Off-Line AGV Routing on the 2D Mesh Topology with Partial Permutation

Jianyang Zeng
Centre for Advanced Information Systems
School of Computer Engineering
Nanyang Technological University
Singapore 639798
Email: zengjy321@pmail.ntu.edu.sg

Wen-Jing Hsu
Centre for Advanced Information Systems
School of Computer Engineering
Nanyang Technological University
Singapore 639798
Email: hsu@ntu.edu.sg

***Abstract*— In this paper, we consider the routing issue for the AGV system in the Large Scalable Flexible Manufacturing System (LSFMS). After we study the existing mature techniques in the packet routing and analyze the differences between the AGV routing and packet routing model, especially on the mesh topology, we propose an off-line algorithm for AGV routing on the 2D mesh layout with partial permutation. The running time of our routing algorithm is $2n + o(n)$ steps for an $n \times n$ mesh layout, which is almost optimal in the worst case.**

I. Introduction

A substaintial amount of research has been done for Flexible Manufacturing Systems (FMS) [1], [4], [5], [8], such as task scheduling and vehicle routing issues. However, the research only focuses on small or manufacturing systems that have fixed given size, and they typically use techniques such as linear programming or operation research [5]. However, as the demand for throughput and productivity increases, the future FMS needs to be scalable. We call such a system Large Scalable Flexible Manufacturing System (LSFMS).

LSFMS usually consists of three parts: storage system (such as Automated Storage/Retrieval System), moving system and controlling system. Generally, the moving system of LSFMS is either conveyors or AGVs. In this paper, we only consider the AGVs as the moving system.

In order to move the products in the LSFMS quickly and without any conflict, developing efficient routing algorithm for the AGV system should be an important research topic in the LSFMS. In many LSFMS applications, the storage area is arranged into rectangular blocks, which leads to a mesh-like path topology. The mesh layout has many advantages, including easy scalability , good fault-tolerant ability, etc. So it has been popularly used in the current FMS.

There are many existing results about AGV routing algorithms on the mesh topology [2]–[8]. [2] and [3] gave the analysis of time and space complexities for some basic AGV routing operations on 2D-mesh topology. The upper bounds of time and space complexities for AGV routing are $\Theta(n^2)$ and $\Theta(n^3)$ respectively, where n denotes the number of stations in the path topology. However, the paper does not give the details of the routing algorithms and techniques to avoid congestion, conflicts, deadlocks, etc. [8] and [4] presented different methods to schedule and route simultaneously in an $n \times n$ mesh-like path topology. The algorithms can schedule and route simultaneously up to $4n^2$ AGVs concurrently at one time. In these papers, the routing process is formulated as a sorting problem. Although there are no conflicts during the permutation, it requires $3n$ steps of well-defined physical moves, which requires AGVs to travel extra distance and consume extra energy to finish the tasks.

For packet routing in interconnection networks, there exist lots of results on the mesh topology [10]–[12], [15]. For example, Valiant's algorithm [12] divided the routing into two phases, namely, from the source to a randomly selected intermediate node and from the intermediate node to the destination. Thompson and Kung gave routing algorithms based on Batcher's Bitonic sort [10], the running time of which is $6n + o(n)$. Schnorr and Shamir presented an upper bound of $3n + o(n)$ for sorting into row major order for the mesh topoloy [16]. Leighton proposed a deterministic algorithm solving the problem in $2n - 2$ steps [15], using constant size queues. The time bound is optimal in the worst case. Suel proposed a deterministic algorithm for packet routing with a running time of $2n + o(n)$ and with small queue size [11].

However, these results can not be applied to AGV routing directly, because there are some fundamental differences between the two routing models. By nature, the size of queue and buffer for AGV routing are much smaller than that in packet routing. Another difference is that in packet routing model, the packets can be injected into the network at any time, but in AGV routing, the total number of AGVs is *fixed*, and usually less than n^2 for an $n \times n$ mesh. In spite of these differences, the results in packet routing are useful for developing efficient routing algorithms for n^k (where $k < 2$) AGVs system.

In this paper, we try to adapt some useful techniques from the packet routing, and propose an off-line AGV routing algorithm on the $n \times n$ mesh topology with partial permutation. In our routing algorithm, AGV routing can be finished in $2n + o(n)$ steps on $n \times n$ mesh layout.

