ABSTRACT
The intrinsic parallelism in bit operations like AND/OR inside a computer word is known as bit parallelism. Since 1992, this bit parallelism is directly used in string matching for matching efficiency improvement. Some of the popular bit parallel string matching algorithms Shift OR, Shift OR with Q-gram, BNDM, TNDM, SBNDM, LBNDM, FBNDM, BNDMq, and Multiple pattern BNDM. This paper discusses the working of various bit parallel string matching algorithms with example. Here we present how bit parallelism is useful for efficiency improvement in various algorithms.

Keywords
String Matching, Bit Parallelism, Shift OR, BNDM, TNDM, SBNDM, LBNDM, FBNDM, BNDMq, SBNDMq, WW Algorithm.

1. INTRODUCTION
Bit parallelism [1] is an intrinsic property of computer in which bit operations are performed parallelly within the computer word in the single clock. With the use of bit parallelism in String Matching algorithm speed of matching is improved up to certain level. String matching [2] algorithms are used in most of the real world applications where pattern extraction is required like as Intrusion Detection system [3][4], Plagiarism detection [5], Data Mining [6] and Bioinformatics [7]. Bit parallel algorithms are faster than the other benchmark character based algorithms like as KMP [4][8], BM [9][10], BMH [11][12], BMHS[13], BMHS2[14], BM[15], Improved BMH[16], Cmmentz Walter[17][18], Wu Manber[19][20] and Aho-Corasick [21][22] etc. Bit Parallel algorithms [23] are based on the non deterministic automata but there is no such automata are present. It is simply the efficient simulation of non deterministic automata. Figure-1 shows how the bit parallel operations are performed inside the computer word. Here computer word size length is 8 bits or 1 byte.

After Shift OR in 1998 Navarro and Raffinot were introduced new bit parallel algorithm named as BNDM (Backward Non Deterministic Matching)[25]. This was an exact single pattern string matching algorithm. In BNDM algorithm we use AND or SHIFT operation which performed in parallel. BNDM is faster than character based algorithms but here pattern size must be smaller or equal than the computer word size.

BNDM set the benchmark in the string matching algorithm. After BNDM in 2003 Peltola and Tarhio introduced improved version of BNDM known as TNDM (Two way Non Deterministic Matching)[26]. In TNDM scanning is much similar to the BNDM except mismatch at last position instead of shifting it scan forward.

In 2003 another algorithm was introduced Simplified BNDM also known as SBNDM [26]. It was an improved version of BNDM. Here, it is not required to find the longest prefix that's why the average length of shift is reduced. Here, the inner most loop becomes simpler.

In 2005 Longtao He, Binxing Fang and Jie Sui proposed a bit parallel algorithm known as Wide Window Algorithm [27]. It use the wide window of size one less than the two time the pattern length to attempt m position in parallel. They combine the bit parallel technique with new wide window concept. It is exact single pattern matching algorithm which is faster in most of the cases.

In 2006 an improved version of Shift OR was introduced by Salmena, Tarhiio and Kyotojik known as Shift OR with Q-gram [28]. Same as Shift OR it is an approximate multiple string matching algorithm. It considers Q character at a time for comparison by doing that the size of automata is reduced and number of comparisons for finding pattern is reduced up to certain level. This algorithm also reduces the false matches.

In 2009 Branislav Durian, Jan Holub, Hannu Peltola and Jorma Tarhiio introduced the concept of Q gram in BNDM algorithm known as BNDMq [29]. It reads the q characters at each alignment before testing the state variable. So most of the cases the number of comparisons are less than in comparison to BNDM required.

Similarly as BNDMq Branislav Durian, Jan Holub, Hannu Peltola and Jorma Tarhiio also introduced SBNDMq [29] in 2009. It is an exact single pattern string matching algorithm. Similar approach of BNDMq was used in SBNDMq which gives the better results in comparison to SBNDM.

In 2010 Changsheng Miao, Guiran Chang and Xingwei combine the concept of Q gram with BNDM and implement exact multiple string matching algorithm known as Filtering based multiple string matching algorithm [30].

Figure 2 shows the evolution of bit parallel algorithm with their description.
Detailed description of the BNDM algorithm. BNDM simulates the BDM algorithm. BDM skips character using suffix automata which is deterministic in preprocessing. To construct deterministic automata is complex task. BNDM simulates the non deterministic version using bit parallelism. BNDM algorithm consist in two phase

**2. BIT PARALLEL ALGORITHM**

**2.1. Shift OR Algorithm**

Shift OR [13] algorithm is an approximate multiple pattern string matching algorithm which means it search number of pattern at a time but there is a possibility of error. In Shift OR algorithm order of searching is from left to right. It is design for large pattern have equal length and equal or less than the word size of a computer.

