Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A.A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

A SORT-BASED INTEREST MATCHING ALGORITHM WITH TWO EXCLUSIVE JUDGING CONDITIONS FOR REGION OVERLAP

Tianlin Li
Wenjie Tang
Yiping Yao
Feng Zhu

College of System Engineering
National University of Defense Technology
Deya Road, 109
Changsha, 410073, CHINA

ABSTRACT

As the IEEE standard for distributed simulation, High-Level Architecture (HLA) defines the data distribution management (DDM) service to filter unnecessary communication among federates. In DDM, interest matching algorithm is essential for the message filtering. Among existing interest matching algorithms, sort-based algorithms have been proven to be the most efficient method in most scenarios. However, existing sort-based algorithms still have some drawbacks, the overhead of sorting the bounds can be further reduced and a portion of unnecessary bit or matching operations can be eliminated. In this paper, we propose a binary search enhanced sort-based interest matching algorithm (BSSIM). Based on two exclusive judging conditions for region overlap, sorting overhead can be remarkably reduced. Moreover, unnecessary bit or matching operations can be eliminated by binary searches. Experimental results show that BSSIM algorithm outperforms existing sort-based algorithms, and considerable performance improvement can be achieved.

1 INTRODUCTION

In distributed simulation, federates exchange information based on the publish/subscribe method. The method provides significant flexibility than explicitly sending messages, but it would also generate large amount of irrelevant data transmissions. The High Level Architecture (HLA) is a general purpose architecture for simulation reuse and interoperability (Yu et al. 2002). HLA provides data distribution management (DDM) services to filter unnecessary data transmissions and therefore reduce communication overhead among federates (IEEE Standard 1516 2000; Morse and Petty 2001). Producers of data utilized DDM services to assert properties of their data (update region), while consumers of data may utilize DDM services to specify their data requirements (subscription region) (Morse and Steinman 1997). Then, the Run Time Infrastructure (RTI) distributes data from producers to consumers based on interest matching between these regions (Van Hook and Calvin 1998). Hence, interest matching plays a key role in data distribution management (Zhang 2000; Tan et al. 2000b). Without interest matching mechanism, lots of unnecessary data distributions will be generated which will seriously degrade the simulation runtime performance. Even though, without high efficient interest matching algorithm, the performance of simulation will also be seriously affected.

Currently, interest matching algorithms mainly include brute force algorithm (also called region-based algorithm), grid-based algorithm, hybrid algorithm, continuous algorithm and sort-based algorithm. They all have their own strong points and weak points. Among them, the sort-based algorithm is the most promising and efficient. However, the existing sort-based algorithms are still confronted with several major drawbacks. In such sort-based algorithms, there are two typical algorithms, they are proposed by Raczy et al and Pan et al (Raczy et al. 2005; Pan et al 2007). In this article, we call them Raczy's sort-based algorithm and Pan's algorithm. For simplicity, the number of update regions and the number of subscription regions are both assumed to be N. In Raczy's sort-based algorithm, the length of the bound list to be sorted is 4*N, so sorting the bound list will occupy majority of overhead. Besides frequent bit operations will generate a lot of additional overhead during generating overlapping information. Similarly, in Pan's sort-based algorithm, the sorting overhead still accounts for a large portion, because there are 4 bound lists whose length is N to sort. What's worse, in the process of calculating which subscription regions intersect with an update region, the two sufficient and necessary conditions must be satisfied concurrently, meaning that additional matching computation and inaccurate matching results filtering operation should be introduced for exact matching results.

In this paper, we propose a binary search enhanced sort-based interest matching algorithm, aiming at reducing the sorting overhead and avoiding unnecessary bit operation. Based on a novel sufficient and necessary condition to judge interval overlapping, the size of the list to be sorted can be remarkably reduced compared with Raczy's and Pan's proposed sort-based algorithm. Moreover, by means of binary searches, additional bit operation can be eliminated. Comparing to the sort-based algorithm proposed, the sufficient and necessary conditions only need two steps to obtain the final overlapping information for each dimension, and the most important is that each step can obtain exact overlapping result, with no additional operations. Hence there is no need to calculate the intersection to filter the wrong matching information, reducing considerable computation complexity compared with Pan's sort-based algorithm.

The rest of the paper is organized as follows. Section 2 introduces several existing interest matching algorithms. Section 3 describes our proposed interest matching algorithm in two aspects, static matching and dynamic matching. Section 4 describes the experiments and analysis of the results. Section 5 concludes the whole paper and overviews the future work.

2 RELATED WORKS

This section mainly introduces existing interest matching algorithms, brute force, grid-based, hybrid and sort-based algorithm, and analyzes their advantages and disadvantages.

2.1 Brute force algorithm

Brute force, also known as region based, compares update region with subscription region one by one to obtain the overlap information. This approach can achieve exact matching results, while the computation complexity is $O(N^2)$.

2.2 Grid-based algorithm

In Grid-based algorithm, the interest space is divided into grids, so all update regions and subscription regions are mapped to one or several grids (Tan et al. 2000a; Berrached et al. 1998). This algorithm considers the regions in the same grid intersect with each other. This algorithm is high efficient, while the matching result is not exact.

2.3 Hybrid algorithm

Hybrid algorithm is the combination of brute force and grid-based algorithm (Tan et al. 2000c). On the basis of grid-based algorithm, brute force approach is applied for each grid to obtain the exact matching results. When the number of grids is G, the cost of the algorithm is $O(N^2/G)$. However, G value is difficult to choose in real simulation, because the number of regions, the distribution of update and subscription regions and the size of interest space are uncertain and complicated.

