A New Randomized Parallel Dynamic Convex Hull Algorithm based on M2M model

YingPeng Zhang
the student in South China University of Technology
Guangzhou, China 510640
YingPeng.Zhang@gmail.com

ZhiZhuo Zhang
the student in South China University of Technology
Guangzhou, China 510640
zzz2010@gmail.com

Qiong Chen
associate professor in South China University of Technology
Guangzhou, China 510640
csqchen@scut.edu.cn

ZhiMingZhou
the student in South China University of Technology
Guangzhou, China 510640
zzm_guitar@163.com

ShengZhou luo
the student in South China University of Technology
Guangzhou, China 510640
lsz_xp@126.com

Abstract

In this paper, we introduce a new randomized parallel convex hull algorithm based on dynamic M2M data structure whose operation such as inserting or deleting costs O(1) time. In practice, this algorithm is faster than the classical convex hull algorithms such as Grahan scan, quick hull and Jarvis march. As with other M2M algorithm, this algorithm share an identical preprocessing which takes the majority of the costing time of the entire algorithm. Such characteristic is very helpful in many applications where a large number of operations need to be executed to the same data set.

1 Introduction

The convex hull of a set of points is one of the most basal problem in computational geometry and is applied in many fields, such as pattern recognition, image processing, statistics and GIS. It also serves as a tool and a building block for a number of other computational geometric algorithms.

Ron Graham presented the first $O(n \log n)$ algorithm for finding the convex hull of points in the plane in 1972[1]. If the points are already sorted by one of the coordinates or by the angle to a fixed vector, then the algorithm takes $O(n)$ time. Another $O(n \log n)$ solution is the divide and conquer algorithm for the convex hull, published in 1977 by Preparata and Hong[2]. This algorithm is also applicable to the three dimensional case. Later, Avis[4] and Yao [5] proved lower bounds of $\Omega(n \log n)$ on the time to find a convex hull, where sorting has an $\Omega(n \log n)$ lower bound, and Convex hull implements sorting impliedly.

R.Jarvis constructed an "output sensitive" algorithm whose running time depends on the output size[3]. Jarvis's algorithm runs in $O(nh)$ time where h is the number of points in the convex hull. In 1986, Kirkpatrick and Seidel [6] computed the convex hull of a set of n points in the plane in $O(n \log h)$ time. (Later, the same result was obtained by Chan using a much simpler algorithm [7].) The same authors showed that, on algebraic decision trees of any fixed order, $O(n \log h)$ was a lower bound for computing convex hulls of sets of n points， where h was the vertices of the convex hull.

Starting with seminal work by Clarkson, randomized algorithms have played an increasingly important role in computational geometry and many randomized convex hull algorithms were proposed [8, 9, 10].

In practice, we often require to compute the convex hull in the points set that is changing in a small scale. So dynamic convex hull algorithm is also widely studied [11-15].

In order to reduce the time complexity, some researchers focus on the problem of designing very fast parallel algorithms for convex hulls[16].

In this paper, we mainly discuss a new efficient randomized parallel Convex Hull algorithm with dynamic data structure based on M2M model which was proposed in [17]. The data structure based on M2M model costs O(n) time to preprocess data. The operations for querying, insertion and deletion on conventional data structure (like kd-tree and quadtree) may cause balance problem of the tree, however, those operations are independent on each point in the M2M structure and can be executed in parallel. By using the parallel computation, the time complexity can be reduced to O(n). We also introduce a new concept of preprocessing sharing, which greatly improves the efficiency of some multi-operation problem such as image processing and pattern recognition. Like the other algorithms based on M2M model, the convex hull algorithm based on M2M model is suitable for dynamic environment, and conveniently makes trade-off between the efficiency and the precision.

Structure of the paper: In Section 2 we introduce the M2M model and its data structure. In Section 3 we introduce the convex hull algorithm based on the M2M model and prove the correctness of this algorithm. In Section 4 several comparison experiments between our algorithm and other famous convex hull algorithms was carrried out. In Section 5, some discussions about the M2M convex hull algorithm are given.