The remainder of the paper is organized as follows. Section 2 describes the mesh routing model and defines some notations that will be used in the analysis of the routing algorithm. Section 3 gives the details of the routing algorithm. In Section

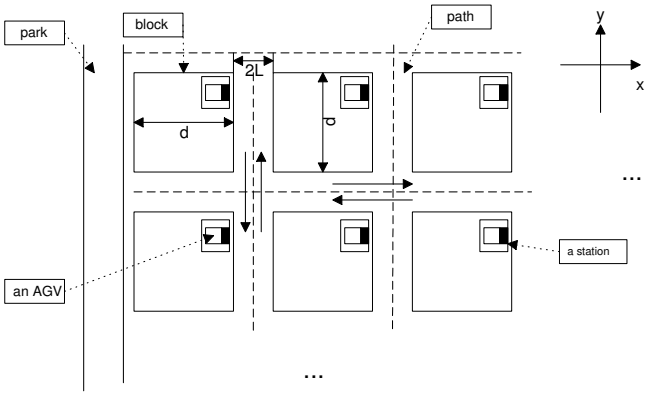

Fig. 1. Realistic mesh layout.

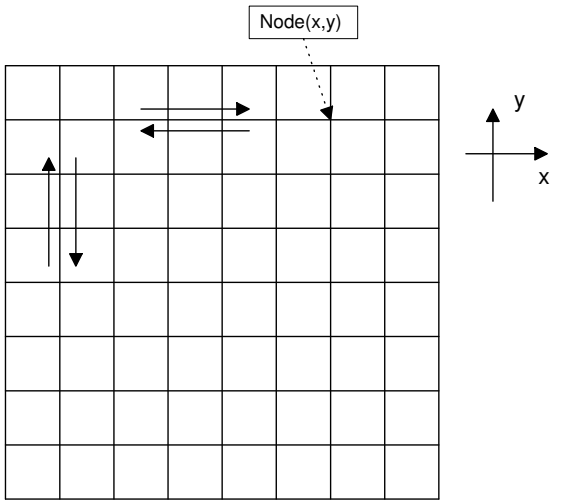

Fig. 2. Simplified mesh routing model.

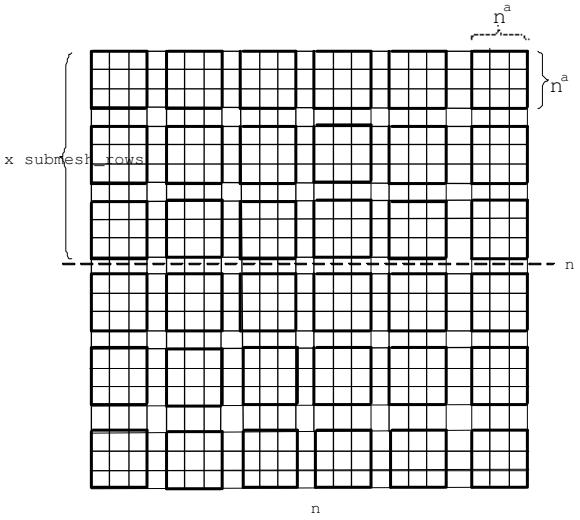

Fig. 3. The partition of mesh layout.

4, we analyze the correctness the time complexity of our routing algorithm. Finally, Section 5 discusses how to extend our routing algorithm by relaxing some of our assumptions, and points out issues for some future researches.

II. Mesh routing model

In order to describe the marrow of our routing process clearly, we begin with a simple but general model in which one yard block has only one station near an intersection of pathways (refer to Fig. 1). In this mesh layout, there are in total $n \times n$ blocks, namely n blocks in each column and n blocks in each row. All blocks have the same size. There are two paths with different directions between two adjacent blocks. Each Block has one station for Pick up or Drop off operation(or P/D station for short), located at the upper right corner of the block. On the upper left-hand side of the mesh, there is a vehicle park where all AGVs are stationed initially and to which they will return upon completion of all tasks.

