object that satisfies certain condition with respect to the query range. Ineffective searches have no effect on the result. So we have:

$$IPWPSR(D) = \bigcup_{R^* \in D} (R^*),$$

$$IPWPSR(D) = \{X \in D \mid X = R^*_{j1}, R^*_{j2}, ..., R^*_{jk}\}$$

D is the search space which contains ranges that can be matched with chains of basic moves.

- Assume that if we show each offset list with w and frequency as f, we have, $w[i] = w[i+f]$ $, 1 \leq i \leq |w| - \alpha$. The offset of w for replicated tandem word is the offset of the first word location.
- And also assume that if we show distance factor with $\mathrm{DF}_{\mathrm{Minkeyword}}$ which is the distance between two minimum keyword and defined as $\mathrm{DF}_{\mathrm{Minkeyword}} = \lfloor DF_{K_{1,2}}, DF_{K_{2,3}}, ..., DF_{K_{n,m}} \rfloor$, where m=#ofMinKeyword, according to our condition it might be lower than $|Q| - 1$, where $|Q|$ is the length of query, in this situation the search range would not overlap.

In the proposed algorithm, we have set the offset based on sorted keywords position. Offset is a distance from the beginning of a document [3]. It can use Inverted file or suffix arrays, mostly Inverted file is used in this context, which is a mapping of words to their place in the document. It takes a user's query as input and returns a set of documents sorted by their relevance to the query.

We reduce the number of comparisons with counting the number of tandem replicated words and also removing unsatisfied searches. We have tried to find replicated words in a list of tandem words with the specified offset which is the output stage of the preprocessing .The main objective is to find the most efficient and relevant answer for the query. There are so many results which contain the query's keywords but users are interested in a much smaller subset. For this purpose we define a range as R^* where it contains nearest neighbors of minimum counted keyword, given the IPNWPSR, we can perform a local search in R^*. We want to explore R^* using a walk that steps from one R^* to a "nearby" one, with the list of defined minimum keyword. This algorithm escapes from ineffective keywords by applying the current partial range.

We consider the keywords which have minimum counter and limit our search area to the range around that keyword. Sometimes, there are few keywords in the document with the minimum number. We consider another factor as distance factor. The distance factor is the number of locations between the two minimum keywords, the value must be greater than the Query length ($|Q| - 1$) otherwise the search range may overlap and search results will not be optimal, if this factor is also the same, we consider one of the keywords with minimum number randomly and the search range limited to the distance around that keyword.

The implementation of this algorithm is intended to count the number of tandem repeat words in a document using inverted file and also limit the search range using the technique of partial range. An inverted file is an index structure which stores a mapping from words to their location in a document or a set of document allowing full text search [5]. We compare each repeat word with the preceding word. So the number of repetitions is reduced and this causes the reduction of search time. The algorithm is applied on the sorted list of the keywords position in a document. In this algorithm, we need that, size of the search interval is specified, for this purpose, a critical range is defined. Range size, is the number of words in which located in a query, finding the range of critical areas is depend on finding a candidate range in a document such a way that no other range include it (Candidate range include the minimum k keyword in query).

In this algorithm, two pointers are used to search a document that Scan offset list from left to right, P_l which refers to the left of the offset and P_r which refers to the last offset in the range.

As we defined R^* as a searched range, Iterated partial search achieved the results as follows:

Set P_l, P_r with the position of the first R^*, then iteratively moves to the next R^* that contains the item of interest. It means the ranges that are irrelevant for the result is removed. P_r moves in a defined range, if its value is greater than the range size, it returns to the next beginning of the interval which P_l is pointed. After a critical range was set, P_l also move forward one place at the offset list. In this algorithm, counter is stored for each offset value. And if its value is greater than one, compare operations do not need to do and we only move forward according to the size of the defined range, the IPNWPSR algorithm doesn't consider all position of word in a document. In fact, we skip the repetitive sequences. After each critical range is defined, minimality is also checked. This continues until we reach end of the list. Our work tries to improve the plane sweep algorithm by efficiently calculating the minimal group of words as a result.