2 Macro-to-Micro Model

2.1 The origin of M2M model

The idea of M2M is derived from human thinking pattern. When people tackle practical problems, at first they used to analyze it at macro level rather than details and then go on to narrow it to exclude some unnecessary factors until they can solve the problem rapidly at an appropriate micro level. With the M2M model, our computer can have the ability as human being to comprehend a problem from a universal perspective. In the more abstract view, the processing from macro to micro is achieving the goal of shrinking the search space. In fact, this idea is inherent in many algorithms of "Decrease-and-Conquer".

Generally, Using M2M model to solve problems include the following two steps:

1) **Preprocessing:** Data set should be divided into a number of similar partitions through Macro to Micro levels. This processing is similar to human being's behavior when developing the view of the problem.

2) **Query:** From macro to micro, shrink the search space at every level and use the algorithms based on M2M to find the solution quickly.

2.2 Terminology Explanation

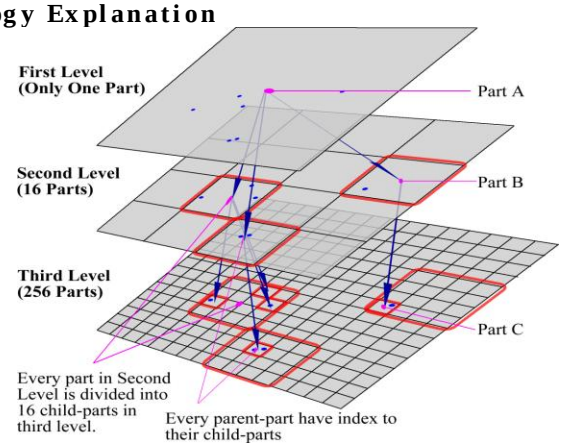

Figure 1: Terminology Explanation

Before the further introduction of the data structure based on M2M model, we firstly explain some terminologies which are used frequently when describing the M2M model.

1) **Level:** From the abstract view, different levels present the different way of data classification according to the different precision. As Figure 1 shows, there are three levels. Only one part belongs to the first level, 16 parts belong to the second level and 256 parts belong to the third level.

2) **Part:** Part is defined as subset of the data points similar to each other. In the M2M model, the part can be designed as a small square. All the data sets in the square belong to such partition. In addition, the size of the parts is nearly the same in the same level (size is referred to the cover area in the two-dimension).

3) **Last-level, next-level, up-level and down-level:** We define the level according to their parts' size. The size of certain level is smaller than its last-level and the size of certain level is bigger than its next-level as well. Take Figure 1 as example, the first level is the last-level of the second level, the third level is the next-level of the second level. All the next-levels under the certain level can be called the down-levels and all the last-levels upper than the certain level can be called the up-levels as well.

4) **Parent-part, child-part, ancestor-part and descendant-part:** The parent-part of a certain part refers to the part belongs to the last-level and contains this part. Similarly, the child part of a certain part is defined as the part included in the part to the last-level. Just as Figure 1 illustrates, part A is the parent-part of part B, part C is the child-part of part B. All the parent-parts belonged to certain part can be called the ancestor-part of this part. Similarly, all the child-part belonged to certain part can be called the descendant-part of this part.

3 The Convex Hull Algorithm Based on M2M Model

3.1 The data structure based on the M2M model

M2M is an algorithm model, basing on which many algorithms can be approached. The data structure to realize this model is flexible, as per different situations. However, there are some basic requirements needed to be satisfied.

1) Given a query point, it should take O(1) time to index the part which the point belongs to.

2) Insertion or deletion of a data point should take O(1) time.

3) Given the index of the part, it takes O(n) time to visit every child-part of the given part, where n is the number of the child-part.

4) Given the index of the part, it takes O(n) time to visit all the data points of this part, and n is the number of the data points of this part.

5) Given the index of the part, it takes O(1) time to get the index of its ancestor-part.

6) The time complexity of preprocessing should be O(n). And parallel calculation should be supported.

It is clear that the algorithm which satisfies all the requirements above achieves the trivial lower bounds and is theoretically optimal.

In order to satisfy those requirements, the following data structure is used in our convex hull algorithm in the planar case:

1) A 2-dimension array index is applicable for every part in the same level. Because query, inserting or deleting an array element approximate cost O(1) time, the 1st and 2nd requirements are satisfied.

2) Every part maintains the index list of its child-parts, so that the 3rd requirement can be satisfied through visiting the child-parts list (alternately, when the number of child-parts is small, the space of the index list is no longer needed because of the regular partition).