Although there are some important details for AGV routing, such as the size of the junction, the radius of a 90° turn, the length of the AGV, etc. [4]–[8], it is reasonable and realistic for us to simplify the mesh model for convenience of analysis and discussion. In the simplified mesh layout model, shown in Fig. 2, there are n^2 junctions of pathways. A junction and the associated neighboring station are collectively regarded as a node. Each node is to assign it with the coordinates (x, y) as its address or ID, where x and y represent respectively the column and row IDs. This mesh layout is modeled by a graph $G = (V, E)$. The $n \times n$ vertices of the graph represent junction nodes, and the bi-directional edges represent two paths between two adjacent junction nodes, and the length of each edge is a constant. When an AGV move from its current station to one of its neighboring stations, we say that it finishes a "step". In each step, each path in one direction between two neighboring stations can only hold one AGV.

We divide the AGV movements into three phases. In the first phase, let AGVs set out from the park to their pick up stations. In the second phase, let AGVs pick up loads and travel to their destinations and drop-off loads. In the third phase, let AGVs return to the park from their drop-off stations. Because it is easy for us to dispatch the AGV moving without any conflict in the first phase and the third phase [17], we can focus only on the second phase when the loaded AGVs move on the mesh layout.

We assume that when an AGV reaches its destination, it enters the buffer and leaves the mesh grid.

We divide the mesh layout into $n^\alpha \times n^\alpha$ submeshes, where $0 \leq \alpha \leq 1$. There are in total $n^{2(1-\alpha)}$ submeshes. Each submesh is assigned by the coordinate (I, J) as its address, where I and J represent the column and row position of the submesh, and are called submesh_row and submesh_column respectively.

At the same time, we partition the mesh into several groups. Each group includes x submesh_rows of the submeshes, as shown in Fig. 3 (the value of x will be determined later).

In our routing model, we assume that the movement pattern is a partial permutation, namely each node can be origin or

destination of one AGV, and the total number of AGVs is n^k, where $0 \leq k \leq 2$. Meanwhile, we also assume that the buffer of each node is three, which means that in each step, each node can only accomodate at most three AGVs.

Based on the routing model, we formally define the following notations.

Firstly, the partial permutation is defined as follows.

$$\sum_{n \times n} = \{\sigma | \sigma : \mathcal{Z}_n \times \mathcal{Z}_n, \sigma \ is \ 1-1, |\sigma| < n^2\}.$$

We also define the following set for the convenience of analysis.

$M_{(I,J)} = \{$The AGVs in submesh $(I,J)\}$;
$M^h_{(I,J)} = \{$The AGVs which are in submesh (I,J) and their destination submesh_columns are in submesh_column $h\}$;
$M_{(*,J)} = \{$The AGVs in submesh_row $J\}$;
$M_{(I,*)} = \{$The AGVs in submesh_column $I\}$;
$M^h_{(*,J)} = \{$The AGVs which are in submesh_row J and their destination submesh_columns are in submesh_column $h\}$;
$M^h_{(I,*)} = \{$The AGVs which are in submesh_column I and their destination submesh_columns are in submesh_column $h\}$;
$|S|$: the cardinality of the set S.

Similar to [11], we also define the following sorted order, which will be used in our routing algorithm later. $\mathcal{I}$ is a bijection from $[n] \times [n]$ to $[n^2]$, for an $n \times n$ mesh. Suppose that $\mathcal{I}(i,j) = k$, when

$$\mathcal{I}(i_1, j_1) < \mathcal{I}(i_2, j_2) \Leftrightarrow [(i_1 < i_2) \wedge (j_1 = j_2)] \vee (j_1 < j_2).$$

We say that the AGVs are sorted in k *major by row order*. Similarly, we can also formally define sorting in k *major by column order*.

III. Routing algorithm

Although our routing algorithm is off-line and deterministic, it originated from the studying of the randomized algorithm for packet routing on fixed-connection topology [11], [12]. Most of these randomized algorithm are the variants of Valiant's algorithm in [12] which divides the routing into two phases. In the first phase, the packets are randomly distributed over global network. In the second phase, the packets are routed to their destinations. Following the same principle, we firstly distribute the AGVs evenly over each group region. The reason why the group region is enough for distributing the AGVs is that we only consider the partial permutation in our routing model, in which the number of AGVs is less than n^2. Secondly, we route the AGVs to their approximate destination submesh. Finally, we route all AGVs to their final destinations within each local submesh. In more details, our routing algorithm consists of 8 steps.