Most of the multiple pattern algorithms build a trie of the pattern in preprocessing phase so as the pattern size increases size of tree also increase which is not practical to maintain. Shift OR algorithm is a simulation of non deterministic automata where we do not need to build any trie. They don’t need to buffer the input. It is real time algorithm suitable to be implemented in hardware.

Shift OR consist into two phase:

1. Pre-Processing Phase
2. Searching Phase

Pre-processing phase: In pre-processing phase we find out the bit vector of the every character of alphabet. In this ‘ith’ bit is zero if and only if character appears at position ‘i’ otherwise place 1 and write it into reverse order.

Searching Phase: The automation has a transition from state ‘i’ to ‘i+1’ on character c if and only if ‘ith’ bit in B[c] is 0. In state vector D where ‘ith’ bit is 0 if and only if state ‘i’ in the automation is active. If 0 is occurs at MSB means we find the pattern at that position.

**Algorithm Shift OR** (text=t1…tn, patterns=p1,…,pk) [24]

1. Pre-processing:
   
   \[
   \text{[Text[i]]} \leftarrow \text{1m}, \text{set } 1 \\
   \text{for } j=0 \ldots \text{k do} \\
   \text{for } i=0 \ldots \text{m-1 do} \\
   \text{B[ [Text[i]]] } \leftarrow \text{B[ [Text[i]]] } \& \sim (s<\text{l}) \\
   \text{end for} \\
   \text{end for} \\
   \]

2. Searching:

While pos < n do

\[
\text{D } = \text{1m} \\
\text{D } = \text{D } \triangleright \text{1} | \text{B [text [pos + j]]} \\
\text{if } D \triangleright \text{1m-1 do pos} \leftarrow \text{pos + 1} \\
\text{else do} \\
\text{count } \leftarrow \text{count +1} \\
\text{Report occurrence at position pos< pos-m+1} \\
\text{D } \leftarrow \text{1m} \\
\text{pos< pos +1} \
\]

Let’s take an example to understand Shift OR algorithm. Suppose FAST, MACC and BATC be the patterns of length 4

\[
\text{and STRINGFASTMATCH be the text of length 15.} \\
\]

**Figure 2: Evolution of Bit Parallel Algorithm**

This paper gives the detailed description of above bit parallel string matching algorithms. Paper starts from very first algorithm based on bit parallelism up to latest one.

**2.2. BNDM Algorithm**

BNDM stands for Backward Non Deterministic Matching [25]. It is exact single pattern string matching algorithm. In BNDM order of searching is from right to left. It uses the concept of bit parallelism from shift OR algorithm [24] and suffix automata from BDM algorithm [25]. This algorithm is a bit parallel simulation of BDM algorithm. BDM skips character using suffix automata which is deterministic in pre processing. To construct Deterministic automata is complex task. BNDM simulates the non deterministic version using bit parallelism.

**BNDM algorithm consist in two phase**

1. Pre-processing phase
2. Searching phase

Pre-processing: In pre-processing phase we find Bit Vector of each Character of the Pattern calculated by putting 1 for occurrence and 0 for non-occurrence and take bit vector D with initial value with all one.

Searching Phase: In searching phase pattern is searched with the help of two logical operator that are AND and SHIFT. Pattern is found when MSB of D is 1 and value of j is 0.

**Table**: Various steps of Shift OR Algorithm

<table>
<thead>
<tr>
<th>TEXT</th>
<th>D</th>
<th>B[T[i]]</th>
<th>Pos</th>
<th>D'</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRINGFASTMATCH</td>
<td>1111</td>
<td>-</td>
<td>-</td>
<td>D'</td>
<td>D all ones</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1110</td>
<td>1011</td>
<td>1</td>
<td>1111</td>
<td>D &amp; B[S]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1110</td>
<td>0011</td>
<td>2</td>
<td>1111</td>
<td>D &amp; B[T]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1110</td>
<td>1111</td>
<td>3</td>
<td>1111</td>
<td>D &amp; B[R]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1110</td>
<td>1111</td>
<td>4</td>
<td>1111</td>
<td>D &amp; B[I]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1110</td>
<td>1111</td>
<td>5</td>
<td>1111</td>
<td>D &amp; B[N]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1110</td>
<td>1111</td>
<td>6</td>
<td>1111</td>
<td>D &amp; B[G]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1110</td>
<td>1110</td>
<td>7</td>
<td>1110</td>
<td>D &amp; B[F]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1100</td>
<td>1101</td>
<td>8</td>
<td>1101</td>
<td>D &amp; B[A]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1100</td>
<td>1011</td>
<td>9</td>
<td>1101</td>
<td>D &amp; B[S]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>0110</td>
<td>0111</td>
<td>10</td>
<td>1011</td>
<td>MSB = 0, Pattern found</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1110</td>
<td>1110</td>
<td>11</td>
<td>1110</td>
<td>D &amp; B[M]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1100</td>
<td>1101</td>
<td>12</td>
<td>1101</td>
<td>D &amp; B[A]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1101</td>
<td>0011</td>
<td>13</td>
<td>1101</td>
<td>D &amp; B[I]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>0110</td>
<td>0011</td>
<td>14</td>
<td>1011</td>
<td>MSB = 0, Pattern found</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>1110</td>
<td>1111</td>
<td>15</td>
<td>1111</td>
<td>D &amp; B[H]</td>
</tr>
</tbody>
</table>