2.4 Sort-based algorithm

Sort-based algorithm proposed by Yu Jun et al uses the idea of sorting bounds of regions to obtain overlap information (Yu et al. 2002). For each dimension, all the upper and lower bounds of update and subscription regions are put into list L, so the length of L is 4*N, assuming the number of update and subscribe region both are N. Next, L is sorted in ascending order. When N is large, the sorting overhead cannot be ignored. Then traverse L and remove a bound value from the queue each time. When the value is the lower bound of update region Ui, then put the id of Ui into set UpdateSet. If the value is the lower bound of subscription region Si, then put the id of Si into SubscribeSet. When the value is the upper bound of update region Ui, then remove the id from UpdateSet, and Ui intersect with all the subscription regions in SubscribeSet. It is same for the upper bounds of subscription regions. In this way, overlap information can be obtained. Bit array operation is introduced to accelerate the insertion and removal of elements for UpdateSet and SubscribeSet.

Raczy et al also proposed a sort-based matching algorithm, which is similar to the sort-based algorithm described above (Raczy et al. 2005; Raczy et al 2002). The difference is that there exists an N*N bit matrix M used to maintain and store the overlap information. Through operating the sorted queue and changing the values in M, the final overlapping information can be obtained. In Raczy's sort-based algorithm, sorting the bounds of regions and large number of bit operations will generate a lot of additional overhead, which means that there is still some potential to improve the performance of sort-based interest matching algorithm. When there is no overlapping or the number of overlapping is less, sorting the bounds occupies large majority of matching time (Yu et al. 2002). Through shortening the length of bound list can obviously reduce the sorting time. Besides, bit operation is introduced to acquire the overlapping relationship when operating the sorted bound list in each dimension. Frequent bit operations will generate large amount of overhead. In worst case, when every subscription region intersect with every update region, the bit operation will result in additional $O(N^2)$ computation complexity.

The above two sort-based algorithms all have $O(N^2)$ storage requirement, which will be limited when N is very large. To reduce the storage overhead, Ke Pan et al also proposed an efficient sort-based algorithm to reduce the storage overhead (Pan et al. 2007). In Pan's sort-based algorithm. They introduced two parameters, maximum update region range size (maxURRS) and maximum subscription region range size (maxSRRS), to construct new sufficient and necessary conditions to judge region overlap. The computation complexity is $O(maxRS/D_{UB} * N^2)$. maxRS is the maximum range size of all regions, and D_{UB} is the size of the routing space. Thus the performance of this algorithm depends on $maxRS/D_{UB}$. When the ratio is small, the algorithm is more efficient. However, when the ratio becomes larger, the performance may become worse. This condition limit the efficiency and generality of the algorithm, which is only suitable for large spatial space. Besides, Pan's sort-based algorithm should sort four bound lists and the sorting overhead will be O(4NlogN) for static matching. Besides, in the process of calculating which subscription regions intersect with an update region, the two sufficient and necessary conditions must be satisfied concurrently, meaning that there are three steps to determine the final overlapping information for each dimension. The first step is to choose the subscription regions satisfying the first condition, and the second step is to choose subscription regions according to the second condition, then calculate the intersection to obtain the exact results, meanwhile some additional matching computation is introduced. Binary search is scheduled when mapping the upper bound or lower bound of update regions onto the lower bound list and upper bound list of subscription regions, while in our proposed algorithm, binary search is directly used to obtain overlapping information.

3 BSSIM ALGORITHM

This section will describe BSSIM algorithm in detail, and provide theoretical analysis of computation complexity. For simplicity, we only focus on how our proposed algorithm works in one dimension.

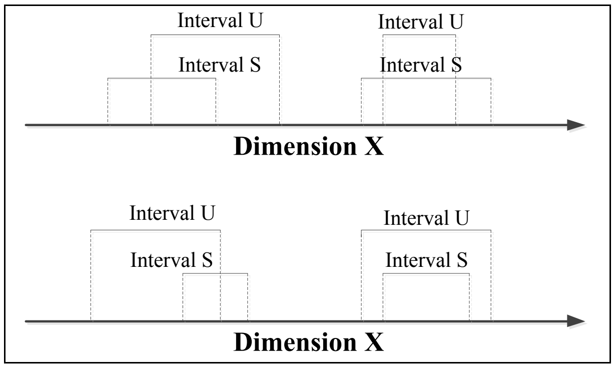


Figure 1: Total overlapping cases between two regions.

3.1 Two Exclusive Judging Conditions for Region Overlap

The common way to judge the overlap relationship of two regions is to compare their upper and lower bounds. For example, there are two regions, update region U and subscription region S. The range of region U is [U_{LB}, U_{UB}), and S is [S_{LB}, S_{UB}). As shown in Figure 1, there are total four cases when U intersects with S. In fact, they can be further categorized into two cases, one is that the lower bound of U locates in S, the other is that the lower bound of S locates in U. According to this observation, we provide two exclusive judging conditions to determine region overlapping relationship. The two conditions are independent, each one of them can be used for judging the overlapping information. Hence, two regions will intersect with each other when one and only one of the two exclusive conditions is satisfied.

$$U_{LB} \in [S_{LB}, S_{UB}). \tag{1}$$

or

$$S_{LB} \in [U_{LB}, U_{UB}). \tag{2}$$

According to above conditions, to obtain the overlapping information in one dimension, the core work is to determine the lower bounds of which update regions locate in each subscription region and lower bounds of which subscription regions locate in each update region. Hence, calculating overlapping information between regions is translated into calculating the overlapping information between all update (subscription) lower bound values with each subscription (update) intervals. Sorting the lower bounds can efficiently reduce the workload of determining which lower bounds of subscription (update) regions locates in each update (subscription) region. These two conditions are more suitable for sort-based algorithm. Firstly, only two lower bound lists need to be sorted, update region lower bounds list and subscription region lower bounds, so the sorting overhead is reduced. Besides, this method need less comparison operation compared with the algorithm proposed by Ke Pan et al.

