Adaptive Generation of Location Update in Cellular Mobile Computing Systems*

LI Guo-hui, LIU Yun-sheng

(*College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan* 430074, *China*)

E-mail: guohuili@public.wh.hb.cn

Received January 10, 2001; accepted July 30, 2001

Abstract: An important issue in the design of a mobile computing system is how to manage the real-time locations of mobile clients. In the existing commercial cellular mobile computing systems, a two-tier architecture is used. However, the two-tier architecture is not scalable and is not suitable to the new mobile computing applications in which there are a very large of mobile users. In the literatures, a hierarchical database structure is proposed. The location databases of different cells are organized into a tree structure to facilitate the search of mobile clients. Although this architecture can distribute the update and searching workload amongst the location databases in the system, it has the problem of heavy location update overhead and long search delay. This paper studies how to generate location update based on the distance-based update method in the tree structure. A location update generation method is proposed to calculate the optimal distance threshold with the objective to minimize the total location management cost.

Key words: mobile computing system; location management for moving objects; location update

One of the most important issues in the design of a mobile computing system is location management of mobile clients[1~4,6~8]. In a cellular mobile network, the whole service area is divided into a collection of inter-connected *cells*. In each cell, there is a base station which communicates with the mobile clients in the cell. The base stations are connected by a high-speed wired network. The mobile clients may move within their current cells or move into other cells. While a mobile client is moving, the system has to record down its real-time location since other mobile clients may generate queries on the location of the client and may want to communicate with the client.

In the existing cellular mobile network, a two-tier database structure is used to manage the locations of mobile clients[13]. In the system, one of the base stations is defined with a location database, called Home Location Register (HLR) which maintains the client profiles, including the real-time locations of the mobile clients. In addition, each base station also maintains a Visitor Location Register (VLR) to record the mobile clients which are currently within the cell responsible by the base station. When a mobile client moves out of its current cell and enters another cell, a new entry of the client location is added into the VLR of the new cell and then the HRL will be also updated. In locating a mobile client for a query, the VLR of the cell, where the query is initiated, will be searched first. If the client cannot be found in the VLR, a request will be sent to the HLR of the client to find out its location. Once the cell, where the location of the client is recorded, has been identified, polling messages will be sent out in the cell to communicate with the client to ensure that it is the right cell where the client is now residing.

---

* **LI Guo-hui** was born in 1973. He is an associate professor at the College of Computer Science and Technology of Huazhong University of Science and Technology. His research interests are database, transaction processing systems and mobile computing systems. **LIU Yun-sheng** was born in 1940. He is a professor and doctoral supervisor at the College of Computer Science and Technology of Huazhong University of Science and Technology. His research interests are advanced database systems and information systems.

This two-tier architecture is simple. However, it has several serious performance problems which make it not suitable to the many new mobile computing applications. The number of mobile clients in a mobile computing system can be very large and the system may need to maintain a large amount of location information for mobile clients. It is obvious that the two-tier architecture is not scalable. Because a mobile client is permanently associated to an HLR, the overheads for locating a client can be very heavy if the mobility of the client is high. It is obvious that the cost for locating a mobile client highly depends on the locations of the calling mobile client and the callee.

To improve the system performance and reduce the total cost for locating a mobile client, a hierarchical database structure[1,12] is proposed to organize the location databases at different cells. Under the hierarchical database structure, each cell has a location database for the mobile clients in the cell. The location databases of different cells are organized into a tree structure with the location database at each cell as the leaf node of the tree. Although the hierarchical structure of the databases can improve the search of the locations of mobile clients, the update overhead for maintaining the real-time locations of mobile clients can be very heavy. When a mobile client crosses a cell boundary into another cell, several location databases may have to be updated and the number of location databases to be updated depends on how the databases are organized and how the client moves. Although various location update generation methods have been proposed in the research in the area, most of them are mainly designed for the two-tier database architecture. To our best knowledge, it still completely lacks of any research in the design of efficient update generation policy for the mobile clients to report their locations specifically for the system using the hierarchical database structure.

One of the proposed methods for the two-tier architecture is called distance-based update method. Although it is simple, it is efficient and it has been widely used in many existing cellular mobile network systems. However, one of the main problems of the distance-based method is that it is very difficult to define the distance threshold which is the key parameter of the method. In this paper, we design an efficient way to calculate the value for distance threshold such that the total cost for location management is minimized. Since an important requirement of the mobile computing systems is to meet the query deadlines, we also provide a solution for generating location updates with the objective to minimize the latency for locating a mobile client. The remaining parts of this paper are organized as follows. Section 1 is the related work. Section 2 discusses a new method to calculate the distance threshold value. Section 3 presents a brief discussion on the properties of the proposed methods. Finally, the conclusions and future work of the paper are given in Section 4.