3) The most micro part maintains a list of the points it contains. When we want to visit the points in a certain part, the breadth search tree can be built by taking query part as the tree

root，then we can traverse every point belong to this part. The time complexity of this processing is O(n), that is, the 4th requirement is also satisfied.

4) Because the partitions are regular, it is easy to calculate the index of parent-part by the index of current part (accomplished by a multiplication for scaling and a floor function to find the integer part index). This processing finishes in constant time, so the 5th requirement is satisfied.

5) As we explain at the 2nd statement above, the preprocessing is composed of a series of insertion. every insertion costs O(1) time, therefore the time complexity of preprocessing is O(n) and the preprocessing can be computed in parallel, which satisfies the 6th requirement.

The data structure described above is built in the step of preprocessing. Because there isn't any dependency of each data points in the M2M preprocessing, it is convenient to run the preprocessing in parallel, which will improve the efficiency greatly (the details about the parallelism of M2M is in [17]). With the same reason, the insertion or deletion of data points can be operated dynamically and have no affection to other points in the M2M structure, so we need not to preprocess the data again when some changes appear.

3.2 The Querying Procedure of The Convex Hull Algorithm Based on M2M Model (M2MCH)

Table 1: Terminology Explanation

Representative-Point	An arbitrary point in the part of original point set which is designated in the preprocessing.
Center-Line of two parts	The line connects the centers of two given parts.
Representative-Line	The line connects the deputies of two given parts.
Vertices of hull	The points that are used to compose the final convex hull.
Center-Hull	The convex hull of the center points of parts in a level.
Representative-Hull	The polygon that is composed by the Representative-Point belonging to Center-Hull in a level.

Table 1 explain the terminologies that are used in the querying procedure of M2MCH. In the M2MCH, there are two basic steps: preprocessing and querying. Preprocessing is identical for many algorithms based on M2M model (more detail in [17]), we describe the details of convex hull querying in the follow.

After preprocessing, the M2M data structure is established. Our algorithm will firstly get all the parts containing at least one point and compute the convex hull of the centers of those parts in topmost level, which is shown in Figure 3. Secondly, all the subparts of the parts which are in the Center-Hull will be considered in the next level. In the next level, the algorithm will check all the parts handled by the last level and similarly find out all the parts containing at least one point, then compute the Center-Hull of those parts, as shown in Figure 4. It is worthwhile to notice that our algorithm only concerns the centers of the parts handled by the last level and containing at least one point, which leads to a great shrink to the searching space. The algorithm repeats similar processing in the following levels, until reaching the bottom level. In the bottom, the algorithm will consider not just the centers of the parts handled by the last level but all the points contained in those parts, and output the convex hull among considering points, as shown in Figure 5.

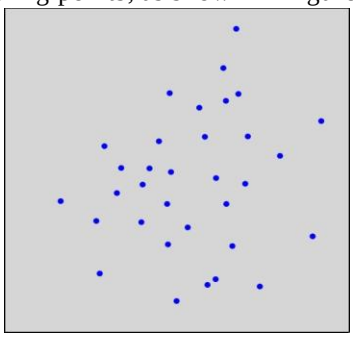

Figure 2: The point set

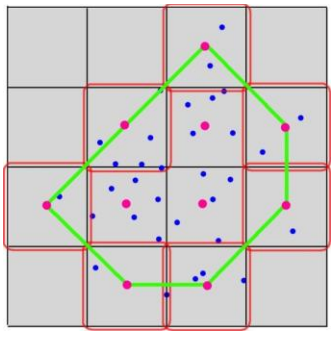

Figure 3: The Center-Hull in the top level

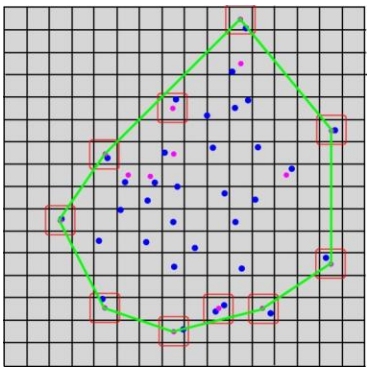

Figure 4: The Center-Hull in the next level

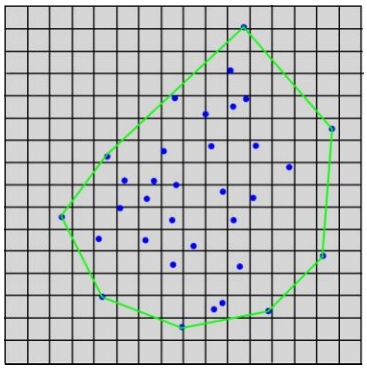

Figure 5: After several levels computing, the algorithm returns the final convex hull in the bottom level