In summary, the judging conditions for region overlap built the foundation of the BSSIM algorithm. Comparing to the sufficient and necessary conditions proposed be Ke Pan et al, our proposed necessary and sufficient conditions have more advantages. Binary search can be used to obtain overlap information directly. Besides the sorting overhead is reduced significantly. It also can avoid bit operations in Raczy's sort-based algorithm and unnecessary matching operations in Pan's sort-based algorithm.

3.2 Algorithm Description

Based on the two conditions for calculating overlapping information, the core work of BSSIM algorithm is to acquire the overlapping information between all lower bounds of update (subscription) regions and subscription (update) regions for each dimension. That is to say, for each update region we should determine the lower bounds of which subscription regions locate in this update region, satisfying condition (2). And for each subscription region we should determine which lower bounds of update regions locate in this subscription region, satisfying condition (1). For each dimension, when all the lower bounds of update (subscription) regions have been compared with all the subscription (update) regions, the overlapping information for this dimension can be obtained. Through calculating the intersection of the overlapping results of all dimensions, we can get the final matching information.

Before calculating the overlapping information, some preparation work is necessary. First, sorting lower bounds of regions can accelerate the efficiency of acquisition on overlapping information. Hence, all lower bounds of update regions and subscription regions are sorted independently in ascending order. Then we can obtain 4 bound lists, as shown in Figure 2, they are SL, SU, UL, UU. SL stores all the lower bounds of subscription regions in ascending order, and SU list stores the corresponding upper bounds. Similarly, UL and UU are used to store the bounds of all update regions. Besides, two auxiliary lists are used to store corresponding update region id and subscription id respectively.

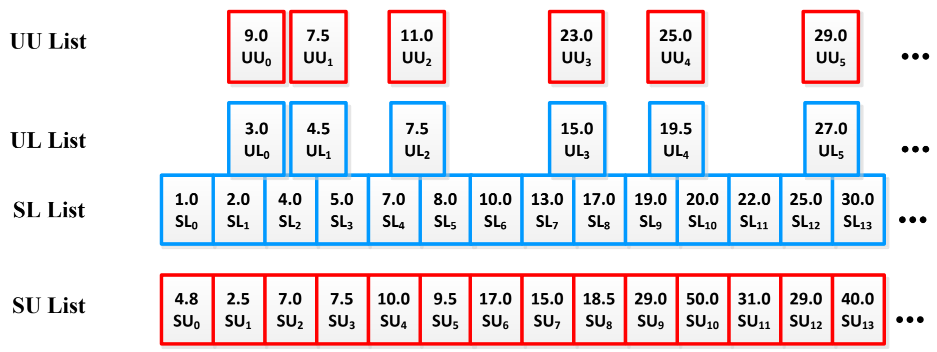


Figure 2: Four lists, SL, SU, UL, and UU.

Based on the judging conditions, binary search method is introduce to obtain overlap results quickly. Given that the lower bound list have been sorted, binary search becomes the best choice for determine which lower bounds of one update (subscribe) locate in one subscribe (update) region. Here we take an update region U as an example. When calculating which subscription region lower bounds locate in U, we should know the position when mapping the lower bound and upper bound of the update region onto the bound list SL. Here two parameters, comparison index upper limit (CIU) and comparison index lower limit (CIL), are introduced to represent the position. The core work of our proposed algorithm is translated into calculating CIL and CIU values. As shown in Figure 3, CIL and CIU of region $[UL_0, UU_0)$ are 2 and 5, similarly, CIL and CIU of interval $[SL_0, SU_0)$ are 0 and 1. For a update region U, CIL denotes the minimum index of the lower bounds that overlap with U, while CIU denotes the maximum index of the lower bounds that overlap with U. Therefore, region $[SL_2, SU_2)$, $[SL_3, SU_3)$, $[SL_4, SU_4)$ and $[SL_5, SU_5)$ overlap with $[UL_0, UU_0)$, while $[UL_0, UU_0)$ and $[UL_1, UU_1)$ overlap with $[SL_0, SU_0)$. When all CIL and CIU values for all update and subscription regions are obtained, the overlap information for each dimension can be obtained.

Hence, the second step is to calculate CIL and CIU values for each update and subscription region for each dimension. In view of list SL and UL have been sorted, CIL values can be easily obtained by comparing values in SL with values in UL. When calculating CIU value for a region, if comparing the region's upper bound with values in SL or UL one by one, calculating CIU values for N regions will generate $O(N^2)$ computational complexity in the worst case. Therefore, we introduce the binary search method to find CIU values efficiently. Binary search method can help locating CIU value quickly with O(lgN) overhead. Considering that the CIL value for a region has been determined, so the searching range for binary search is reduced to [CIL, N]. Here, when CIL value are becoming larger, the search range [CIL, N] is becoming smaller and the cost of calculating CIU becomes smaller, less than O(logN). Then the computational complexity of calculating a CIU value is O(logN) in the worst case.