In this section we formalize the IPNWPSR algorithm for searching. As a consequence of the IPNWPSR algorithm and basic plane sweep and number of tandem replicated word, and also removing the comparison of ineffective keywords, number of comparisons related to IPNWPSR algorithm can be formulated as follows:

$$C_n(IPNWPSR) = \sum_{k=1}^{|\#ofMinKeyword|} \left(\sum_{j=1}^{|D|} \left(\sum_{i=1}^{|Q|} (i \times \pi_i) \right) \right) - \sum_{j=1}^{|\alpha|} \left(\sum_{i=1}^{|Q|} (i \times \pi_i) \right) \quad (1)$$

As shown in formula (1), number of comparison in IPNWPSR is the sum of all ranges which in each range number of comparison is the difference between the number of comparison in plane sweep algorithm and the number of comparison in replicated words. So according to Formula (1), we reduce the number of comparison in our proposed algorithm.

|D| = Length of Document offset list - |Q| -1

|Q| = Length of Query

$$\pi_i = \begin{cases} 1 & \text{if match occurred in Doc list} \\ 0 & \text{O.W} \end{cases}$$

Let π_i denote the Availability factor of a word in a query, which is describing the number of comparisons made. π_i set to one if position i considered with the algorithm, otherwise, the number of comparison is zero and it is also set to zero, so only position i is applied in Formula(1).
The number of comparison for replicated word is calculated from following equation:

$$\alpha = \sum_{k=1}^{|D|} (freq_k - 1), \text{ if k>1.} \quad (2)$$

α = Number of Replicated word

The algorithm makes a scan over the offsetlist. The pointers represent the start and end position of a range in the offsetlist. The first loop advances until it exceeds the end of range lists and the second loop is used to move forward in each range. As shown in Fig.1 the algorithm is used to generate quick answer to user without a need to invoke the full searching process.

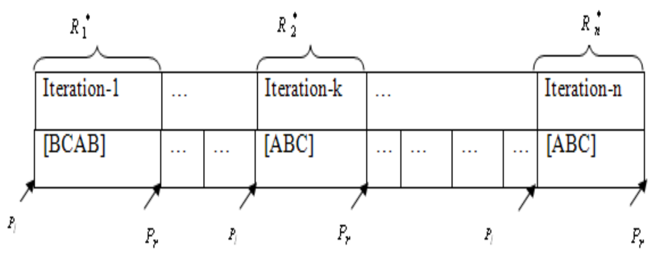

Fig.1 Iteration in ranges in the offsetlist

The following diagram shows general scheme of the proposed IPNWPSR range selection approach. It consists of two main stages. In the first stage, we get the minimum keyword in a query and define search range. Then, in the second stage, called partial search stage, several ranges extract a candidate answer and its corresponding support text for a given question, it evaluates all candidate answers. At the end, the correct answer having the greatest confidence value is selected as the final response. In the case that we have no relevant range, the system returns a nil response.

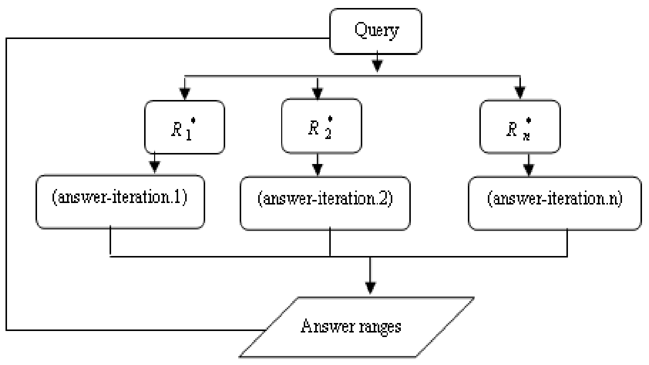

Fig.2 General diagram of IPNWPSR range selection answering process.