1 Related Work

The research in mobile computing systems has received a lot of interest in recent years. One of the most important topics is location management. In the last few years, different location update methods have been proposed. Most of proposed methods are for systems using the two-tier location database structure. These proposed methods can be summed up into the following four basic policies, *location-area update, time-based update, distance-based update and movement-based update.*

In the location-based update method, the collection of all the cells in the system is grouped into a number of disjointed location areas. A mobile client updates its location when it enters another location area. In the *time-Based update* method, a mobile client updates its location periodically every pre-specified time interval[5]. In the *distance-based update* method, a mobile client updates its location whenever the distance between the current cell and the last registered cell exceeds a pre-defined threshold value. In the *movement-based update* method, a mobile client updates its location if the number of cells it has traveled since the last location update exceeds a pre-defined threshold value. In this paper, we will concentrate on the distance-based update method since it is widely used in the existing cellular mobile network systems.

In addition to the above methods, in Ref.[9], a paging method is proposed in which the whole service area is divided into *location areas* (LAs) and the cells in a location area are paged simultaneously. When there is an incoming call to a mobile client, the LAs are sequentially paged for the client following a pre-defined paging strategy, which defines the order for paging the LAs. To improve the probability of finding the mobile client and reduce the paging cost, in Ref.[10], based on the velocity and direction information of a mobile client, the system estimates the most possible cell that the client is residing. In Ref.[11], an active tracking policy is proposed to find out the location information of a mobile client by using non-utilized system resources to get the more detailed information on mobile clients' locations.

In recent years, a hierarchical location database structure is proposed to organize the location databases in the system[1]. It also discusses the problems such as location caching, replication, concurrency control in the hierarchical location databases. Although the hierarchical database architecture can speed up locating a mobile client, up to now, to our best knowledge, it still lacks of any efficient location update generation policy for the hierarchical structure. It is the purpose of this paper to study how to generate location update based on the distance-based method for a system with hierarchical databases for location management. Our main focus is on the definition of the optimal distance threshold for the distance-based.

2 Location Update Generation

In this section, we will introduce a method to calculate the optimal *distance threshold* for the distance-based update method so that the total location management cost and the searching latency for a mobile client can be minimised. In the distance-based method, when a mobile client x moves from its previously-residing cell, $old\_cell(x)$, into the current cell, $cur\_cell(x)$, an update will be generated to update its location to the location database of $cur\_cell(x)$ if the distance between the two cells is greater than a *pre-specified threshold*. (We call it *distance threshold* in this paper.)

In a hierarchical location database architecture, the organisation of the location databases is static and the system can pre-compute the distance between any two cells and store the distances in a matrix. When a mobile client moves across a cell boundary into another cell, the system can easily query the matrix to get the distance between the old cell and the new cell. However, how to set the distance threshold is not an easy problem. Inappropriate distance threshold value can have a serious impact on the system performance, i.e., heavy cost in updating the location of a mobile client and a long delay in locating a mobile client. If the distance threshold value is small, the locations of mobile clients will be updated frequently and a lot of system resources will be consumed on processing the location updates. On the contrary, if the distance threshold value is large, the location uncertainties of mobile clients will be large and the total cost and time delay for locating a mobile client will be very heavy. Consequently, the timing requirement of many queries cannot be satisfied. In the following sub-sections, we will first introduce a cost-based method to determine the optimal value for the distance threshold so that the total update and searching cost can be minimised. Then, we will introduce a time-based method to determine the distance threshold so that the total latency delay for locating a mobile client will be minimised.

Basically, location management in a cellular mobile computing system consists of two procedures: *location update* to report the new location of a mobile client and *paging* for a mobile client. Thus, in the design of location update generation policy, we need to consider the update cost and the paging cost if we want to minimise the total location management cost.

