BIOINFORMATION
Discovery at the interface of physical and biological sciences

open access www.bioinformation.net Prediction model

Volume 9(16)

OPTSDNA: Performance evaluation of an efficient distributed bioinformatics system for DNA sequence analysis

Mohammad Ibrahim Khan & Chotan Sheel*

Department of Computer Science & Engineering (CSE), Chittagong University of Engineering & Technology (CUET), Chittagong - 4349, Bangladesh; Chotan Sheel - Email: chotan_cuetcse03@yahoo.com; *Corresponding author

Received August 26, 2013; Accepted September 01, 2013; Published September 23, 2013

Abstract:
Storage of sequence data is a big concern as the amount of data generated is exponential in nature at several locations. Therefore, there is a need to develop techniques to store data using compression algorithm. Here we describe optimal storage algorithm (OPTSDNA) for storing large amount of DNA sequences of varying length. This paper provides performance analysis of optimal storage algorithm (OPTSDNA) of a distributed bioinformatics computing system for analysis of DNA sequences. OPTSDNA algorithm is used for storing various sizes of DNA sequences into database. DNA sequences of different lengths were stored by using this algorithm. These input DNA sequences are varied in size from very small to very large. Storage size is calculated by this algorithm. Response time is also calculated in this work. The efficiency and performance of the algorithm is high (in size calculation with percentage) when compared with other known with sequential approach.

Keywords: Distributed Bioinformatics System, DNA Sequence, Optimal Storage, Sequential Approach, Performance Measurement.

Background:
Distributed Computing (DC) provides a cost effective framework with efficient execution of a solution on multiple computers connected by a network. For Distributed Computing (DC), large tasks are divided into smaller problems which can then be executed on multiple computers at the same time independent of each other. The task must be broken up into independent problems to minimize inter-computers communication; otherwise distributed computing will not be effective **[1, 2].** Over the past few years, the intermixing of computer science and the complexity of biology has lead to the prosperous field of bioinformatics **[2].** Advances in molecular biology and technology for research have facilitated the process of sequencing of large portions of genomes in various species. Today computers have made medical research more efficient and accurate, by using parallel and distributed computers and complex biological modeling. Bioinformatics, is one of the newer areas, and has opened our eyes to a whole new world of biology **[1].**

The fusion of computers and biology has helped scientists learn more about species, especially humans. With the aid of the computers, we have learned a great deal about genetics, but there still stand many unanswered questions, that are being researched today **[1].** DNA sequence analysis can be a lengthy process ranging from several hours to many days. This paper builds a performance measurement of distributed system using OPTSDNA storing system algorithm on analysis of DNA sequence which provides the solution for many bioinformatics related applications

The overall goal of this paper is to build a performance measurement of optimal storage of Distributed Bioinformatics Computing System for DNA (OPTSDNA) sequence analysis and draw performance curve on storage system and response time. We compared the measurement data of OPTSDNA algorithm with sequential approach data. OPTSDNA algorithm is capable of storing various length of DNA sequence in a Database by compressing the DNA sequence. We observed this algorithm by using a single computer and

 © 2013 Biomedical Informatics

multiple computers. Deferent lengths of DNA sequences are stored in database to compare its response time. For measuring performance we use our previous work algorithm OPTSDNA **[1].**

Different methods had been used to store DNA sequence in Database. To obtaining an image of a mass-storage device **[3]** the sequence of Genome is used Reverse Engineering code. Reverse engineering files on the mass – storage device is equivalent to design and maintenance specification. Obtaining one full human sequence will be technical challenges. Computers will play a crucial role in the entire process, from robotics to control experimental equipment to complex analytical methods for assembling sequence fragments. Indexing for large sequence Database uses the n-gram wavelet transformation **[4]** upon one field and multi-fields index structure under the relational DBMS environment. Results show the need to consider index size and search time while using indexing carefully. Increasing window size decreases the amount of I/O reference and complexity is O (mn).