The last step of our proposed algorithm is to generate the overlapping information. For each dimension, there is a M*N bit matrix used to store the overlapping information. M stands for the number of update regions and N is the number of subscription regions. Each update region corresponds to one row, and each subscription region corresponds to one column. Each value represents an overlapping relation between a update and subscription region. All values of the matrix are initialized as 0. When a update region overlaps with one subscription region, the corresponding bit is set as 1, used to store the overlapping information. When the CIL and CIU values for each update and subscribe region have been determined, so the values in the bit matrix can be assigned as 1 according to CIL and CIU values. Therefore the bit matrix can contain the overlapping information for one dimension. When all dimensions are finished, Calculating the intersection of all bit matrices can obtain the final overlap information.

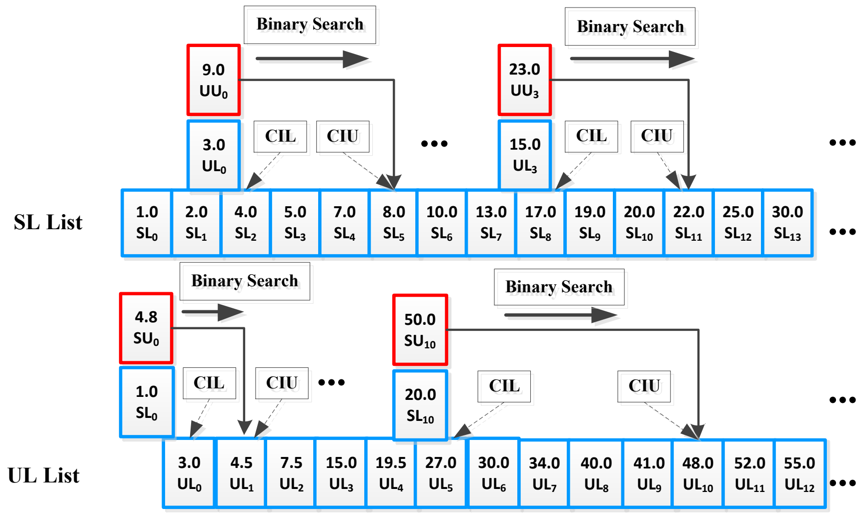


Figure 3: Calculating CIU values with binary search.

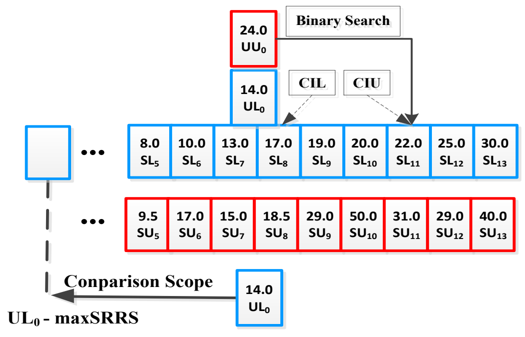
In summary, the procedure of BSSIM algorithm for static matching can be summarized as three phases. As shown in Figure 4, the first phase is sorting lower bounds of all update and subscription regions independently (step 3-5). The second phase is to calculate CIL and CIU values for each update and subscription region (step 6-11). Last phase is the assignment of bit matrices to generate overlap information based on CIL and CIU values (step 12-14).

```
BSSIM(Update, Subscription, M, N){
    1.  for each dimension d
    2.      initialization of bit matrix Matrix[d], all element is 0;
    3.      insert lower bounds of Update regions into UL[d];
    4.      insert lower bounds of Subscription regions into SL[d];
    5.      quicksort UL[d], SL[d];
    6.      initialize UpdateCIL[d][M] by comparing UL[d] and SL[d];
    7.      initialize SubscribeCIL[d][N] by comparing SL[d] and UL[d];
    8.      for each update region u
    9.          binary search in SL[d] to initialize UpdateCIU[d][u];
    10.     for each subscription region s
    11.         binary search in UL[d] to initialize SubscribeCIU [d][s];
    12.     update bit matrix Matrix[d] according to UpdateCIL[d] and UpdateCIU[d];
    13.     update bit matrix Matrix[d] according to SubscribeCIL[d] and SubscribeCIU[d];
    14. calculating the intersection of all bit matrices of each dimension;
}
```

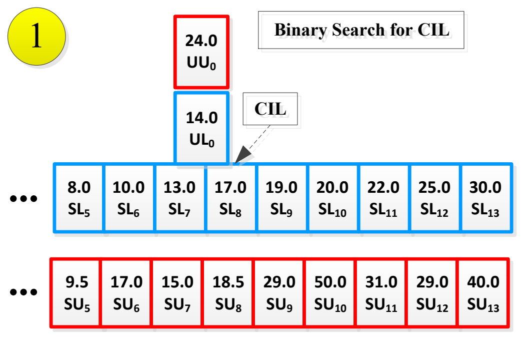
Figure 4: Procedures of BSSIM for static matching.

All above is the algorithm description for static matching, when coming to dynamic matching, that is to say, when an update or subscription region has changed, how BSSIM algorithm works. In Raczy's sort-based algorithm, all matching relationships between update and subscription regions should be re-calculated again when one update or subscription region changes. While in BSSIM algorithm, there is no need to re-calculate the interest matching information for every update and subscription regions, we only need to calculate the overlapping information for the changed regions again.