In the cost-based distance threshold method, we first calculate the cost for processing a location update. Then, we compare: (1) the total cost in locating a mobile client when a location update is generated to report the new location of a mobile client; with (2) the total cost in locating a mobile client if a location update is not generated to

report its new location. Finally, we calculate the optimal distance threshold value by letting the cost for location update equal to the saving in cost. As can be seen from the following discussion, the location update cost increases linearly with the distance between the new cell and the old cell of a mobile client. The saving in cost increases exponentially with the distance between the new cell and the old cell of a mobile client. When a mobile client moves across a cell boundary and the distance between the old cell and the current cell is greater than the distance threshold, the generation of a location update can decrease the total cost in locating a mobile client dramatically. If the distance between the former cell and the current cell of a mobile client is less than the threshold, the generation of location update should be deferred since the generation of location update incurs more cost than the saving in cost due to the location update.

2.1 Terminology

The ratio of the number of calls for locating a mobile client over the number of cell boundary crossing defined as call to mobility ratio (CMR) in Ref.[1] can have an important impact on the optimal value of the distance threshold. If CMR is small, i.e., there are not many location calls to a mobile client per cell boundary crossing, the generation of a location update from a mobile client will not result in much saving in the cost for locating the mobile client. So generation of a location update should be deferred for this case. On the contrary, if CMR is large, i.e., there are a large number of calls to a mobile client per cell boundary crossing, the generation of a location update can reduce the location cost significantly.

The *least common ancestor* of location databases DB_i and DB_j is denoted as $LCA(DB_i,DB_j)$. The height of $LCA(DB_i,DB_j)$ to the leaf nodes of the tree is denoted as $lca(DB_i,DB_j)$. (We assume that all of the leaf nodes of the location databases are at the same level in the hierarchical location database tree).

Definition 1. The distance between cells i and j, termed $dis(i,j)$, is defined as the height of the least common ancestor of the two leaf location databases which are responsible for cell i and cell j respectively:

$$dis(i,j)=lca(LDB(i),LDB(j))$$

If $i=j$, then $dis(i,j)=0$.

Let $cur\_cell(x)$ be the cell where mobile client x is now residing at.

Definition 2. The distance between clients x and y, termed $dis(x,y)$, is defined as the distance of the two cells in which x and y are now residing, namely

$$dis(x,y)=dis(cur\_cell(x),cur\_cell(y)).$$

The above definition captures the locality of two mobile clients. It can be seen easily that a mobile client takes a smaller cost to find the location information of another mobile client if the distance between them is smaller.

Similar to Ref.[12], when we calculate the optimal value for the distance threshold, we consider the following related costs in location update and lookup procedure:

F: The cost of sending a message to an arbitrary site knowing its physical address;

L: The cost of following a link in the tree of the location databases, i.e., sending a message to the parent or child node of a location database;

U: The cost of a database update;

Q: A database query cost;

P: The cost of polling for a specific client in a cell.

2.2 Cost-Based distance threshold calculation

To simplify the discussion, we assume that in the hierarchical location database structure, each internal node has d sub-nodes and we denote $dis(new\_cell(x),old\_cell(x))$ as dis. When a client x moves from cell $old\_cell(x)$ to cell $cur\_cell(x)$, if a location update will be generated, the database entries for x in both from $old\_cell(x)$ up to

LCA(*cur_cell*(*x*),*old_cell*(*x*)) and from *LCA*(*cur_cell*(*x*),*old_cell*(*x*)) down to *cur_cell*(*x*) have to be updated.

For example, when a mobile client *x* moves from its old cell, *j*, into a new cell, *i*, its location entries in databases *LDB*(*i*) and all the related location databases in the tree have to be updated by performing the following procedure. First, a message is sent from location database *LDB*(*i*) to its parent node to determine whether a location entry for *x* already exists in it or not. If the entry cannot be found, a record pointing to location database *LDB*(*i*) will be created at the parent node. The searching procedure is repeated until *LCA*(*LDB*(*i*),*LDB*(*j*)) is reached. Then, the procedure continues from the database *LCA*(*LDB*(*i*),*LDB*(*j*)) down to *LDB*(*j*) to clear the location entry for *x*. We can see that the whole searching procedure consists of 2*lca*(*LDB*(*i*),*LDB*(*j*)) times of messages passing and processing. Since all the database entries for *x* from the database *LDB*(*j*) to the maximum child database of *LCA*(*LDB*(*i*),*LDB*(*j*)) should be deleted and the location information of *x* in the databases *LCA*(*LDB*(*i*),*LDB*(*j*)) should be updated. For all the databases from the other maximum child database to the database *LDB*(*i*), there will be an additional database update. Totally there are 2*dis*(*old_cell*(*x*),*cur_cell*(*x*))+1 times of location updates.