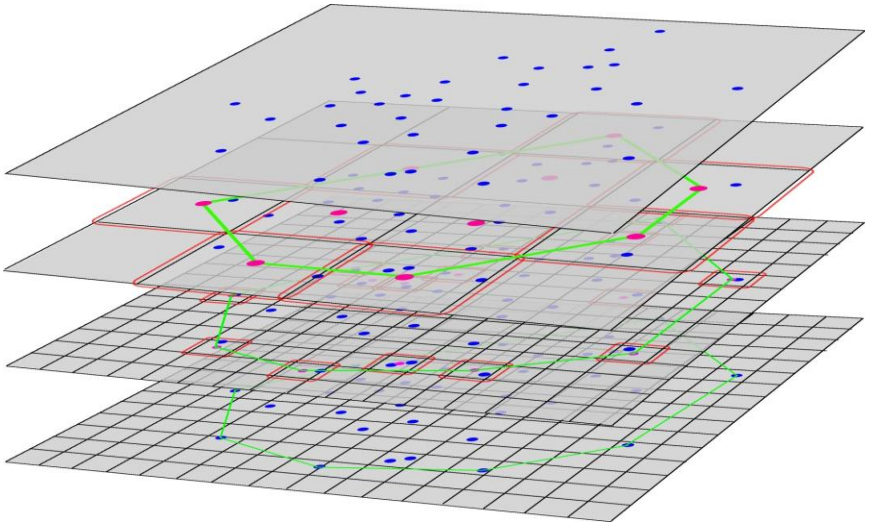

Figure 6: The summary of the whole processing in the above example

Although the description above can express the general idea of M2MCH, it can't guarantee correctness of the final output. In order to ensure the correctness of our algorithm, we will not only consider the parts containing Center-Hull but also includes the parts intersecting with the Representative-Hull. In the following section, we will prove the correctness of the convex hull algorithm based on M2M model.

3.3 The Correctness of M2MCH

In order to prove the correctness of M2MCH, we first introduce four lemmas.

Lemma1: In the current level, all the parts whose centers are outside of the Center-Hull contains no point (hull points as well) in them.

Lemma2: The area inside of Representative-Hull contains no hull points.

Lemma3: All parts have at least one intersection with the Representative-Hull, if their centers are inside of Center-Hull but outside of the Representative-Hull.

Lemma4: All the parts which contain hull points have at least one intersection with the Representative-Hull.

Proof: According to Lemma 1 and Lemma 2, all the hull points exist in the parts that satisfy two conditions. One condition is that those parts' centers should be inside of the Center-Hull. The other is that those parts should be some area outside of the Representative-Hull. According to Lemma 3, the parts whose centers are inside of Center-Hull but outside of the Representative-Hull and which also satisfy above two conditions, have been proved to have at least one intersection with the Representative-Hull. The rest parts which satisfy the two

conditions above but not conform to the precondition of Lemma 3, are those whose centers are inside both Center-Hull and Representative-Hull and are some area outside of the Representative-Hull. This kind of parts also have at least one intersection with the Representative-Hull, because the center points of them inside mean that some area of those parts is inside in the Representative-Hull. Meanwhile, some other area of them is outside, and it is evident that there is at least intersection between those parts and the Representative-Hull. Therefore, the Lemma 4 is proved.

Proof of the correctness of M2MCH:

The proof uses the following **loop invariant**:

Initialization: The querying processing begins from the top most level, which includes all the points on the original points set. Hence, at the initialization, all hull points will be included in the first level.

Maintenance: According to Lemma 4, our algorithm adds all the parts which have intersections with the Representative-Hull, in another word, including all the hull points, to the ChildSet. Hence, it guarantees that the input part set of next iteration contains all the hull points.