Definition4. Let R_D be the precision of the ranges that are relevant to the query in the IPNWPSR algorithm and also retrieved from the plane sweep algorithm, in which a better solution has been found. R_D in a document can be defined as the conditional probability which denotes the probability that ranges have within a document. This parameter shows the ratio of relevant ranges that were retrieved from the IPNWPSR and also plane sweep algorithm.

$$R_D = \frac{\#(IPNWPSR - \text{Re}treived.ranges)}{\#(Plane - sweep.Candidate.ranges)} \quad (3)$$

$$R_D = \frac{|IPNWPSR - \mathrm{Re}levent.ranges \cap Plane - sweep.candidate.ranges|}{|Plane - sweep.Candidate.ranges|}$$

Every repeat within ranges must span the minimum characters of some substring in every iteration and the forbidden combinations of adjacent ranges are removed.
The purpose of the proposed IPNWPSR algorithm is to reduce the efforts in identifying appropriate keywords set to allocate the desired documents more efficiently.
We remove the sequences which contain the replicated data and allow related ranges. As shown in Fig.3, the algorithm is illustrated below.

Definition5. A 'tandem replicated word' is a string of the form $X^{s_\alpha} Z^{y_\alpha} ... X^{s_\alpha} W^{q_\alpha}$
Where $s_\alpha, y_\alpha, q_\alpha \geq 1$ for some $\alpha \in \{1,2,..., n\}$.

The concatenations of the tandem replicated word is as follows:
Document :{ BCCCCABABABC}

$$\prod_{i=1}^{13} W^{q_i} X^{r_i} Z^{y_i} W^{q_i} Z^{y_i} W^{q_i} Z^{y_i} W^{q_i} X^{r_i}$$

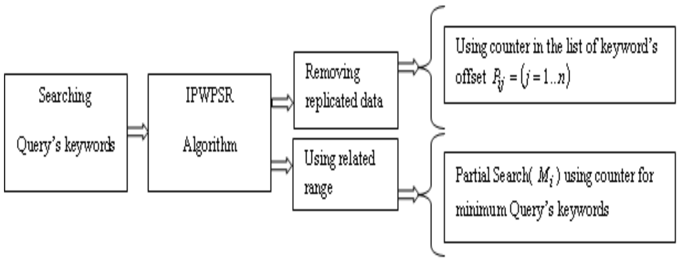
Searching Query's keywords
IPWPSR Algorithm
Removing replicated data
Using related range
Using counter in the list of keyword's offset $P_{ij} = (j = 1...n)$
Partial Search(M_i) using counter for minimum Query's keywords

Fig.3 Searching IPNWPSR algorithm process

Where $q_i \in \{1\}$; $r_i \in \{1,4\}$; $y_i \in \{1\}$(according to Definition5).

In this section we describe a proposed algorithm and show its improved average running time. IPNWPSR algorithm is defined below:

(1) Sort offset of keyword $P_{ij} = (j = 1...n)$ in a document in increasing order, we also add counter for replicated tandem words to the list.
(2) Add the number of each query's keywords in a offsetList
(3) Get the minimum replicated keyword from the list considering the distance factor
(4) Repeatedly increase i by one until we get end of the Min-keyword offsetList
(5) If we have passed end of the list, sort interval in a heap with considering the tandem replicated word counter and output them, finish.
(6) Repeatedly increase P_l by one until the current range is a candidate range or we have passed the end of the list.
(7) If we have passed end of the partial list, go to step 4.
(8) Repeatedly increase P_r by one until the current range is not a candidate range.
(9) The range $(P_l\ , P_{r-1})$ is a critical range. Compare the size of range $(P_l\ , P_{r-1})$ with the stored minimum range and replace the minimum range with the current range if $(P_l\ , P_{r-1})$ is smaller.
(10) Go to step 4.