To illustrate the searching and update procedure, we can refer to the example hierarchical location databases shown in Fig.1. It is supposed that mobile client *x* moves from the cell corresponding to the location database *DB*5 to the cell corresponding to location database *DB*8. To update the location of *x*, an entry for *x*'*s* location is added into *DB*8. Then the system searches the location information of *x* upward in the hierarchical databases until reaching the least common ancestor of *DB*5 and *DB*8, *DB*0. There is an entry for *x*'*s* location information at *DB*0. Then, the system continues to search for the location information of *x* until it reaches the leaf node *DB*5. After that, all the information for *x*'*s* location in *DB*5 up to *DB*0'*s* child nodes will be deleted, and in each location database from *DB*0 down to *DB*8, an entry for *x*'*s* location will be added. Thus, the total cost for the location update of *x* will be:

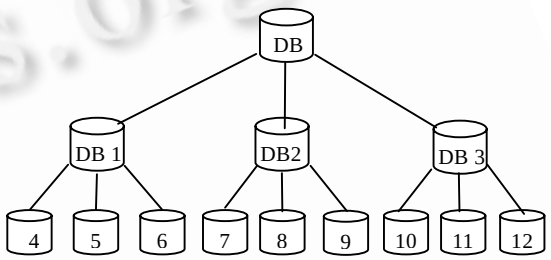

Fig.1 Hierarchical databases to store the mobile clients locations

$$update\_cost(dis)=2dis(old(x),new(x))\times L+(2dis(old(x),new(x))+1)\times U. \quad (1)$$

It is assumed that after the completion of the location update, a mobile client *y* in cell *cur_cell*(*y*) calls *x*. To find the current location of *x*, a message is sent from the cell in which *y* is now residing to its parent location database node. If the parent node does not have an entry for *x*'*s* location, a message is sent to the upper level node until the database *LCA*(*cur_cell*(*y*),*cur_cell*(*x*)) is reached where there exists an entry for *x*'*s* location. Then, a message is sent downwards following the hierarchical database structure until it reaches the current location cell of *x*. Then the system polls for mobile client *x*. In the above procedure, we can see that there are totally 2*lca*(*cur_cell*(*y*),*cur_cell*(*x*)) times of message transmission and processing, 2*lca*(*cur_cell*(*y*),*cur_cel* (*x*))+1 times of database queries, and one polling for a specific mobile client in a cell. Thus, the total cost for the case where the mobile client generates an update to report its new location is

$$2dis(cur\_cell(y),cur\_cell(x))\times L+(2dis(cur\_cell(y),cur\_cell(x))+1)\times Q+P. \quad (2)$$

If *x* does not generate a location update after it has crossed a cell boundary, the call from client *y* will be processed according to *x*'*s* old location entry in the location databases. After finding *x* is not in *old_cell*(*x*), the system polls all the possible cells to find the new location of *x*. The average number of polling is $1/2d^{dis}$. At last a forwarding pointer is linked from *old_cell*(*x*) to *cur_cell*(*x*). For example, in the example hierarchical location databases shown in Fig.1, a mobile client *x* moves from the cell corresponding to *DB*12 to the cell corresponding to the *DB*7 and the related location databases are not updated. When there is a location call for *x*, the old location

information of x will be used first and the system polls for x in the cell corresponding to $DB12$. After finding that x is not in the cell, all the cells whose distances from the cell $old\_cell(x)$ are less than the distance between $old\_cell(x)$ and $cur\_cell(x)$ are all the possible cell where x is now residing. They will poll for x until x is found. At most, there are d^{dis} cells to be polled. The average number of cells to be polled for x is thus $1/2d^{dis}$. In the example architecture shown in Fig.1, at the worst case, all the nine cells are possible to be polled for locating x and the average number of polling will be $3^2/2$. So, the first location call cost without any location update is

$$2dis(cur\_cell(y),old\_cell(x))\times L+(2dis(cur\_cell(y),old\_cell(x))+1)\times Q+P\times\frac{1}{2}d^{dis}+F+U. \quad (3)$$

After the first location call for x is resolved, the total costs for the following location calls of x for the case where a location update is generated from x to report its new location and for the case where no location update is generated from x will be similar. The only difference is that there will be one more database query and one more message sending for the no update generation case.