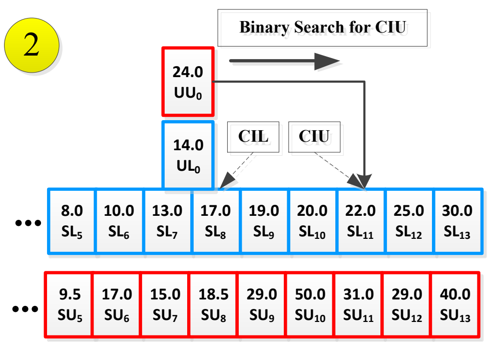
For example, when an update region changes, the original overlapping information of this region is removed, all the values of the corresponding row in the bit matrix are set to 1. Then binary search method is used to calculate the CIL and CIU value for the changed update region through comparing the lower and upper bound with SL bound list. Now we can determine that the lower bounds of which subscription regions locate in the changed update region. However, this is not enough according to the judging conditions for region overlap described above. Next, we should calculate out in which subscription regions that the lower bound of changed update region R locates. In the view of Pan's sort-based algorithm, we also introduce two parameters, maxURRS and maxSRRS, representing the maximum update region range size and maximum subscription region range size. However, these two parameters are used in a different way. They are used to quickly calculate out which subscription regions contains the lower bound of the changed update region R. As shown in Figure 5, maxURRS and maxSRRS can confine the left comparison scope, binary search is used to acquire the right comparison scope exactly. Through this way, the comparison overhead for dynamic matching can be reduced in a large degree. In Figure 5, the maxSRRS is 30.0, so UL_0 should compare with the SU values whose corresponding SL value is no less than (UL_0 - maxSRRS). Otherwise, there is no need to compare UL value with the SU values. Compared with Pan's sort-based algorithm, the dynamic matching method of BSSIM is more suitable for large spatial environment.


24.0 UU0
Binary Search
14.0 UL0
CIL
CIU
8.0 SL5 | 10.0 SL6 | 13.0 SL7 | 17.0 SL8 | 19.0 SL9 | 20.0 SL10 | 22.0 SL11 | 25.0 SL12 | 30.0 SL13
9.5 SU5 | 17.0 SU6 | 15.0 SU7 | 18.5 SU8 | 29.0 SU9 | 50.0 SU10 | 31.0 SU11 | 29.0 SU12 | 40.0 SU13
Conparison Scope
UL0 - maxSRRS

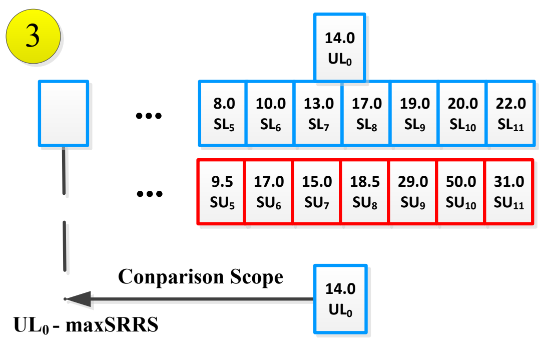
(a): Overall process.


1
Binary Search for CIL
24.0 UU0
14.0 UL0
CIL
8.0 SL5 | 10.0 SL6 | 13.0 SL7 | 17.0 SL8 | 19.0 SL9 | 20.0 SL10 | 22.0 SL11 | 25.0 SL12 | 30.0 SL13
9.5 SU5 | 17.0 SU6 | 15.0 SU7 | 18.5 SU8 | 29.0 SU9 | 50.0 SU10 | 31.0 SU11 | 29.0 SU12 | 40.0 SU13

(b): First phase.


2
Binary Search for CIU
24.0 UU0
14.0 UL0
CIL
CIU
8.0 SL5 | 10.0 SL6 | 13.0 SL7 | 17.0 SL8 | 19.0 SL9 | 20.0 SL10 | 22.0 SL11 | 25.0 SL12 | 30.0 SL13
9.5 SU5 | 17.0 SU6 | 15.0 SU7 | 18.5 SU8 | 29.0 SU9 | 50.0 SU10 | 31.0 SU11 | 29.0 SU12 | 40.0 SU13

(c): Second phase.


3
14.0 UL0
8.0 SL5 | 10.0 SL6 | 13.0 SL7 | 17.0 SL8 | 19.0 SL9 | 20.0 SL10 | 22.0 SL11
9.5 SU5 | 17.0 SU6 | 15.0 SU7 | 18.5 SU8 | 29.0 SU9 | 50.0 SU10 | 31.0 SU11
Conparison Scope
14.0 UL0
UL0 - maxSRRS

(d): Third phase.

Figure 5: Dynamic matching for changing one update region.

3.3 Complexity Analysis

To analyze the computation complexity of BSSIM algorithm for static matching, for simplicity, we suppose there are N update regions and N subscription regions. Sorting N update regions and N subscription regions with quick sort method requires about O(2NlogN). While in the Raczy's sort-based algorithm, sorting the bounds requires about O(4Nlog(4N)), Pan's sort-based algorithm need O(4NlogN) computation complexity. Calculating CIL values for N update regions and N subscription regions requires O(N) computation complexity. Calculating CIU values using binary search requires O(NlogN) in worst case. The search range of binary search method for calculating a CIU value is [CIL, N]. It is obviously that CIL values can be changes from 0 to N, so the computation complexity for calculating CIU values will be O(NlogN-N) in normal conditions. The computation complexity for generating the final overlapping information in each dimension depends on the exact matching numbers. In total, the computation complexity of BSSIM algorithm is O(NlgN), but better than the O(NlgN) of Raczy's and Pan's sort-based algorithm.

