WPSR: Word Plane Sweep Replicated, Present a Plane sweep Algorithm to optimize the use of Replicated data in a document

Arash Ghorbannia Delavar[1] and Elahe Moghimi Hanjani[2]

**[1] Department of computer, Payame Noor University,
Po Box 19395-3697 Tehran, Iran**

**[2] Department of computer, Payame Noor University,
Po Box 19395-3697 Tehran, Iran**

Abstract

This paper describes the plane sweep algorithm for optimal use of tandem replicated data in a document. It is based on an observation that the original plane sweep algorithm is used to search in documents. Plane sweep algorithm does not feature a fast algorithm to find the word that is repeated in tandem in a document.

With the help of effective parameters we could make a new technique to create the algorithm that detect the number of tandem replicated words in a document and reduce the number of compares, thus reducing the number of keywords in a document speed up our search algorithm. For this purpose we need to have a link between words and documents that the proposed algorithm WPSR provides a similar solution as original plane sweep algorithm. However, considering the volume of data we get the canonical form that this situation helps us to increases recognition of duplicate words in large scale. In proposed algorithm time complexity with lower order has been created than the basic algorithm, also terms of the algorithm with the WPSR system reliability, consider the best web search. Finally, system efficiency and performance increase more than the previous similar algorithm in this field.

***Keywords**: Plane sweep algorithm, Replicated data, String matching, Optimized algorithm, Web search, Text retrieval, Proximity search.*

1. Introduction

Strings are kind of data that should be noted in web searching. Find the highest degree of similarity is defined to find the most relevant results, and strings that are searched. For this purpose, various methods can be used. Web pages and links between words in a document will affect the output. It is difficult to get documents which are mostly related to the query.

Find this communication process is offline and the word should be searched, so we're looking for a good way.

For example, when the query is “apple computer support”, what the user intended is most likely the customer support page in the Apple computer site. However, some other documents may contain all three keywords in totally different contexts (irrelevant results). So the issue for the search engine is to find the relevant documents and show the relevant ones first. Many heuristics are used to compute the relevance of a document. Examples include the Page Rank model and the hub and authority model, both based on the links between the documents [3]. 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 [1,3,4,17,18].

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 tandem replicated words, so that WPSR 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.

Running time of WPSR Algorithm can be achieved in time $o((n-\alpha)\log k)$, 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 works

There are several algorithms for finding exact tandem repeats. A number of algorithms consider motifs differing only by substitutions, using the Hamming distance as a measure of similarity. Others have considered insertions and deletions by using the edit distance, and also using DNA. Most of these algorithms have two phases, a scanning phase that locates candidate tandem repeats, and an analysis phase that checks the candidate tandem repeats found during the scanning phase [9]. We use the concept of finding repeated data in preprocessing phase to decrease the running time of plane sweep algorithm.

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 $P1$ and $P2$ within distance d in $o(m_1 + m_2)$ 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 $o(\log n)$ time algorithm, but the construction takes $o(n^2)$ time. Manber and Baeza-Yates also proposed an $o(n \log n)$ time algorithm, but it takes $o(dn)$ space [3,4].

They assume that the maximum distance d is known in advance. Sadakane and Imai proposed an $o(n \log k)$ 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 [3,4].

Plane sweep algorithm consists of two stages, early stage is pre-processing step, at this stage we have a list of offsets that are sorted by keyword in a document, and offsets are labeled using the keyword index (i if the keyword is w_i) during the merge. Because the input lists are already sorted, it is easy to show that the first stage takes $o(n \log k)$ time. The second stage is the main part of the algorithm. The second stage takes $o(n)$ time [3]. The algorithm shows the various combination of inputted keyword in a document.

The algorithm is as follows:

(1) Repeatedly increase P_r by one until the current range is a candidate range or we have passed the end of the list.

(2) If we have passed end of the list, finish.

(3) Repeatedly increase P_l by one until the current range is not a candidate range.

(4) The range $(P_l - 1, P_r)$ is a critical range. Compare the size of range $(P_l - 1, P_r)$ with the stored minimum range and replace the minimum range with the current range if $(P_l - 1, P_r)$ is smaller.

(5) Go to step 1 [1].

In our algorithm, we use the repetitive structures in which many copies of a word appear consecutively and reduce running time in plane sweep algorithm, so that we present an optimal algorithm. In the next section we discussed more about our algorithm.