Indexing and Retrieval for Genomic Database uses CAFÉ indexed scheme **[5, 8]** and it shows that the indexed approached results in significant, saving in computationally intensive local alignment, and that index-based searching is as accurate as existing exhaustive search scheme and it is better than BLAST. Dynamic Programming **[6, 7]** has time and space complexity of O(nm) for two strings S and Q of lengths n and m, for database comparisons it will needs matrix of size n * m. Hence for long sequence and large database this method will be not practical in term of both space and time. Dictionary based indexing **[6]** for a database of sequence Si (i; 1,2,….n), creates index structure of size n corresponding to database size, predefining query lower bound length (L) to be equal to log(n) assumed. Query with larger length will be partitioned into smaller parts. All substrings of length L mapped to integers using hasing function and for queries larger than L split it into sub-queries, then search each sub-query alone and combine the results. This method indexes all possible strings of a pre-specified length L. Dictionary based index size is larger than the database.

The specific objective of this paper for performance analysis of DNA sequences are given below: **(1)** Store various sizes of DNA sequences using OPTSDNA algorithm; **(2)** Implement them on loosely couple distributed network such as regular local area network; **(3)** We use four, five, and six consecutive nucleotides division for storage of DNA sequence data; **(4)** Calculate the storage size for four, five, and six consecutive nucleotide divisions of DNA sequence data in Mega Bytes (MB); **(5)** The performance of storage system is compared with sequential approach; **(6)** Calculate the response time in storage data.

This paper is organized in five sections. Section 2 discusses the methodology of OPTSDNA algorithm. The database design of distributed OPTSDNA algorithm is discussed in section 3. Section 4 discusses the Experimental Results and Conclusion is included in section 5.

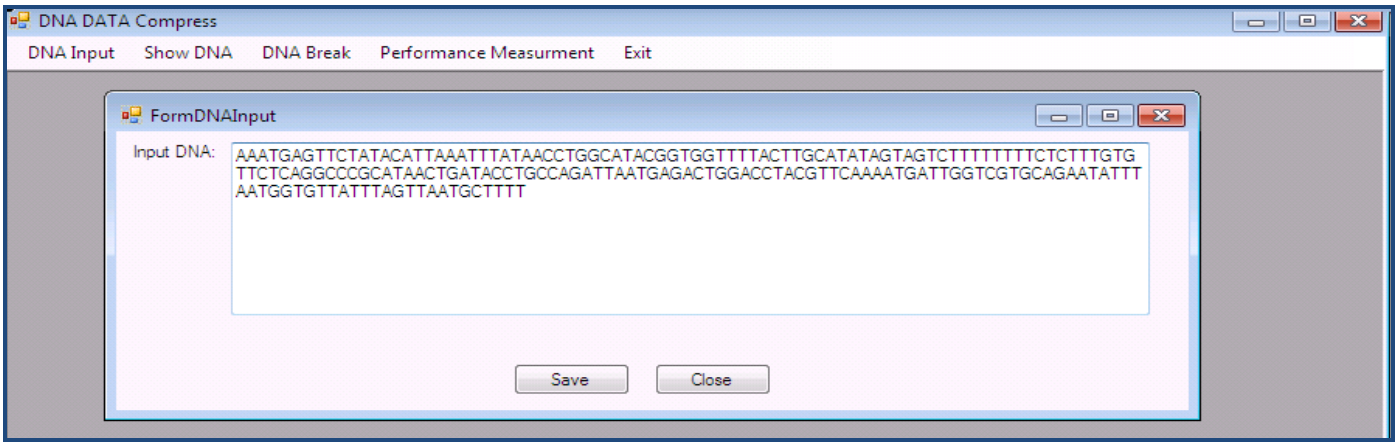


Figure 1: DNA data input.

Methodology:

Implementation of OPTSDNA Algorithm

A Dot Net based client server system was developed for this project **[9].** OPTSDNA algorithm is developed by using Visual Basic Dot Net **[9]** and SQL Server 2008 **[10].** Various interface of OPTSDNA storing system is designed by using Visual Basic Dot Net and we used database as SQL Server 2008. For implementing this algorithm required machine tools are: a) Windows 7, b) SQL Server 2008, c) Visual Studio 2010, d) RAM 2GB etc. In this paper, a client provides the user input from Graphical User Interface (GUI) and then sends this input to one or more server computers as directed by the user. The processing options are shown on **Figure 1, Figure 2, Figure 3a & 3b** GUI. Firstly, we store DNA sequence using Figure 1 GUI process. We show longer DNA sequence process by using **Figure 2** GUI. We also show no. of break DNA sequence process by using **Figure 3a** GUI. Then we measure the performance by using **Figure 3b** GUI.