When coming to dynamic matching, for simplicity, we suppose that only one update region has changed. First step is to remove the original overlapping information about this region, the computation complexity depends on the number of subscription regions overlapping with the changed update region. Second step is to calculate CIL value for this region with binary search method, because the lower bound list has been sorted, so calculating CIL value will require O(logN) computation complexity. The third step is to calculate CIU value, through binary search the cost is less than O(logN). Then comparing the upper bound value of the changed update region with the subscription region upper bounds limited by CIL value and (UL_0 - maxSRRS) value. In total, the computation complexity for dynamic matching will be O(logN) in normal conditions, and O(N) in worst case.

4 EXPERIMENT RESULTS

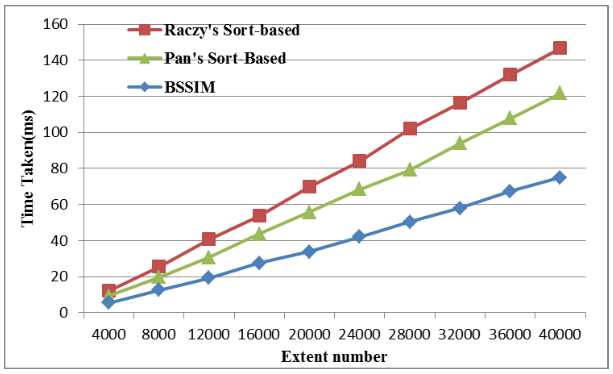
In this section, several experiments are constructed to compare our proposed algorithm with Raczy's and Pan's sort-based algorithm. Gary Tan et al introduced the concept of overlap rate to verify algorithm efficiency, which represents the ratio of total update and subscription regions to the routing space. Therefore overlap rate is also used as an experimental parameter. At this time, overlap rate is the ratio of the total length of update and subscription regions to the length of interest space. In the below formula, I is the region length, L stands for the length of the interest space. The length of all regions is same and fixed, but the lower bounds of all regions are distributed in the interest space uniformly. When the overlap rate and number of regions are determined, the length of interest space can be determined.

$$overlap\ rate = \frac{\sum_{i=1}^{N} I_i}{L}$$

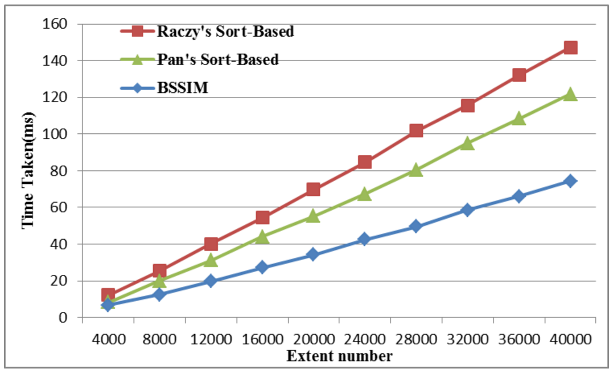
Different overlap rates are utilized to construct more comprehensive experiments, in order to fully verify the efficiency of the algorithm. There are 3 overlap rates, namely 0.01, 1 and 100, represent low overlap rate, medium overlap rate and high overlap rate respectively. The number of update and subscription regions extends from 4000 to 40000 in incremental steps of 4000. In each experiment, the experiment program uses Raczy's sort-based, Pan's sort-based algorithm and BSSIM algorithm to calculate the overlap information between update and subscription regions using same data set independently. Then modify the data set and repeat the program for 100 iterations and calculates the average performance for the 100 iterations.

4.1 Static Matching

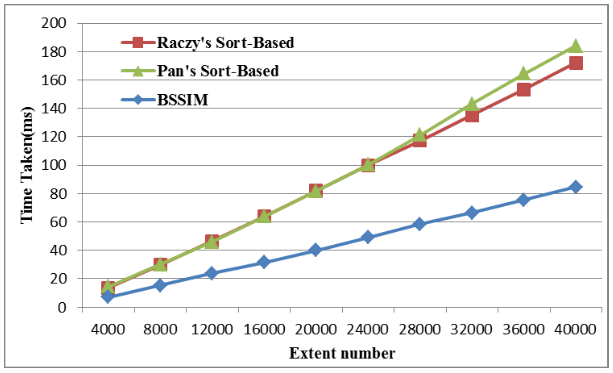
As shown in Figure 6, our proposed BSSIM algorithm is not obviously affected by the overlap rate. In fact, according to analysis in previous section, the time complexity is not affected by the overlap rate. At different overlap rates, our algorithm always performs much better than Raczy's and Pan's proposed sort-based algorithm when the number of update and subscription regions changes from 4000 to 40000. About



(a): Overlap rate is 0.01.



(b): Overlap rate is 1.



(c): Overlap rate is 100.

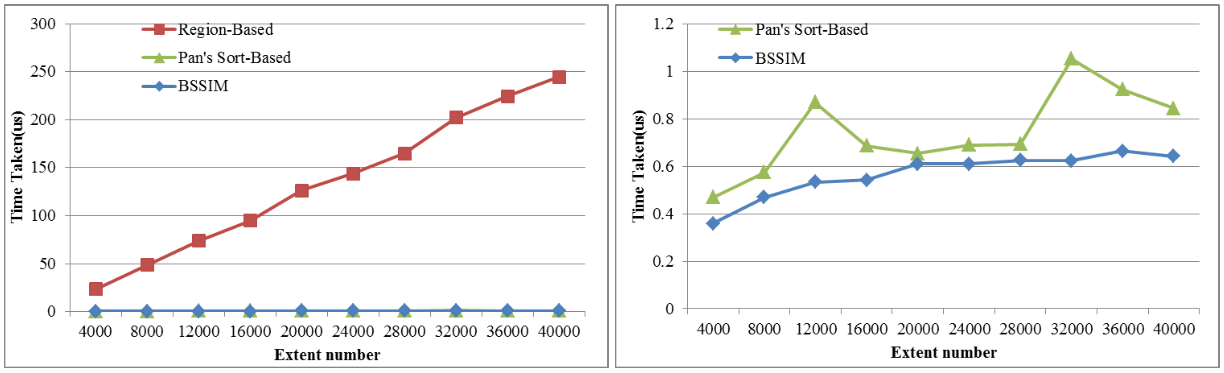
Figure 6: Performance comparison for static matching.