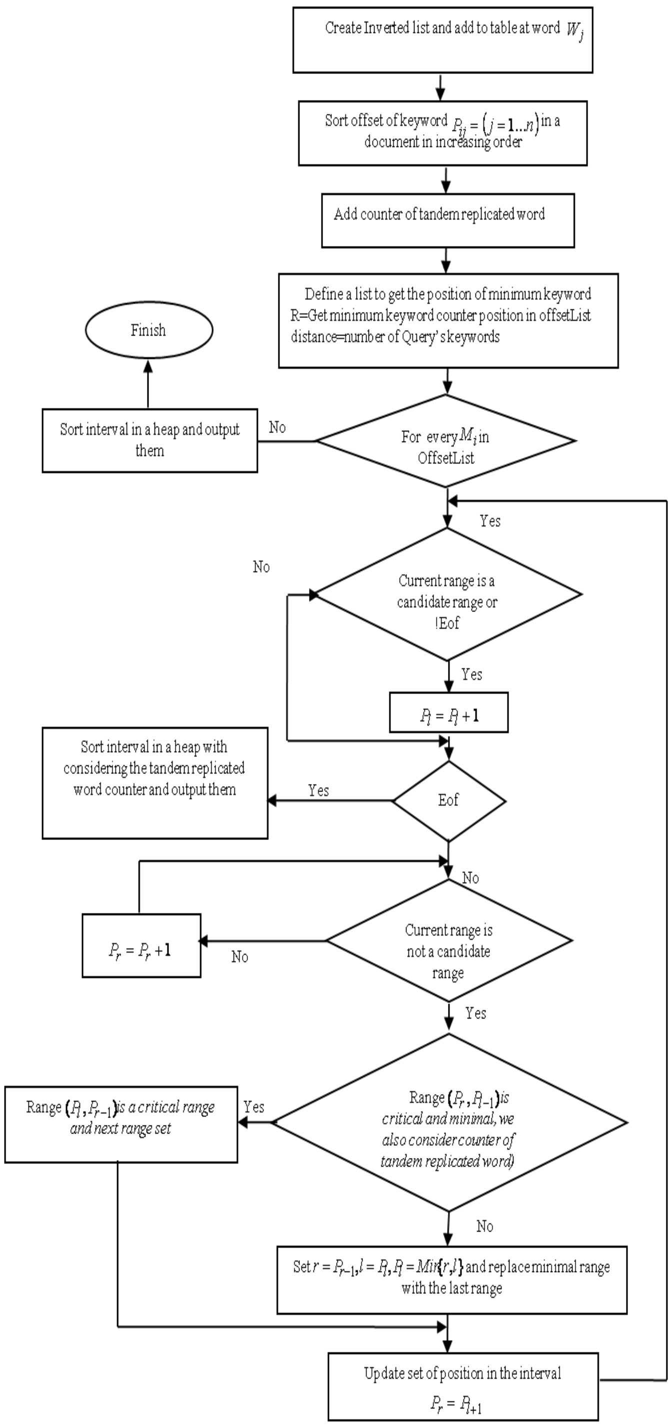
Create Inverted list and add to table at word W_j
Sort offset of keyword $P_{ij} = (j = 1...n)$ in a document in increasing order
Add counter of tandem replicated word
Define a list to get the position of minimum keyword R=Get minimum keyword counter position in offsetList distance=number of Query's keywords
Finish
Sort interval in a heap and output them
No
For every M_i in OffsetList
Yes
No
Current range is a candidate range or !Eof
Yes
$P_l = P_l + 1$
Sort interval in a heap with considering the tandem replicated word counter and output them
Yes
Eof
No
$P_r = P_r + 1$
No
Current range is not a candidate range
Yes
Range (P_l, P_{r-1}) is a critical range and next range set
Yes
Range (P_r, P_{l-1}) is critical and minimal, we also consider counter of tandem replicated word)
No
Set $r = P_{r-1}, l = P_l, P_l = Min\{r,l\}$ and replace minimal range with the last range
Update set of position in the interval $P_r = P_{l+1}$