Termination: At the bottom level, the correct convex hull (all the hull points) is within the final input set according to loop invariant in maintenance. The inner algorithm can generate the correct convex hull if and only if the input set contains the desiring points. This completes the proof.

4 Experiment

To learn more about the performance of M2M algorithm, we construct the following experiments to compare the convex hull algorithm based on M2M with the classical covex hull algorithm: Graham scan [1], quick hull [2] and Jarvis march [3]. (Jarvis march is time-consuming, so the result didn't show in the diagram), In the experiments, planar points are generated randomly from uniform distribution. Table 4 shows the experimental enviroment.

Table 4: The experimental environment

CPU:	Intel(R) Pentium(R) 1.73GHz
Memory:	1 GB
Operation System:	Windows XP sp2
Programming Language:	C# 2.0

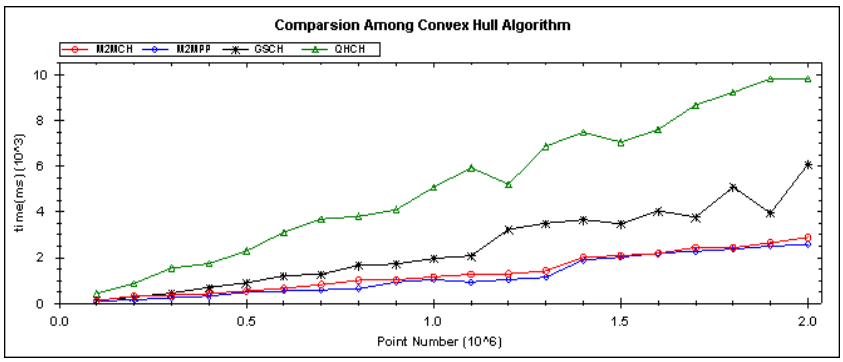

Figure 8: The comparison of consumed time among convex hull algorithms.

In the small scale points set, the Graham's scan has the best performance, and the ratio of preprocessing in M2MCH is about 50%, in order words, the ratio of querying convex hull processing in M2MCH is also about 50%.

Whereas in the large scale points set, beyond 1 million, M2MCH outperforms other algorithms as showing in Figure 8. It is obvious that Graham's scan costs about twice more time than M2MCH as the number of points keeps increasing. In addition, the ratio of preprocessing in M2MCH is up to 95%, that is, the ratio of querying in M2MCH is less than 5%. This fact suggests that the cost can be reduced greatly when the preprocessing is computed in parallel.

5 Discussion

5.1 The advantage of Convex Hull Algorithm of M2M model

Compared to conventional algorithms, Convex hull algorithm based on M2M model have following advantage:

1) **High parallelism:** The preprocessing of M2M algorithm which running time accounts for 95 percent of the total time for convex hull algorithm in practice, can be run in multi-independent channels at the same time which can reduce the running time complexity to O(1) in theory [17].

2) **Dynamic structure:** The operation of M2M data structure such as inserting or deleting can be finished in O(log n) time in the worst case and in O(1) time in most cases. Moreover, these additional operations will not delay or influence the processing of querying convex hull.

3) **Preprocessing sharing:** The algorithms based on M2M model share an identical preprocessing which takes the mainly part of the costing time of the entire algorithm. Such characteristic is very helpful in the image processing field where many operations need to be executed on the same image (point set as well) and many algorithms used in image processing can be approached based on the M2M model. All those algorithms share one preprocessing, thus the efficiency of the whole processing is greatly improved.

4) **Trade-off between the efficiency and the precision:** As with other M2M algorithm, convex hull algorithm based on M2M can easily trade-off between the efficiency and the precision, that is, it is a probably approximately correct algorithm. For example, if we output the Deputy Hull of certain level instead of the convex hull in the bottom level, the time of computation is shorter but the result is approximately correct, and it is no doubt that the more closer the stop level is to the bottom, the more precise the result is.

5) **Trade-off between the time efficiency and space cost:** The parameters of M2M model such as the number of levels and the way of partition at each level can be changed on certain purpose. Generally speaking, the more subtler of the Macro-Micro levels being divided and the more smaller the ratio of the partition between the adjacent levels is , the higher the cost of the space may be, and the higher the time efficiency may get.