*Pattern are found at position 7 & 11

**Figure 3: Various steps of Shift OR Algorithm**

Figure 3 shows the various step involved in the shift OR algorithm. Initially we set the value of D = 1111 (All one) and update D when a character c is read from the text as follow D = (D<<1) | B[c]. Whole searching is carry out from left to right one by one if 0 is occurred at MSB it means pattern is found.

In Shift OR algorithm pre-processing and search are very simple and only bitwise logical operations Shift and AND are used and no Buffering is required. It is a real time algorithm. Time delay to process one text character is bounded by a constant depend only on pattern length. Here all pattern length must be equal.
For $c \in \Sigma$ do $B[c] \leftarrow 0m$
For $i \in 1...m$ do $B[pm-i+1] \leftarrow B[pm-i+1] | 0m-110i-1$
Searching
Pos $\leftarrow 0$
While (pos<=n-m) do
$J \leftarrow m$, last $\leftarrow m$
$D \leftarrow 1m$
While $d! = 0m$ do
$D \leftarrow D & B[Tpos+j]$
$J \leftarrow j-1$
If $D & 10m-1! = 0m$ then
If $j>0$ then last $\leftarrow j$
Else report an occurrence at pos+1
$D \leftarrow D<<1$
End of While
Pos $\leftarrow pos+last$
End of while
Let us understand the working of whole algorithm with the help of an example. Let Text $T$ = ‘STRINGFASTMATCH’ and Pattern $P$= ‘FAST’

Initially $J=4$, last=4, pos. = 0 and bit vector $D = 1111$

Than we perform the BNDM algorithm. There various step are shown in the Figure 4 with proper explanation.

Figure 4: Shows the various steps perform by BNDM

BNDM algorithm become very fast string matching algorithm except very sort (0-6) or very long (90-150) pattern. It is faster than the previous algorithm Shift OR, BDM and Occupies very less space perform various operation in parallel. It is very Simple and flexible algorithm. But we have all pattern assume to be less than or equal to the word size of computer.

2.3. TNDM Algorithm

TNDM stands for Two ways Non Deterministic Matching [26]. It is Exact Single pattern string matching algorithm. It is almost same as the BNDM algorithm [25] there is slight changes in the case of mismatch occur at first position instead of shifting TNDM look forward to find suffix of reverse pattern. The number of examined characters is less than BNDM therefore matching is faster.

The simulation of TNDM Algorithm can be understood by the help of the Figure 5 here mismatch is occurred at last so it scan forward to find the maximum suffix of pattern after shifting to maximum suffix the algorithm is run same as BNDM algorithm[25].

Figure 5: Working explanation of TNDM algorithm

2.4. SBNDM Algorithm

SBNDM [26] stands for Simple Backward Non Deterministic Matching. SBNDM is much similar to the BNDM [25] algorithm there is a slight change in term of shifting. Due to this it is quit faster than the BNDM algorithm. Here we do not require finding the longest prefix. Let T is the text of length n and P is the pattern of length m to be searched. At each alignment window of P in T, Scan T from right to left until the suffix of the window is not a factor of P or an occurrence of P is found. Shifting of SBNDM is done according to these i.e. shift window by m if no factor is found, shift by 1 if P found otherwise next alignment is start at last factor. Figure 6 describes the various steps of the SBNDM algorithm.

Figure 6: Example of SBNDM

By using the SBNDM concept the average length of shift is reduced by doing so the innermost loop of algorithm become simpler. It is faster than the BNDM algorithm.