Fig.4 IPNWPSR algorithm flowchart

To illustrate the searching procedure we present an example:

Suppose we have a query of "BCA" that is searched in the following document offset list, the algorithm needs to compute all the related keywords:

Document :{ BCCCCABABABC}

Any offset in the following list are:

$K_1 \in \{0,6,8,10\}$,

0 6 8 10

Keyword in Doc	B	C	A	B	A	B	A	B	C
# of repeated words	1	4	1	1	1	1	1	1	1

Fig.5 Searching ranges on $K_1 \in \{0,6,8,10\}$

$K_2 \in \{1,11\}$ and

1 11

Keyword in Doc	B	C	A	B	A	B	A	B	C
# of repeated words	1	4	1	1	1	1	1	1	1

Fig.6 Searching ranges on $K_2 \in \{1,11\}$

$K_3 \in \{5,7,9\}$.

5 7 9

Keyword in Doc	B	C	A	B	A	B	A	B	C
# of repeated words	1	4	1	1	1	1	1	1	1

Fig.7 Searching ranges on $K_3 \in \{5,7,9\}$

Search has been defined from left to right with respect to the range for 3-keyword search operation is performed. The minimum keyword in this document is C, the distance factor is $DF_C = [6]$, where $DF_{C_{1,2}}$ is greater than the query length so the range is acceptable .There are two partial range in the offset list, R_1^* and R_2^*.

Where :

$R_1^* = BCAB, R_2^* = ABC$

Finding the solution for the above example:

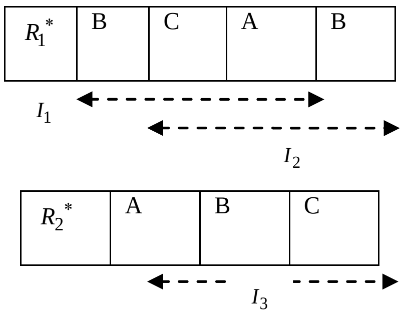

Fig.8 Result of IPNWPSR algorithm is shown as I_1, I_2, I_3

R_1^*, R_2^* are the partial ranges that are the output of the IPNWPSR algorithm and I_1, I_2, I_3 are the result range, which I_1, I_2 are in R_1^* and I_3 in R_2^*.

Here is an efficient example:

Document :{ ABABABABACCC}

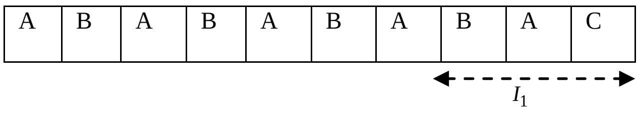

Fig.9 An efficient result of IPNWPSR algorithm

It is one of the best examples which show the effectiveness of the algorithm better. In plane sweep algorithm we should trace the offsetlist from beginning to end but in this situation we just trace the target range which is I_1 in the above example.

But somehow in a situation which all the keyword in a query repeated, it works the same as plane sweep algorithm. Here is the example:

Document :{ ABCABCABCA}

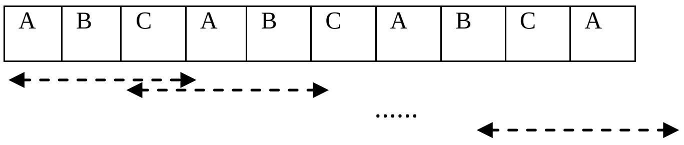

Fig.10 Ranges that show the result in situation which is similar to plane sweep algorithm

4. TEST RESULTS

The comparison between Plane Sweep Algorithm with WPSR[1] and IPNWPSR (Table 1), Show that the algorithm has been worked better, especially in high volume repetitive data. Information in Table 1 are used to generate the data sets. The data sets and test results are used to assess performance measures for the algorithm under test. We compared the three search algorithm on the data set.