3. Proposed WPSR Algorithm

We know the questions in the retrieval operation so we just try to find relationship of data according to the document contents so that, even search space is large, we achieve optimal response. When you are looking for a query in a document, it is possible that the same keywords in search results will include query but it has a meaning in totally different context, indeed the result is not what you are expected, We try to have a relationship based on keywords and query that we find in the search document, and also we need to reduce the number of comparison so that search operation performs faster. For this purpose 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 WPSR algorithm. We reduce the number of comparisons with counting the number of tandem replicated words. The main objective is to find the most efficient and relevant answer for the query.

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.

The implementation of this algorithm is intended to count the number of tandem repeat words in a document using inverted file. 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.

If we show each offset list with w and frequency as f, we have, $w[i] = w[i + f], 1 \le i \le |w| - \alpha$. The offset of w for

replicated tandem word is the offset of the first word location.

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, Pointer P_l, which refers to the left of the offset and P_r which is refer to the last offset in the range. 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 WPSR 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.

As a consequence of the WPSR algorithm and basic plane sweep and also number of tandem replicated word, number of comparisons related to WPSR algorithm can be formulated as follows:

$$C_n(WPSR) = C_n(PS) - C_n(\alpha) \quad (1)$$

As shown in formula (1), Number of comparison in WPSR is the difference between the number of comparison in plane sweep algorithm and the number of comparison in replicated words. So according to Formula (5), we reduce the number of comparison in our proposed algorithm.
$C_n(PS)$ and $C_n(\alpha)$ is defined below:

$$C_n(PS) = \sum_{j=1}^{|D|}\left(\sum_{i=1}^{|Q|}(i \times \pi_i)\right), \quad (2)$$

$$C_n(\alpha) = \sum_{j=1}^{|\alpha|}\left(\sum_{i=1}^{|Q|}(i \times \pi_i)\right) \quad (3)$$

|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(2,3).

The number of comparison for replicated word is calculated from following equation:

$$\alpha = \sum_{k=1}^{|D|}(ctr_k - 1), \text{ if k>1.} \quad (4)$$

α = Number of Replicated word

Due to changes, the following formula is reached:

$$C_n(WPSR) = \sum_{j=1}^{|D|}\left(\sum_{i=1}^{|Q|}(i \times \pi_i)\right) - \sum_{j=1}^{|\alpha|}\left(\sum_{i=1}^{|Q|}(i \times \pi_i)\right) \quad (5)$$

Definition 1.
A 'tandem replicated word' is a string of the form

$$X^{s_\alpha} Z^{y_\alpha} \ldots X^{s_\alpha} W^{q_\alpha}$$

Where $s_\alpha, y_\alpha, q_\alpha \geq 1$ for some $\alpha \in \{1,2,\ldots, n\}$.

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

(1) Sort offset of keyword $P_{ij} = (j = 1 \ldots n)$ in a document in increasing order, we also add counter for tandem replicated words to the list.
(2) Repeatedly increase P_r by one until the current range is a candidate range or we have passed the end of the list.
(3) If we have passed end of the list, sort interval in a heap with considering the tandem replicated word counter and output them, finish.
(4) Repeatedly increase P_l by one until the current range is not a candidate range.
(5) The range $(P_l - 1, P_r)$ is a critical range. Compare the size of range $(P_l - 1, P_r)$ with the stored minimum range and replace the minimum range with the current range if $(P_l - 1, P_r)$ is smaller.
(6)Go to step 2[1].

WPSR pseudo code is as follows:

Preprocessing stage:

Create table for all word in document which are similar to the Query's keywords.

i←0, k←0, j←0

While (i< |D|)

//Create inverted list and add to table at word W_j

Add position and also key_i (index in query keywords) to the list.

i←i+1

Sort offset of keyword $P_{ij} = (j = 1...n)$ in a document in increasing order

W_{first} = OffsetList[k]. key_i

While (j< |OffsetList|)

Begin

//add counter of tandem replicated word

if (W_{first} !=null)

//Skip over replicated word

Begin

k←k+1

W_{next} = OffsetList[k]. key_i

End/* *End if**/

if (W_{next} !=null and $W_{next} == W_{first}$)

Begin

Add counter OffsetList[k] by one

Substitute the position of W_{first} with W_{next}

End/**End if**/

j←j+1

End /** End while**/

Main stage:

Set P_r , P_l to addressing the start of the OffsetList

For every P_r in OffsetList

Begin

If (Current range is a Candidate range OR !Eof)

$P_l = P_l + 1$

If (Eof)

Sort interval in a heap with considering the tandem replicated word counter and output them, finish.