2.5. Wide Window String Matching Algorithm

Longtiao He, Binxing Fang and Jie Sui proposed an algorithm known as WW (wide window) Algorithm [27] in which text is divided into $n/m$ where $n$ is the length of text and $m$ is the length of pattern. It opens a window with size $2m-1$. in the window the algorithm attempts $m$ possible occurrence position in parallel. In this window is dividing it into two parts. First one is denote as A1 of size $p-1$ and second part as A2 of size $p$. With the help of the Figure 7 we can understand how pattern is divided into window.
Now in this algorithm we have to find out the all prefixes of A2 which is also the suffixes of the pattern m and shift the window directly if it fails. Let r denotes the length of longest prefix. Than we find the suffix of A1 which is also prefix of pattern m equal to the length of m-r if such cases are found means we got the pattern otherwise not.

Figure 7: Shows how Pattern is divided into Window

Now in this algorithm we have to find out the all prefixes of A2 which is also the suffixes of the pattern m and shift the window directly if it fails. Let r denotes the length of longest prefix. Than we find the suffix of A1 which is also prefix of pattern m equal to the length of m-r if such cases are found means we got the pattern otherwise not.

Figure 8: Demonstrate process of searching in WW Algorithm

Figure 8 shows the process of WW algorithm here we have a pattern ‘GFASTM’ of size 6 so window size is 2m-1 that is 11. Now longest suffix of pattern in A2 is ‘ASTM’ from A1 we got ‘GF’ it means pattern is found. WW is efficient for searching short pattern. It suits for off line pattern matching as well as high speed online pattern matching.

2.6. Shift OR with Q-Gram

Shift OR with Q-gram [28] is an enhanced version of the Shift OR algorithm. In Shift OR with Q-gram algorithm we take Q character at a time for comparison by doing so the size of automata is reduced and the number of comparison for finding pattern is reduced up to certain level. The Q-gram can be of two types Consecutive Q-gram or Overlapped Q-gram. In Consecutive Q-gram we read pattern in a sequence of Q character at a time while in Overlapped Q-gram we take Q character from each character of the patterns. Here the pattern length of each pattern must be same. Shift OR with Q-gram carried out in three phases First phase: Initialization Phase where initialization of the variable is carried out. Second phase: Pre-processing phase where bit vector of the various Q gram are taken place. Third phase: Searching phase where searching of the pattern in the text is carried out. Let us take an example of Shift OR Consecutive 2-QGram where Text is STRINGFASTMATCH and Patterns are ‘GFASTM’, ‘ABATCH’ and ‘TMACCT’ hence the consecutive 2-gram of patterns are “GF”, “AS”, “TM”, “AC”, “CT”, “AB”, “AT” and “CH” by doing so the number of bit in bit vector is reduced to the half of the pattern length. Here 2-gram of the pattern is treated as single character. The ith bit of the bit vector is set to zero if there is an occurrence of 2-gram in the ith position of the pattern otherwise set to one. So bit vector of our pattern is as follow.

\[
\begin{align*}
\end{align*}
\]

Matching the text: STRINGFASTMATCH

Initialise \( D = 111 \) and Update \( D \) when a character \( c \) is read from the text by \( D = (D << 1) \) \( B\{c\} \).

Various steps involved are discussed in the Figure 9 here patterns are searched similar to Shift OR by taking q character at a time. Searching is start from left of the text taking ST first we got all one means mismatch so shift the window and take the next q gram TR and same process is repeated till the text end.

<table>
<thead>
<tr>
<th>TEXT</th>
<th>D</th>
<th>B[T]</th>
<th>D'</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRINGFASTMATCH</td>
<td>111</td>
<td>-</td>
<td>-</td>
<td>Initial D all one</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>110</td>
<td>111</td>
<td>111</td>
<td>D OR B[T]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>110</td>
<td>111</td>
<td>111</td>
<td>All one means no match</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>110</td>
<td>111</td>
<td>111</td>
<td>Shift one position right</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>110</td>
<td>111</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>110</td>
<td>110</td>
<td>110</td>
<td>Partial match found</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>100</td>
<td>101</td>
<td>101</td>
<td>D OR B[AS]</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>100</td>
<td>011</td>
<td>011</td>
<td>MSB 0 means pattern found</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>100</td>
<td>011</td>
<td>011</td>
<td></td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>100</td>
<td>011</td>
<td>011</td>
<td>MSB 0 means pattern found</td>
</tr>
</tbody>
</table>

MSB 0 Means pattern found. Hence pattern found at 6 & 10

Figure 9: Shows the various steps of Shift OR with Q Gram

In the example two patterns is found at 6 & 10 but second one is not in the patterns so it is a false match. By taking the q gram the time efficiency of the algorithm is improved but other factor is unchanged it is still not working for pattern larger than the word size and unequal pattern length. Here the number of false matches is reduced up to the certain level.