Table1: Results of the tests on these offsets

Datasize	Plane sweep	WPSR	IPNWPSR
900	0.0239	0.0226	0.0225
1800	0.0424	0.0356	0.0352
3600	0.0775	0.061	0.0605
7200	0.1449	0.1162	0.1124
14400	0.2751	0.2199	0.2157
28800	0.5584	0.4395	0.4234

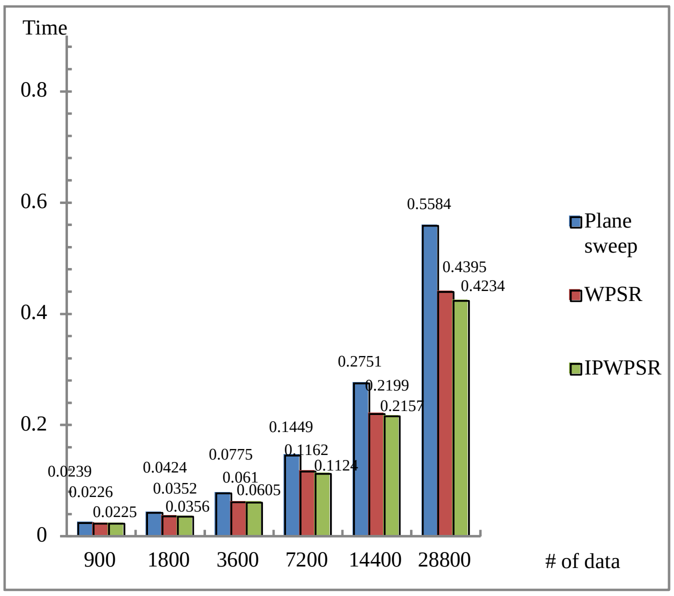

Fig. 11 Number of Comparisons made by WPSR, IPNWPSR and plane sweep algorithm with 3-keywords query and $W_{sim} = 0.4$.

Table2: Results of the tests on these offsets, 4-keywords, R_D =0.6, $W_{sim} = 0.5$

Datasize	Plane sweep	IPNWPSR
1080	0.0325	0.01887
2160	0.0427	0.0274
4320	0.0774	0.0446
8640	0.1509	0.0809
17280	0.3095	0.1545
34560	0.5941	0.3063

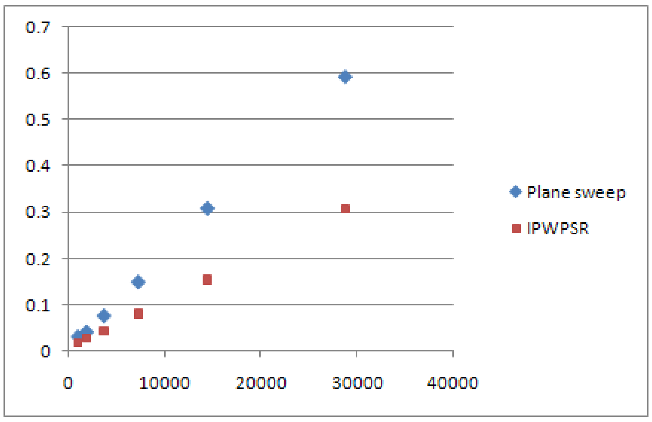

Fig. 12 Number of Comparisons made by WPSR and plane sweep algorithm with 4-keywords query.

Table3: Results of the tests on these offsets, 3-keywords, R_D =0.7, $W_{sim} = 0.6$.