If (Current range is not a Candidate range)

$P_r = P_r + 1$

If (Range (P_l, P_{r-1}) is a Critical range and minimal)

Range is a Critical range and next range set

Else

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 $P_r = P_{l+1}$.

End /**End For**/

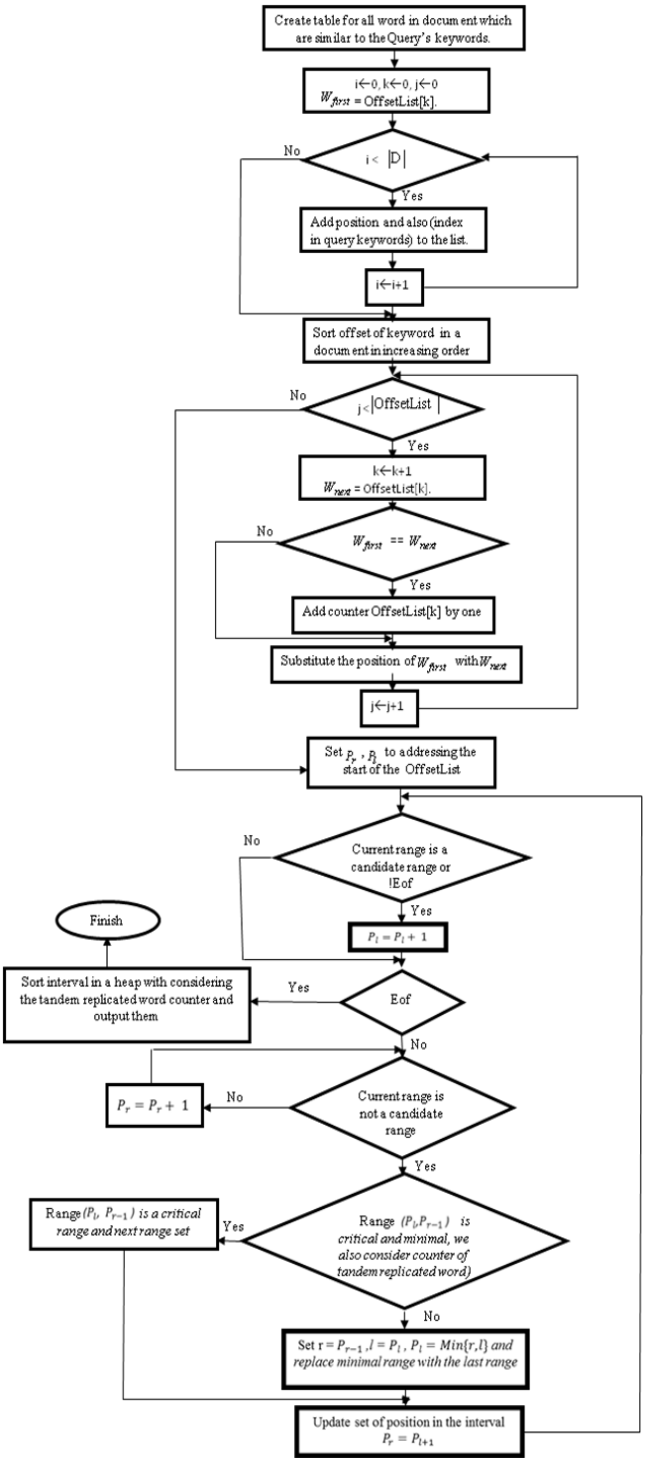

Fig. 1 WPSR algorithm flowchart

Example 1

To illustrate the searching procedure we present an example:
Suppose we have a query of "ABC" that is searched in the following document offset list:

Document :{ AAABCCCCCCCBA}

Table 1: Example of a merged list of WPSR algorithm

Keyword in Doc	A	B	C	B	A
# of repeated words	3	1	7	1	1
Offset	0	3	4	11	12
key	1	2	3	2	1

Any offset in the following list are:

$K_1 \in \{0,12\}$, $K_2 \in \{3,11\}$ and $K_3 \in \{4\}$.
Search has been defined from left to right with respect to the range for 3-keyword search operation is performed.
I_1 and I_2 are above the range that the output of the algorithm.