When we subtract equation (2) equation from (3), we can get the saving cost for processing the first location call due to the generation of a location update:

$$(3)-(2)=2(dis(cur\_cell(y),old(x))-dis(cur\_cell(y),new(x)))\times(L+Q)+F+U-P+P\times\frac{1}{2}d^{dis}.$$

In the following (CMR−1) times of calls after the cell boundary-crossing, the total searching costs at the no update case is (CMR−1)×(Q+F) more than the case where location database updates are generated.

So the total saving cost in the CMR times of location calls due to the generation of location updates is

$$saving(dis)=2(dis(cur\_cell(y),old(x))-dis(cur\_cell(y),new(x)))\times(L+Q)+F+U+P\times\frac{1}{2}d^{dis}+(CMR-1)\times(Q+F)-P.$$

If we assume that the distance between y and x's current locations and the distance between y's current location and x's previous location is the same, the saving cost due to the location update is

$$saving(dis)=F+U+P\times\frac{1}{2}d^{dis}+(CMR-1)\times(Q+F)-P. \quad (4)$$

Let $saving(dis) = update\_cost(dis)$, then

$$F+U+P\times\frac{1}{2}d^{dis}+(CMR-1)\times(Q+F)-P=2dis(L+Q)+Q+P. \quad (5)$$

We calculate the distance value, $Distance\_Threshold_{cost}$, which satisfies Eq.(5). It is the distance threshold for location update generation. When the distance between the new cell and old cell of x is more than $Distance\_Threshold_{cost}$, the location update cost is less than the saving cost due to the location update. For this case, the mobile client should generate a location update.

3 Discussion

One of the advantages of the presented method for calculating the distance threshold is that it is simple to implement and can adapt to the dynamic properties of mobile clients. Furthermore, unlike the conventional distance-based method, we do not need to assign a static value to the distance threshold. Actually, in practice, it is very difficult to determine such a static distance threshold value due to the dynamic properties of the systems. Instead, the distance threshold is defined based on the system parameters and how a mobile client moves in our proposed methods. Every time, a new distance threshold will be defined for a mobile client when it moves across a cell boundary.

It can be seen that most of the system parameters can be computed easily before the system setup, i.e., the distance between the cells can be pre-calculated and stored in a symmetric *distance matrix A*. All the cost factors

involved in the distance threshold calculation can be given in a uniform unit.

4 Conclusions and Future Work

Tracking the locations of mobile clients is central to mobile computing systems. In this paper, we study a variance of distance-based location update policy for hierarchical location database structure. The hierarchical location database structure has the ability to accommodate the increase in client population in the system. However, the high location update overhead can cause a serious penalty to the system performance. The location update distance threshold must be deliberately set to achieve the total location management cost reduction. We introduce a distance threshold value setting policy considering all the cost factors involved in the location management.

In the future mobile computing systems, the systems may need to manage a large amount of real-time information in addition to the locations of mobile clients. The static hierarchical database structure may not be able to meet the real-time requirements of the systems since the number of mobile clients within a cell can be highly dynamic. The total delay in retrieving the real-time information will be highly unpredictable. Currently, we are modifying the static hierarchical tree structure to accommodate the dynamic mobile workload properties in a cell. The next step is to study how the proposed location update generation method can be applied to the new database architecture.

References:

[1] Evaggelia, P., George, S. Locating objects in mobile computing. IEEE Transactions on Knowledge and Data Engineering, 2001,13(4):571~592.

[2] Sajak, K.D., Sanjoy, K.S. Adaptive location prediction strategies based on a hierarchical network model in a cellular mobile environment. The Computer Journal, 1999,42(6):473~486.

[3] Xie, H., Tabbane, S., Goodman, D. Dynamic location area management and performance analysis. In: Proceedings of the 43rd IEEE Vehicular Technology Conference. Seacaucus, NJ, 1993. 536~539.

[4] Plassmann, D. Location management strategies for mobile cellular networks of 3rd generation. In: Proceedings of the 44th IEEE Vehicular Technology Conference. Stockholm, 1994. 649~653.

[5] Rose, C. Minimizing the average cost of paging and registration: a timer-based method. ACM/Baltzer Journal of Wireless Networks, 1996,2(2):109~116.

[6] Rokitansky, C.H. Knowledge-Based routing strategies for large mobile networks with rapidly changing topology. In: Proceedings of the ICCC'90. New Delhi, India, 1990.

[7] Akyildiz, I., Ho, J. On location management for personal communication networks. IEEE Communications, 1996,34(9):138~145.