82%-129% performance improvement compared with Raczy's proposed sort-based algorithm and 57%-117% performance improvement compared with Pan's proposed sort-based algorithm can be achieved at different overlap rates.

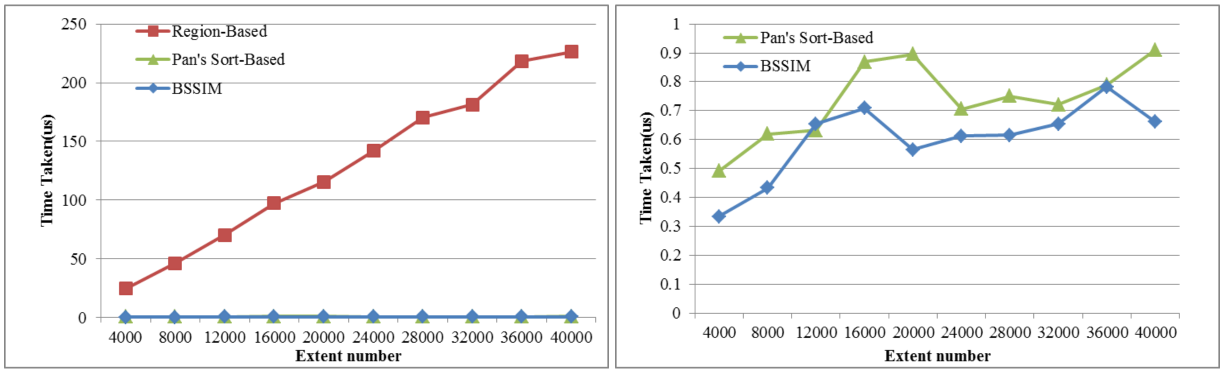
There are two main reasons why our proposed algorithm can take such a big superiority for static matching. One reason is the reduction of sorting overhead. In Raczy's proposed sort-based algorithm the length of bound list is 4*N, so the computation complexity of sorting bound list is O(4Nlog(4N)), when N is large, sorting bound list will occupy a large proportion of overhead. Similarly in Pan's proposed sort-based algorithm, there are for bound lists to sort, the sorting cost is O(4NlogN). While in our proposed BSSIM algorithm, there only two lower bound list to be sorted. The sorting computation complexity is O(2NlogN). Hence, our proposed BSSIM algorithm have advantages in sorting, especially when the sorting occupies a large proportion of computational overhead in practice. The other reason is that when generating overlap information, Raczy's proposed sort-based algorithm introduces a large number of bit operations. The length of the bit array is N, and all the N bits are operated at the same time. So large proportion of bit operations are unnecessary which will introduce a lot of overhead. In Pan's proposed algorithm, there are two sufficient and necessary conditions for overlapping. To prove the overlapping information between two regions, the two conditions must be satisfied at the same time. However, single condition will generate some wrong overlapping information which will introduce large amount of overhead. In our proposed BSSIM algorithm, on one hand we avoid the frequent bit array operations. On the other hand, our proposed sufficient and necessary conditions will not generate wrong overlapping information. Anyone of the conditions can obtain exact overlapping relationship.

4.2 Dynamic Matching

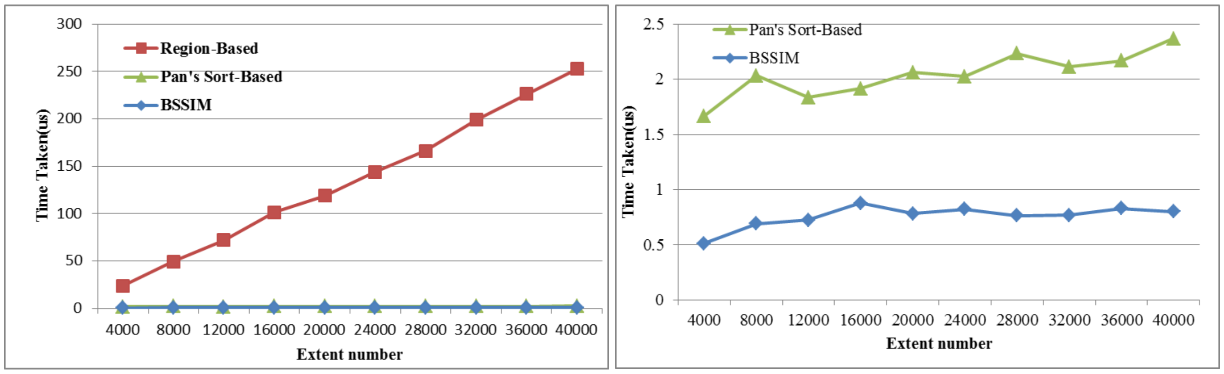
Dynamic matching means that re-calculating the matching information when one or several update or subscription regions have changed. As shown in Figure 7, to validate the efficiency of our proposed algorithm for dynamic matching, we compare it with two other algorithms, they are region-based algorithm (brute force algorithm), Pan's sort-based algorithm. Raczy's proposed sort-based algorithm cannot deal with dynamic matching of a selective region modification without repeating the static matching of all regions, which is very costly (Pan et al 2007). So in dynamic matching experiments, we don't compare BSSIM algorithm with Raczy's proposed sort-based algorithm.