Database Design of Distributed OPTSDNA Algorithms

The distributed algorithm (OPTSDNA) is based on client server model. For distributed system, the proposed framework avoids duplicate computations on server machines. The input of OPTSDNA algorithm is large no. of DNA sequence. Input is divided into multiple segments such as four, five, and six nucleotide division. Then the segment generated code which is we called encode. Then we store in database only encode of DNA segments.

DNA Segment Encoding

In this part we describe total length of DNA sequence of field's entry in database and how the length of such fields is encoded

© 2013 Biomedical Informatics

and packed into the other segments. We use three tables such as: **Table 1 (see supplementary material)** is 'Original DNA Data', **Table 2 (see supplementary material)** is 'Break DNA' and **Table 3 (see supplementary material)** is 'Coding Storage' for encoding.

Experimental Result

Experimental Setup

OPTSDNA storing system is designed by using Visual Basic Dot Net and we used database as SQL server 2008. For implementing this algorithm required machine tools are: a) Windows 7, b) SQL Server 2008, c) Visual Studio 2010, d) RAM 2GB etc.

Space Requirement

For calculating performance first calculate the storing size of database using OPTSDNA algorithm. Then we compare its capacity size with sequential approach. Calculate capacity size for four, five and six nucleotide division segments. Then we calculate the response time for four, five and six nucleotide division segment process. Store capacity and decrease of size is calculated using the following formula: **(for formulas please see supplementary materials).**

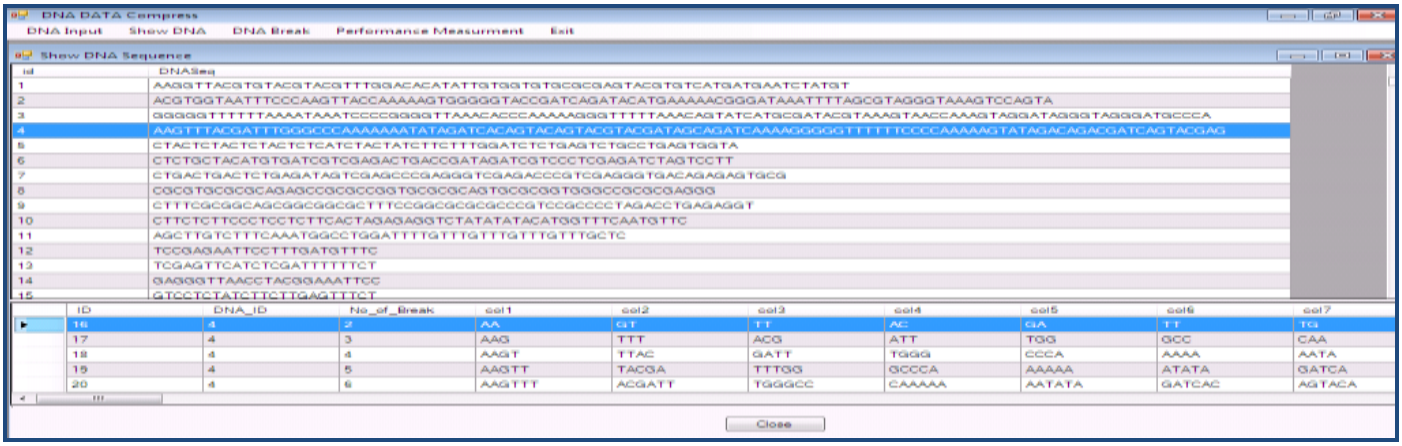


Figure 2: Show DNA sequence with no. of break and code entry.