[8] Jannink, J., Lam, D., Shivakumar, N., *et al*. Efficient and flexible location management techniques for wireless communication systems. Wireless Network, 1997,3(5):361~374.

[9] Rose, C., Yates, R. Minimizing the average cost of paging under delay constraints. ACM/Baltzer Journal of Wireless Networks, 1996,2(3):109~116.

[10] Guang, Wan, Eric, Lin. Cost reduction in location management using semi-realtime movement information. ACM/Baltzer Journal of Wireless Networks, 1999,5(5):245~256.

[11] Hanoch, L., Zohar, N. Active tracking: locating mobile users in personal communication service networks. ACM/Baltzer Journal of Wireless Networks, 1999,5(6):467~477.

[12] Pitoura, E., Fudos, I. An efficient hierarchical scheme for locating highly mobile users. In: Proceedings of the 6th ACM International Conference on Information and Knowledge Management (CIKM98). Bethesda, Maryland, 1998. 218~225.

[13] Mouly, M., Pautet, M.B. The GSM System for Mobile Communication, Cell and System. Louise Bruneau, F-91120, Palaiseau, France ISBN 2-9507190-0-7, 1992.

移动计算系统中的自适应位置更新策略

李国徽, 刘云生

(华中科学技术大学 计算机科学与技术学院,湖北 武汉 430074)

摘要: 移动计算系统中一个很重要的问题就是管理移动客户机的实时位置.在现有商业的移动计算系统中,采用一种两层的体系结构.这种两层的结构不具有可扩展性,因而不能适应具有大量移动用户的新型移动计算应用.人们提出了一种新的层次结构的位置数据库结构,系统中的位置数据库组成一个树形结构以方便移动用户位置查找.尽管这种结构把位置更新及查询的任务在系统中的所有位置数据库中进行了分摊,它也有其自身的弱点:位置更新的代价较大及位置查询的延迟较长.研究了树形位置数据库结构中的位置更新策略,给出了一个位置变更的阈值计算方法,以使得位置管理的代价最少.

关键词: 移动计算系统;移动对象的位置管理;位置更新

中图法分类号: TP393 文献标识码: A

****

第 12 届全国计算机辅助设计与图形学学术会议

征文通知(第 2 次)

由中国计算机学会 CAD 与 CG 专业委员会主办、信息产业部电子第十五研究所和贵州工业大学承办的第 12 届全国 CAD 与 CG 学术会议,定于 2002 年 8 月 20 日～24 日在贵阳召开.本次会议所征论文均按期刊要求和格式,经专家审定录用后,除在本次会议的会议录上刊登外,还将根据情况分别在《计算机辅助设计与图形学学报》、《计算机工程与应用》、《机械与电子》和《贵州工业大学学报》等专业刊物上公开出版.有关此次会议的内容请查看信息产业部电子第十五研究所网页.欢迎踊跃投稿和参加会议.

一、征文内容: 计算机图形学(图形算法与技术);智能 CAD 技术;人机接口技术;虚拟现实;CAD/CAM 技术与应用;计算机辅助几何设计;计算机动画、仿真技术;计算机支持的协同设计;容错技术;测试、诊断与可测试性设计;软件测试与软件可靠性;工程数据库与系统集成;科学计算可视化;多媒体技术和图象处理;电子设计自动化;CIMS;地理信息系统 GIS;计算机辅助工业设计;计算机辅助概念设计;机械、建筑、轻工及其他 CAD

二、重要日期

征文截止时间: 延至 2002 年 3 月 31 日(收到时间) 论文录取通知日期: 2002 年 4 月 30 日

论文正稿交付日期: 2002 年 5 月 31 日 会议时间: 2002 年 8 月 20 日～24 日

三、论文要求及说明

1.反映在 CAD/CG 及有关技术领域中的技术和应用成果;

2.未在其他刊物和会议上发表过;

3.每篇文稿不超过 5000 字(含图表);

4.论文初稿和正稿一律采用 Office 2000 格式的电子版交付;

5.论文录取通知通过 E-mail 发送.请作者务必提供可靠的 E-mail 地址及其他形式的通信地址;

6.作者自留底稿,恕不退稿.2002 年 4 月 30 日后未接到稿件录用通知,作者可自行处理.

四、联系方式: 100083 北京 619 信箱 24 分箱 杨晓燕 收

Email: kjw@nci.ac.cn 联系电话: 010-62325831 http://www.nci.ac.cn