Datasize	Plane sweep	IPNWPSR
1080	0.0292	0.0245
2160	0.0526	0.0305
4320	0.1285	0.0502
8640	0.1907	0.0931
17280	0.3976	0.1898
34560	0.7350	0.3602

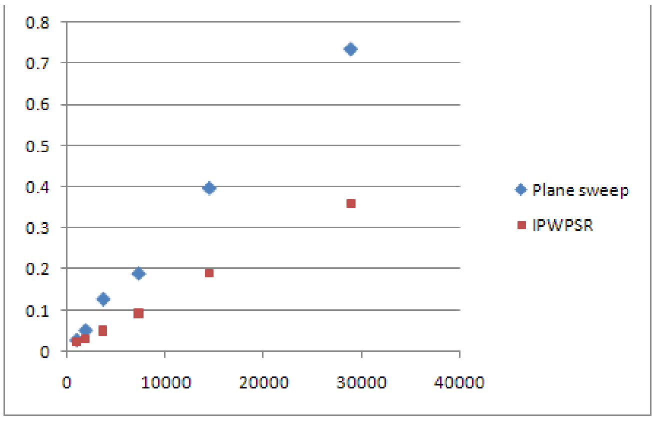

Fig. 13 Number of Comparisons made by IPNWPSR and plane sweep algorithm with 3-keywords query.

The diagram shows the offset list of document's words, where the repetition factor (W_{sim}), is defined below:

$$W_{sim} = \frac{\sum |\alpha|}{|D|}, 0 \le W_{sim} \le 1 . \quad (4)$$

Where $|D|$, is the size of the offset list, and $|\alpha|$ is the frequency of replicated words in the list.

In order to study the effect of tandem replicated word with iterative partial range, we ran some experiment on a different lists size, and the result is shown above. Experiments views the sequence as it has been produced by a random file with the specific repetition factor of tandem replicated word.

Runtime Analysis aims to determine, the time a search algorithm needs to find an optimal solution. In this study, it must be noted that the run time depends not only on repetition factor of the input document, but also on the number of keywords in query. From fig.11 we observe that using tandem replicated word with iterative partial range make all size of the random sample better, especially in a large size, IPNWPSR leads to a better running time.

The proposed algorithm is expected to improve search accuracy and effectiveness for a novice user in the field of interest. For experience users, with the fast-growing availability of information online, who may not be aware of most the updated critical keywords, the proposed system is also expected to improve search efficiency.

Finally, we see an improvement in IPNWPSR algorithm as a result of the changes in plane sweep algorithm. However, the improvement varies with the test condition. Furthermore, the proposed system is flexible and can easily be integrated with other search algorithms to improve search results.

5. CONCLUSION

We have introduced IPNWPSR algorithm for escaping from ineffective and also tandem replicated keywords. The approach introduces Plane Sweep algorithm, that is the base algorithm used to search for keywords and removing the tandem replicated word comparisons and also using partial search technique for escaping from the keywords that are ineffective.

Experimental results show that the algorithm performs well in practice, since it is not only reduces the comparison but also speed the search algorithm. The effect of algorithm on high volume of data is more significant, and also the algorithm is robust, and highly effective.

5. REFERENCES

[1] Arash Ghorbannia Delavar, Elahe Moghimi Hanjani, WPSR: Word Plane Sweep Replicated, Present a Plane sweep Algorithm to optimize the use of Replicated data in a document, International Journal of Computer Science Issues, Vol. 9, Issue 2, No 3, 2012.

[2] Alan Feuer, Stefan Savev, Javed A.Aslam, "Implementing and evaluating phrasal query suggestions for proximity search", Elsevier, College of Computer and Information Science, 2009.

[3] Feuer Alan, Savev Stefan, Javed A. Aslam, Evaluation Of phrasal query suggestions, in: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (CIKM '07), Lisboa, Portugal, 2007.

[4] K. Sadakane, H. Imai, Fast algorithms for *k*-word proximity search, IEICE Trans. Fundamentals E84-A (9) (September 2001) 312–319.