Step 1 Within each group, we rename each AGV so that it can be referred to as the first destination submesh in each submesh_column. Let us firstly consider the first group. We rename the AGVs in $M^1_{(I,1)}$ from 1 to $|M^1_{(I,1)}|$, and rename AGVs in $M^1_{(I,2)}$ from $|M^1_{(I,1)}|+1$ to $|M^1_{(I,1)}|+|M^1_{(I,2)}|$, and so on. Generally for the AGVs in $M^1_{(I,J)}$, they are renamed with numbers from $\sum_{i=1}^{J-1}|M^1_{(I,i)}|+1$ to $\sum_{i=1}^{J}|M^1_{(I,i)}|$, where $|M^1_{(I,0)}| = 0$ and $1 \leq J \leq x$. Next, for the AGVs in $M^2_{(I,J)}$, they are renamed from $\sum_{i=1}^{x}|M^1_{(I,i)}| + \sum_{i=1}^{J-1}|M^2_{(I,i)}| + 1$ to $\sum_{i=1}^{x}|M^1_{(I,i)}| + \sum_{i=1}^{J}|M^2_{(I,i)}|$, where $|M^2_{(I,0)}| = 0$ and $1 \leq J \leq x$. More generally, for the AGVs in $M^h_{(I,J)}$, where $1 \leq J \leq x$, they are renamed from $\sum_{j=1}^{h-1}\sum_{i=1}^{x}|M^j_{(I,i)}| + \sum_{i=1}^{J-1}|M^h_{(I,i)}| + 1$ to $\sum_{j=1}^{h-1}\sum_{i=1}^{x}|M^j_{(I,i)}| + \sum_{i=1}^{J}|M^h_{(I,i)}|$. We follow the same scheme for each submesh_column I, and similarly for each group. Suppose after the renaming operation, each AGV is renamed with a number N_i. Denote by D_i the submesh_row that the AGV wants to move into along the same column_submesh, we let

$$D_i = (N_i - 1) \bmod x + 1.$$

Step 2 Sort the AGVs in each submesh in D_i *major by row order*, using the same method as that in [4], [8].

Step 3 Route each AGV to the submesh D_i along the column. After step 2, the AGVs that want to enter submesh D_i have been distributed almost evenly along the row direction. Thus, it can be sure that not too many AGVs enter the submesh across the same edge. However, within one submesh, there still exists a case that one column has one more than the average number of AGVs moving to the submesh D_i. So in the worst case, there is one column having x AGVs than the average number of AGVs moving to D_i in each column. Here we use the same scheme, called the counter scheme, used in [11]. In each row of submesh D_i, we maintain one counter for each row, initially set to zero. Suppose that $C_1, C_2, ..., C_{n^\alpha}$ are the counters for the row $1, 2, ..., n^\alpha$ of the submesh D_i, respectively. When one AGV enters the submesh D_i, it detects whether C_i is less than n^α (detecting order is from 1 to n^α). If so, it moves to the row i along the column and let $C_i = C_i + 1$. If the station there is empty, it stays there, otherwise, it moves to another empty station along the row.

Although in the worst case, one AGV moving to the row direction may meet two AGVs coming from both directions of the column, this problem can be solved easily since we have the buffer of size three in every station.

Step 4 Sort the AGVs in each submesh in *destination submesh_column major by column order*, using the same method in [4], [8].

Step 5 Route each AGV to the submesh on the destination column along the row. The method is similar to that in step 3.

Step 6 Sort the AGVs in each submesh in *destination submesh_row* major by *row order*, using the same method as that in [4], [8].

Step 7 Route each AGV to the submesh on the destination row along the column, using the same method as that in step 3 and step 5.

Step 8 Sort all AGVs in each submesh to their destinations, using the same method as that in [4], [8].

In our routing algorithm, the effect achieved from step 1 to step 3 is the same as the first phase of Valiant's algorithm,

namely to distribute the AGVs evenly over the region of each group. Step 4–step 8 is equal to the second phase of Valiant's algorithm. But with more complication, we firstly route the AGVs to their approximate destination in a submesh from step 4 to step 7. Finally, we perform the local routing within each submesh to route each AGV to its final destination.