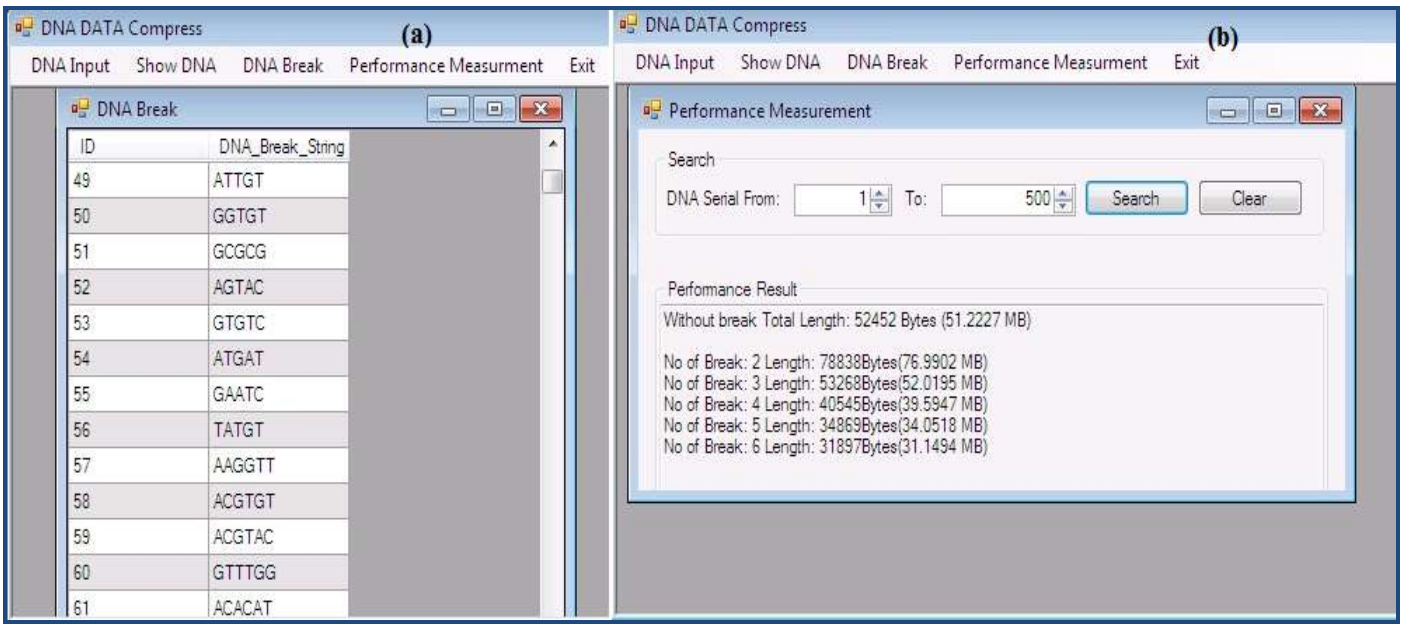


Figure 3: **(a)** DNA data compression with generated break code; **(b)** Performance Measurement of DNA Sequence Entry.

The comparative space requirement between sequential approach and proposed system are given in **Table 4 & 5 (see supplementary material**) From **Table 4** we observe that in sequential approach the required size for 1000 DNA sequence entry in database is 90.06 MB. But in OPTSDNA algorithm required size is less then sequential approach (shown in **Figure 4**). From **Figure 6**, we observe that OPTSDNA algorithm almost 42.21 % (Six Segment Division Nucleotide) size decreases from sequential approach. If we increase our input size then percentage of decrease size is increase and space required is much more than sequential approach. After analysis from **Table 4** and **Figure 5**, our paper proved that OPTSDNA algorithm is much more efficient than sequential approach and others.

We know in compressed entry the required time is higher than the normal entry. Our OPTSDNA algorithm uses encoding technique. In encoding technique storage system response time is higher than the sequential approach. But from **Table 5**, we observe that the needed time in OPTSDA system is not much larger than sequential approach (only 5 ms is lager in Six Segment Division Nucleotide).If we use the speedy processor then we decrease the response time easily. Performance

© 2013 Biomedical Informatics

analysis of comparative response time for storing DNA is shown in **Figure 6**.