[5] Chirag Gupta, Gultekin Ã–zsoyoglu, Z. Meral Ã–zsoyoglu. Efficient k-word proximity search. In The 24th International Symposium on Computer and InformationSciences, ISCIS 2009, 14-16 September 2009, North Cyprus. pages 123-128, IEEE, 2009.

[6] Zobel J.,Moffat A.,Inverted files for text SearchEngines,ACM computing surveys(C SUR),v382,2006.

[7] B.J. Jansen, A. Spink, T. Saracevic, Real life, real users, and real needs: a study and analysis of user queries on the web, Inf. Process. Manage. 36 (2) (2000) 207–27.

[8] S.Kim,I.Lee,K.Park,A fast algorithm for the generalized k-keyword proximity problem given keyword offset , Inf.Process. Lett. 91(3)(2004)115–120.

[9] S. Lawrence, C.L. Giles, Searching the web: general and scientific information access, IEEE Communications 37 (1) (1999) 116–122.

[10] Atheer A. Matroud1, M. D. Hendy and C. P. (2011) Tuffley NTRFinder: a software tool to find nested tandem repeats.

[11] R. Uricaru, A. Mancheron, E. Rivals, Novel definition and algorithm for chaining fragments with proportional overlaps, J. of Computational Biology, Vol. 18(9), p. 1141-54, 2011.

[12] E. Adebiyi, E. Rivals, Detection of Recombination in Variable Number Tandem Repeat Sequences, South African Computer Journal (SACJ), 39, p. 1–7, 2007.

[13] E. Rivals, A Survey on Algorithmic Aspects of Tandem Repeat Evolution, International Journal on Foundations of Computer Science, Vol. 15, No. 2, p. 225-257, 2004.

[14] Atheer A. Matroud **,** Michael D. Hendy **,** Christopher P. Tuffley **,**An algorithm to solve the motif alignment Problem for approximate nested tandem repeats, RECOMB-CG'10 Proceedings of the international conference on Comparative genomics, 2010.

[15] Hongxia Zhou, Liping Du, Hong Yan, Detection of tandem repeats in DNA sequences based on parametric spectral estimation. IEEE transactions on information echnology in biomedicine a publication of the IEEE Engineering in Medicine and Biology Society (2009).

[16] G M Landau, J P Schmidt, D Sokol, An algorithm for approximate tandem repeats.Journal of computational biology a journal of computational molecular cell biology, (2001).

[17] Rasolofo Yves, Savoy Jacques, Term proximity scoring for keyword-based retrieval systems, 25th European Conference on IR research, ECIR 2003.

[18] Hao Yan, Shuming Shi, Fan Zhang, Torsten Suel, Ji-rong Wen, Efficient Term Proximity Search with Term-Pair Indexes, the 19th ACM conference on Conference on information and knowledge management CIKM 10 (2010).

[19] Ralf Schenkel, Andreas Broschart, Seungwon Hwang, Martin Theobald, Gerhard Weikum, Efficient Text Proximity Search, String Processing and Informatio Retrieval (2007).

[20] Navarro, G. and Salmela, L. 2009. Indexing variable length substrings for exact and approximate matching. In

Proceedings of the 16th International Symposium on String Processing and Information Retrieval (SPIRE'09). Springer, Berlin, 214--221.

[21] Inbok Lee, Sung-Ryul Kim: An Algorithm for the Generalized *k*-Keyword Proximity Problem and Finding Longest Repetitive Substring in a Set of Strings. International Conference on Computational Science (4) 2006: 289-292,2006.

[22] Taehyung Lee, Sung-Ryul Kim, Kunsoo Park: Approximate Word Sequence Matching on an Inverted File Index. IWOCA 2008: 100-114,2008.

[23] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2), 2006.

[24] Hao Yan, Shuming Shi, Fan Zhang, Torsten Suel, Ji-rong Wen, Efficient Term Proximity Search with Term-Pair Indexes, the 19th ACM conference on Conference on information and knowledge management CIKM 10 (2010).