Table 2: Find the solution for Example 1

word	A	B	C	B	A
# of repeated words	3	1	7	1	1
key	1	2	3	2	1
offset	0	3	4	11	12

I_1 I_2

The concatenations of the tandem replicated word is as follows:

Document :{ AAABCCCCCCCBA}

$\prod_{i=1}^{13} W^{q_i} X^{r_i} Z^{y_i} X^{r_i} W^{q_i}$
Where $q_i \in \{1,3\}$; $r_i \in \{1\}$; $y_i \in \{7\}$ (according to Definition1).

4. Test Results

The comparison between Plane Sweep Algorithm with WPSR (Table 3), Show that the algorithm has been worked better, especially in high volume repetitive data.

Table 3: Results of the tests on these offsets

Data Size(Offset)	Plane Sweep(cpu time)	WPSR (cpu time)
900	0.0306	0.0211
1800	0.0562	0.0350
3600	0.0988	0.0624
7200	0.1889	0.1153
14400	0.3825	0.2145
28800	0.6984	0.3997

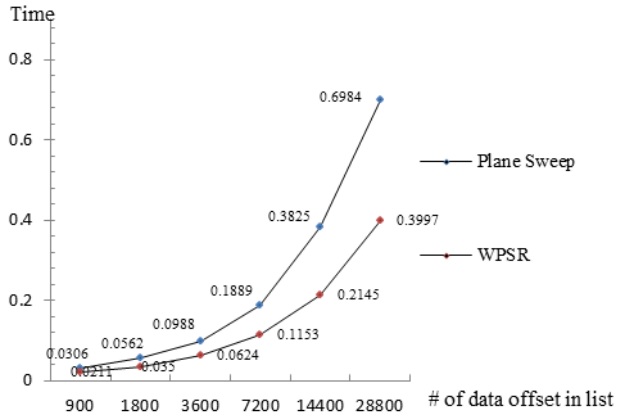

Fig. 2 Number of Comparisons made by WPSR and plane sweep algorithm with 3-keywords query and $W_{sim} = 0.4$.

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

$$W_{sim} = \frac{\sum |\alpha|}{|D|}, 0 \le W_{sim} \le 1. \qquad (6)$$

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

Based on this analysis we compare the new method with the original algorithm of plane sweep. In order to study the effect of tandem replicated word for WPSR algorithm, we ran some experiment on a different lists size, and the result is shown on table3. Experiments views the sequence as it has been produced by a random file with the specific repetition factor of tandem replicated word ($W_{sim} = 0.4$). 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.
In fact, if repetition factor (it means the number of tandem repeat words in the document), and also the number of keywords in question is greater, the WPSR algorithm work better.

From fig.2 we observe that using tandem replicated word make all size of the random sample better, specially in a large size, WPSR leads to a better running time.

5. Conclusions

Finding specific keyword in the search engine optimization is one of the main goals in data extraction. Plane Sweep algorithm, is one of the algorithm that is used to search for keyword. Plane Sweep Algorithm search function is based on a list of keywords offsets. As the occurrence of the tandem replicated word in the search document is possible. So to reduce the search time, the proposed algorithm is presented; studies are shown in Table 3. This algorithm would be optimal particularly in a high volume of tandem replicated data.

References

[1] Alan Feuer, StefanSavev, JavedA. Aslam, "Implementing and evaluating phrasal query suggestions for proximity search", Elsevier, College of Computer and Information Science, 2009.

[2] 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.

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

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

[5] Zobel J.,Moffat A.,Inverted files for text Search Engines,ACM computing surveys(C SUR),v38-2,2006.

[6] 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) 207–227.

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

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

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

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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).

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

[17] 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).

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

Arash Ghorbannia Delavar received his M.Sc. and Ph.D. degree in computer engineering from Sciences and Research University, Tehran, IRAN, in 2002 and 2007. He obtained the top student award in Ph.D. course. He is currently an assistant professor in the Department of Computer Science, Payame Noor University, Tehran, IRAN. He is also the Director of Virtual University and Multimedia Training Department of Payame Noor University in IRAN. Dr. Arash Ghorbannia Delavar is currently editor of many computer science journals in IRAN. His research interests are in the areas of computer networks, microprocessors, data mining, Information Technology, and E-Learning.

Elahe moghimi hanjani received her B.Sc. in computer engineering from Azad University central Tehran branch, Tehran, IRAN, in 2008, and is a M.Sc. student in computer engineering in Payame Noor University. Her research interests include optimizing text retrieval algorithm, datamining.