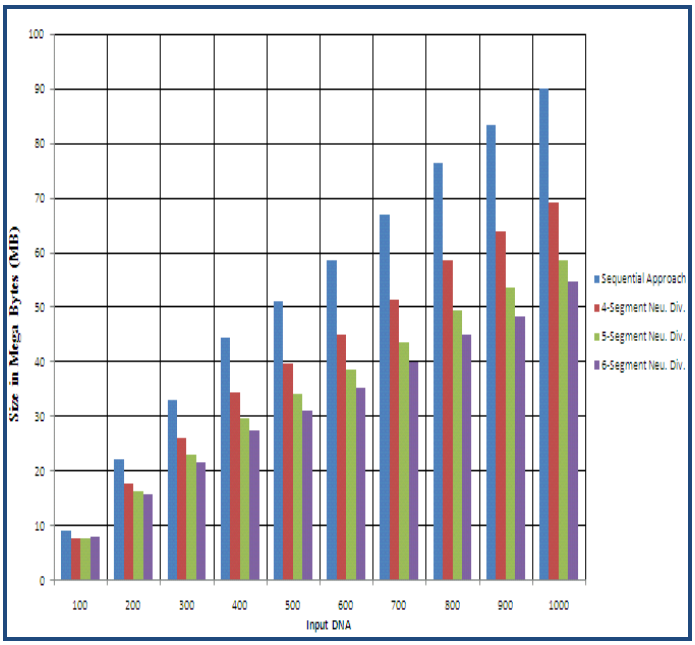
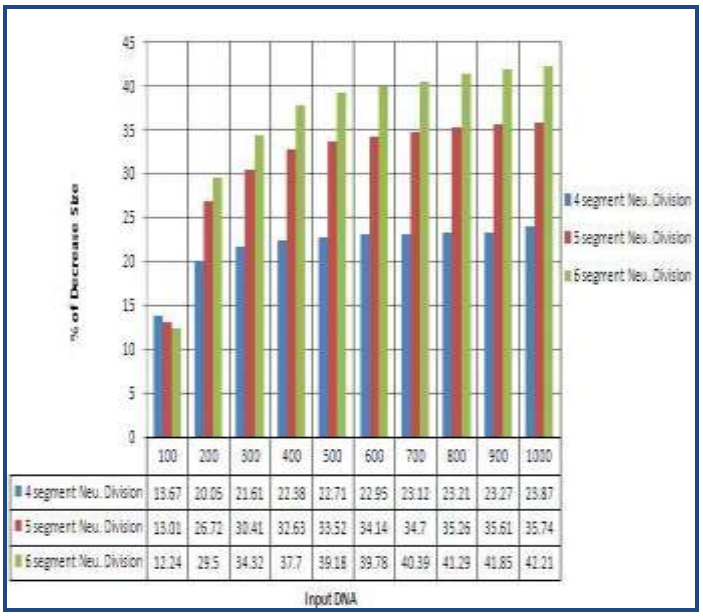


Figure 4: Performance Measurement of OPTSDNA Algorithm



	100	200	300	400	500	600	700	800	900	1000
4 segment Neu. Division	13.67	20.05	21.61	22.38	22.71	22.95	23.12	23.21	23.27	23.87
5 segment Neu. Division	13.01	26.72	30.41	32.63	33.51	34.14	34.7	35.26	35.61	35.74
6 segment Neu. Division	12.14	29.5	34.32	37.7	39.18	39.78	40.39	41.29	41.85	41.21

Figure 5: Performance Analysis of Percentage of Decrease Size

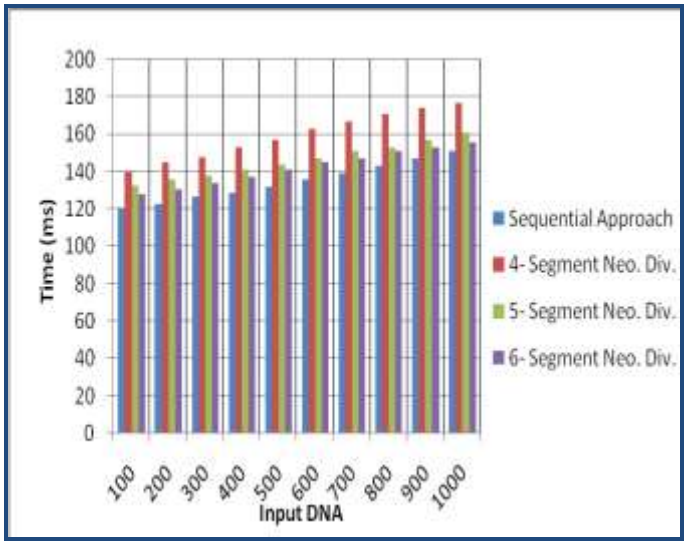


Figure 6: Performance Analysis of Comparative Query Time Requirement for Storing DNA Sequence.

Conclusion:

The algorithm helps to store DNA sequences of varying length in the database. OPTSDNA gives the low CPU cost which is an important factor of query performance. The advantage of OPTSDNA is that the percentage of decrease in size is increased even if the amount of DNA sequence is increased. Data compression saves costs and compression has (almost) no additional CPU overhead in this case. OPTSDNA (1) removes the duplication of DNA data entry in the database; (2) requires fewer bytes than original data to represent in database; (3) save I/O bandwidth and disk size; (4) use multi dictionary based database for DNA sequence entry.