(a): Overlap rate is 0.01.



(b): Overlap rate is 1.



(c): Overlap rate is 100.

Figure 7: Performance comparison for dynamic matching.

In the experiments for dynamic matching, we only consider the condition that only one update region will change. At the same time, the 3 overlap rates (0.01, 1 and 100) are used to construct different overlapping scenarios. First, three algorithms are used for static matching to obtain matching results, then change one update region, re-calculate the overlapping information. We can recognize that our proposed algorithm outperforms Pan's sort-based algorithm at different overlapping rates and different extent numbers. Especially when the overlap rate is 100, the performance improvement can be 200%, because Pan's sort-based algorithm will generate large amount of unnecessary overlapping information in high overlap rate, and filtering the wrong matching information will cost a lot.

5 CONCLUSION AND FUTURE WORK

In this paper, we proposed a binary search enhanced sort-based interest matching algorithm (BSSIM) with two exclusive judging conditions for region overlap. Based on the conditions, binary search can efficiently obtain the overlapping information without unnecessary bit operations and additional comparison operations. Besides, the sorting overhead is remarkably reduced through reduce the number or length of bounds list to be sorted. Experimental results show that BSSIM algorithm outperforms the sort-based algorithm at different overlap rates and extent numbers.

Next, our proposed algorithm will be applied to multidimensional scenarios, and more experiments should be done to optimize BSSIM algorithm for dynamic matching in more complex situations.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China under Grant (No. 61702527).

REFERENCES

Berrached, A., M. Beheshti, and O. Sirisaengtaksin. 1998. "Evaluation of Grid-based Data Distribution in the HLA". In *Proceedings of the 1998 Conference on Simulation Methods and Applications*, November 1th-3th, Orlando FL, 209-215.

IEEE Standard 1516 (HLA Rules), 1516.1 (Interface Specification) and 1516.2 (Object Model Template). Sep. 2000.

Morse, K. L. and J. S. Steinman. 1997. "Data Distribution Management in the HLA Multidimensional Regions and Physically Correct Filtering". In *Proceedings of the 1997 Spring Interoperability Workshop*, March, paper no. 97S-SIW-052.

Morse, K. L. and M. D. Petty. 2001. "Data Distribution Management Migration from DoD 1.3 to IEEE 1516". In *Proceedings of the 5th IEEE International Workshop on Distributed Simulation and Real Time Applications*, August, 58-65.

Pan, K., S. J. Turner, W. Cai, and Z. Li. 2007. "An Efficient Sort-Based DDM Matching Algorithm for HLA Applications with a Large Spatial Environment". *21st International Workshop on Principles of Advanced and Distributed Simulation*. San Diego, USA.

Raczy, C., J. Yu, G. Tan, T. S. Chuan. 2002. "Adaptive Data Distribution Management for HLA RTI". In *Proceedings of 2nd European Simulation Interoperability Workshop*, Harrow, UK.

Raczy, C., G. Tan, and J. Yu. 2005. "A Sort-Based DDM Matching Algorithm for HLA". *ACM Transactions on Modeling and Computer Simulation*, 15(1):14-38.

Tan, G., A. Raczy, Y. Zhang, and F. Moradi. 2000a. "Grid-based Data Management in Distributed Simulation". In *Proceedings of 33rd Annual Simulation Symposium*, Washington, U.S.A., April.

Tan, G., L. Xu, F. Moradi, and Y. Zhang. 2000b. "An Agent-based DDM Filtering Mechanism", In *Proceedings of MASCOTS*, August, San Francisco CA, USA.

Tan, G., Y. Zhang, and R. Ayani. 2000c. "A Hybrid Approach to Data Distribution Management". In *Proceedings of the 4th IEEE International Workshop on Distributed Simulation and Real-Time Applications*, August, pp. 55-61.

Van Hook, D. J. and J. O. Calvin. 1998. "Data distribution management in RTI 1.3". In *Proceedings of the Simulation Interoperability Workshop*, March, paper no. 98S-SIW-206.

Yu, J., C. Raczy, and G. Tan. 2002. "Evaluation of Sort-based Matching Algorithm for the DDM". In *Proceedings of the 16th Workshop on Parallel and Distributed Simulation*, May, Washington, USA.

Zhang, Y. 2000. *A Simulation Platform for Investigating Data Filtering in Data Distribution Management*, M.Sc thesis, School of Computing, National University of Singapore, Singapore.

AUTHOR BIOGRAPHIES

TIANLIN LI is a graduate student at National University of Defense Technology with e-mail address: ltl@mail.ustc.edu.cn.

WENJIE TANG is Senior Lecturer in the Modeling and Simulation department at National University of Defense Technology. His research focus is system simulation, parallel and distributed simulation. His e-mail address: tangwenjie@nudt.edu.cn.

YIPING YAO is full professor for system simulation at National University of Defense Technology. His research focus is system simulation, parallel and distributed simulation. His e-mail address is ypyao@nudt.edu.cn.

FENG ZHU is Senior Lecturer in the Modeling and Simulation department at National University of Defense Technology. His research focus is system simulation, parallel and distributed simulation. His e-mail address: zhufeng@nudt.edu.cn.