5.2 Other application of M2M model

With the help of M2M model, the algorithms can be designed to solve many classical problems: such as nearest points, convex hull, TSP, cluster, path finding, collision detection and so on. Thus, tasks in many application fields (including geography information system, data mining, pattern recognition, image processing and real time strategy game) related to those fundamental algorithm can be better performed. Taking image processing for example, we can preprocess for a specific points set, and then do convex hull querying, nearest points querying, or area querying with out preprocessing again.

6 Conclusion

In addition to the less time cost than the classic algorithms in the large scale points set, the M2MCH algorithm integrates the characteristics of parallel, dynamic and randomized computations, even through the recent researches [8-16] just focus on one of these characteristics. Moreover, the unique preprocessing sharing character bigger theoretical value and more application potential.

7 Future work

1) **Perfecting M2MCH:** M2M model is a newborn computation model, base on which the convex hull algorithm is also an original one. Together with the advantage of the algorithm, there are some configuration issues outstanding, such as the affect of different inner algorithm, the way of stratification and how to select the Representative-point.

2) **The Time Complexity of M2MCH:** As we know, the average time complexity depends on the distribution of the input set; therefore, it is difficult for us to define the time complexity without knowing the real point set. Briefly, from the trend of the comparison

to Graham's scan and Quick hull which are O(nlogn) time complexity, we believe the time complexity of M2MCH is no greater than O(nlogn).

3) **Extending To Higher Dimension:** It wouldn't be difficult to extend M2M algorithms for higher dimension, but it requires higher-dimension surface construction algorithm (as draw-line algorithm used in the planar case) and higher-dimension convex hull algorithm as an inner algorithm。

4) **Contribution To M2M Model**: As we have iterated, many graphics algorithms can be designed base on M2M model, and benefit from its general advantages. Hitherto, we have implemented some algorithms based on M2M model that include nearest points searching, convex hull and path finding, but there are many other algorithms waiting for research.

References

[1] R. L. Graham, An efficient algorithm for determining the convex hull of a finite planar set, Information Processing Letters 1 (1972) 132–133.

[2] F. P. Preparata, S. J. Hong, Convex hulls of finite point sets in two and three dimensions, Communications of the ACM 2 (20) (1977) 87–93.

[3] A. Jarvis, On the identification of the convex hull of a finite set of points in the plane, Information Processing Letters 2 (1973) 18–21.

[4]Avis, D. Comments on a lower bound for convex hull determination. Inform. Process. Lett.11 (1980), 126.

[5]Yao, A.C. A lower bound to finding convex hulls. J. ACM 28 (1981), 780-787

[6] D. G. Kirkpatrick, R. Seidel, The ultimate planar convex hull algorithm?, SIAM Journal on Computing 15 (1) (1986) 287–299.

[7] T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. Discrete Comput. Geom., 16(4):361–368, 1996. Eleventh Annual Symposium on Computational Geometry (Vancouver, BC, 1995).

[8]Clarkson, K.L. New applications of random sampling in computational geometry. Discrete Comput. Geom. 2(1987), 195-222.

[9]Clarkson, K.L. A randomized algorithm for closest-point queries. SIAM J. Comput. 17(1988), 830-847.

[10] R. Wenger, Randomized quick hull, Algorithmica 17 (1997) 322–329.

[11] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. J. Comput. System Sci., 23(2):166–204, 1981.

[12] J. Hershberger and S. Suri. Applications of a semi-dynamic convex hull algorithm. BIT, 32(2):249–267, 1992.

[13] J. Hershberger and S. Suri. Off-line maintenance of planar configurations. J. Algorithms, 21(3):453–475, 1996.

[14] T. M. Chan. Dynamic planar convex hull operations in nearlogarithmic amortized time. Journal of the ACM, 48(1):1–12, January 2001.

[15] Gerth Stolting Brodal and Riko Jacob Dynamic Planar Convex Hull. 43 rd Annual IEEE, 2002.

[16] Neelima Gupta and Sandeep Sen, Faster output-sensitive parallel algorithms for 3D convex hulls and vector maxima, Journal of Parallel and Distributed Computation, 63 (2003) 488–500

[17] YingPeng Zhang, ZhiZhuo Zhang, Qiong Chen *A New Nearest Neighbour Searching Algorithm based on M2M Model. The International MultiConferenceof Engineers and Computer Scientists 2007（IMECS 2007）. 2007*