References:

[1] Sheel C *et al. International Journal of Computational Bioinformatics and In Silico Modeling*. 2013 **2**: 106
[2] R Kumar *et al. IEEE Spectrum*. 2007 **7**: 358
[3] Wohoush SM & Saheb MH, *International Journal of Biometrics* (IJBB), 2011 **5**: 267
[4] Robbbins RJ, *Bioinformatics Research*. 1994 **5**: 258
[5] Williams HE & Zobel J, *Nucleic Acid Research*. 2002 **20:** 203
[6] Kahveci T & Singh A, *A Singh Pacific Symposium on Bio-computing*. 2003 **8**: 303
[7] Rashid NAA *et al. IEEE Spectrum*. 2006 **6**: 699
[8] Ozturk O & Ferhatosmanoglu H, Proceeding of the Third IEEE Symposium on Bioinformatics and Bioengineering (BIBE'03), 2003.
[9] Petroutsos E. "Mastering Visual Basic.NET", 2nd Edition, 2006
[10] Mistry R, "Microsoft SQL Server 2008 Management and Administration", McGraw Hill Edition, 2006.

Edited by P Kangueane
Citation: Khan & Sheel, Bioinformation 9(16): 842-846 (2013)
License statement: This is an open-access article, which permits unrestricted use, distribution, and reproduction in any medium, for non-commercial purposes, provided the original author and source are credited

BIOINFORMATION

open access

Supplementary material:

Formulas: Store capacity and decrease of size is calculated using the following formula

Store Capacity	$= m \sum_{i=1}^{t} Dt + m \sum_{j=1}^{t} Ct + n \sum_{k=1}^{s} Ss$ $= m \left(\sum_{i=1}^{t} Dt + \sum_{j=1}^{t} Ct \right) + n \sum_{k=1}^{s} Ss$ .....[Ref. 1]
Size Decrease	$\frac{SequentialApproachSize - N\_SegmentsNucleotideDivisionSize}{SequentialApproachSize} \times 100\%$

Table 1: Attributes and length in Bytes of "Original DNA Data"

Attribute	Type	Length (B)
DNA_SL	Auto Increment	4
DNA	Text	200

Table 2: Attributes and length in Bytes of "Break DNA"

Attribute	Type	Length (B)
Break_DNA	Text	6
Code	Auto Increment	3

Table 3: Attributes and length in Bytes Of "Coding Storage"

Attribute	Type	Length (B)
Input	Number	4
No_Break	Number	1
Col_N	Number	3

Table 4: Comparative Space Requirement

DNA Input	**Size in Mega Bytes (MB)**			
	Sequential Approach	**OPTSDNA Algorithm**		
		4 – Segment Division Nucleotide	**5 – Segment Division Nucleotide**	**6 – Segment Division Nucleotide**
100	9.07	7.83	7.89	7.96
200	22.34	17.86	16.37	15.75
300	33.18	26.01	23.09	21.79
400	44.32	34.4	29.86	27.61
500	51.22	39.59	34.05	31.15
600	58.52	45.09	38.54	35.24
700	66.88	51.42	43.67	39.87
800	76.48	58.73	49.51	44.9
900	83.23	63.86	53.59	48.4
1000	90.06	68.56	57.87	52.05

Table 5: Comparative Query Time Requirement for Storing DNA Sequence

DNA Input	**Size in Mega Bytes (MB)**			
	Sequential Approach	**OPTSDNA Algorithm**		
		4 – Segment Division Nucleotide	**5 – Segment Division Nucleotide**	**6 – Segment Division Nucleotide**
100	9.07	7.83	7.89	7.96
200	22.34	17.86	16.37	15.75
300	33.18	26.01	23.09	21.79
400	44.32	34.4	29.86	27.61
500	51.22	39.59	34.05	31.15
600	58.52	45.09	38.54	35.24
700	66.88	51.42	43.67	39.87
800	76.48	58.73	49.51	44.9
900	83.23	63.86	53.59	48.4
1000	90.06	68.56	57.87	52.05

© 2013 Biomedical Informatics