2.7. BNDM with Q-gram

BNDM with Q-gram [29] is an improved variation of the BNDM algorithm which reads the q gram at each alignment before testing the state variable. In this algorithm loop has been made as sort as possible in order to quickly advance m+q-1 position. Here q can be varies according to our requirement. The whole algorithm can be easily understood with the help of the example given below.

Text \( T = \) ‘STRINGFASTMATCH’ and Pattern \( P = \) ‘FAST’ Assume we have 2-gram

Pre-processing: In pre-processing phase we find Bit Vector of each Character of the Pattern calculated by putting 1 for occurrence and 0 for non-occurrence.

\[
\begin{align*}
\end{align*}
\]

Searching Phase: Initially we take variable \( i \) and set to m+q-1 where m is pattern length

Than we perform the BNDMq algorithm whose various step are described in the Figure 10 where we don’t enter the loop until the q gram is matched.

<table>
<thead>
<tr>
<th>Text</th>
<th>i</th>
<th>F(i/q) / D</th>
<th>j</th>
<th>First</th>
<th>D'</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRINGFASTMATCH</td>
<td>3</td>
<td>0000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>D= B[r] &amp; (B[i]&lt;1)</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>6</td>
<td>0000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>D all zero shift window</td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>9</td>
<td>0010</td>
<td>8 &amp; 6</td>
<td>0100</td>
<td>D not zero enter in loop</td>
<td></td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>9</td>
<td>0100</td>
<td>7 &amp; 6</td>
<td>1000</td>
<td>D &amp; B[AS]</td>
<td></td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>9</td>
<td>1000</td>
<td>6 &amp; 6</td>
<td>-</td>
<td>MSB=1, pattern found</td>
<td></td>
</tr>
<tr>
<td>STRINGFASTMATCH</td>
<td>12</td>
<td>0000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Figure 10: Step by step Description of BNDMq algorithm

It is become very fast string matching algorithm in the case of no match is found. In this case it does not enter the main loop and directly shift the window. It occupies same space as take in BNDM and word size limitation is still exists.
2.8. Multiple Pattern BNDM

BNDM algorithm is a single pattern string matching algorithm which is very fast one because it uses the concept of bit parallelism. Changsheng Miao, Guiran Chang and Xingwei Wang convert the BNDM algorithm in multiple pattern BNDM algorithms [30]. They develop Filtering Based Multiple String Matching Algorithm by Combining q-Grams and BNDM.

Let’s understand the concept of algorithm by considering an example Text: STRINGFASTMATCH, Text size: 15, Patterns: FAST, MACC and BATC pattern length (m): 4

In preprocessing phase bit vector of the various character are calculated which is shown in the Figure 11 which are calculated in similar fashion as calculated in BNDM algorithm.

![Figure 11: Shows calculation of bit vector](image)

Initialize q = 2, i = m-q+1. The whole searching process is described in the Figure 12 step by step.

![Figure 12: Shows the various steps perform by Multiple BNDM](image)

In multiple pattern BNDM string matching algorithm multiple patterns can be searched which were not possible in BNDM Algorithm. We need filtering after pattern match to find the exact match which will take extra time to search.

3. CONCLUSION

Bit parallel string matching algorithms are the new series of efficient algorithm. Each algorithm is efficient in compare to standard character based algorithm. BNDM, TNDM, SBNDM, WW, BNDMq are the bit parallel single pattern string matching algorithm whereas Shift OR, Shift OR with Q-gram, multiple pattern BNDM are used for multiple pattern matching. Multiple pattern matching is difficult in compare to single pattern matching for multiple matching no efficient exact string matching is possible using bit parallelism. These algorithms have some disadvantages. Major disadvantage is these algorithms is working on patterns of length less than or equal to computer word. These algorithms were working on the pattern of equal length.

4. FUTURE WORK

Most of the bit parallel algorithm is single pattern string matching algorithm so using this concept multiple patterns string matching algorithm can be implemented. Word size limitation present in the bit parallel algorithm can also be avoided.

5. REFERENCES


[13] JingboYuan, Jinsong Yang and Shunli Ding,” An Improved Pattern Matching Algorithm Based on BMHS”, In the proc.


[29] Branislav Durian, Jan Holub, Hannu Peltola and Jarma Tarhio,”Tuning BNDM with q-grams”, In the proc. Of workshop on algorithm engineering and experiments, SIAM USA, pp. 29-37, 2009.