The routing algorithm is simple and need only local control mechanism in each submesh. Therefore, it is easy to implement.

IV. Analysis of the routing algorithm

In order to argue for the correctness of the routing algorithm and analyze its time complexity, we give the following lemmas and theorems.

Lemma 4.1: *After step 2 of the routing algorithm, the number of the AGVs in each submesh is not more than* $n^{2\alpha}$.

Proof: The renaming scheme can be also expressed in the following way. The AGVs with the same destination submesh_column in a submesh_column within a group, are allocated evenly to each submesh. However, the number of AGVs with the same destination submesh_column can not always be an integral multiple of x. Since the AGVs with the next destination submesh_column is allocated in the submesh_column to make sure that the number of the AGVs in each submesh after step 3 is kept the same. Therefore, the average number of the AGVs in each submesh is not more than $n^{2\alpha}$. ∎

Lemma 4.2: *Suppose that after step 3, the number of AGVs with destination subemesh_column* h *in submesh* (I, J_1) *and* (I, J_2) *are* $|M^h_{(I,J_1)}|$ *and* $|M^h_{(I,J_2)}|$ *respectively, where* $J_1 \neq J_2$. *We have* $||M^h_{(I,J_1)}| - |M^h_{(I,J_2)}|| \leq 1$.

Proof: Omitted. ∎

Lemma 4.3: *Suppose that after step 3, the number of AGVs with destination subemesh_column* h *in submesh_row* J_1 *and* J_2 *are* $|M^h_{(*,J_1)}|$ *and* $|M^h_{(*,J_2)}|$ *respectively, where* $J_1 \neq J_2$. *We have* $||M^h_{(*,J_1)}| - |M^h_{(*,J_2)}|| \leq n^{1-\alpha}$.

Proof: Since there are in total $n^{1-\alpha}$ submesh_columns, and according to Lemma 4.2, it is easy to get the proof. ∎

Lemma 4.4: *Suppose that* $x \geq \frac{n^k}{n^{2\alpha}-n^{1-\alpha}}$, *where* $\alpha > \frac{1}{3}$. *After step 5, the number of AGVs in each submesh is not more than* $n^{2\alpha}$.

Proof: Since there are at most n^k AGVs, in the worst case, suppose that they are destined in the same submesh_column. According to Lemma 4.3, we know that after step 5, the number of AGVs in a submesh_column is at most $n^{1-\alpha}$ greater than the average submesh_row number of AGVs in the group, namely, we have $|M^h_{(*,J)}| \leq \frac{n^k}{x} + n^{1-\alpha}$. Substituting the value of x in the inequality, we can get that $|M^h_{(*,J)}| \leq n^{2\alpha}$. ∎

Theorem 4.1: *Our algorithm works correctly to route each AGV to its destination, when the following is satisfied*

$$\begin{cases} \alpha > \frac{1}{3} \\ n^k \leq n^{1+\alpha} - n^{2(1-\alpha)} \\ x \geq \frac{n^k}{n^{2\alpha}-n^{1-\alpha}} \end{cases}$$

Proof: Lemma 4.1 and 4.4 guarantee that our routing algorithm work smoothly when we "shuffle" the AGVs from one submesh to another one. Since the size of the group x is decided by n^k, the number of AGVs in the mesh layout and the parameter α we choose, we still need to make sure that $x \cdot n^\alpha \leq n$, because there are at most n stations in one column. This constraint can be satisfied if the above conditions are required. Therefore, we get the proof. ∎

Theorem 4.2: *The running time* T *of our routing algorithm is* $2n + \frac{n^{k+\alpha}}{n^{2\alpha}-n^{1-\alpha}} + 15n^\alpha$ *steps.*

Proof: In our routing algorithm, steps 2, 4, 6 and 8 take $3n^\alpha$ steps each, according to [8] [4]. Steps 3 takes $x \cdot n^\alpha + n^\alpha$. Steps 5 and 7 take $n + n^\alpha$ steps each. Therefore, $T = 2n + x \cdot n^\alpha + 15n^\alpha$. Substituting $x = \frac{n^k}{n^{2\alpha}-n^{1-\alpha}}$, and we have $T = 2n + \frac{n^{k+\alpha}}{n^{2\alpha}-n^{1-\alpha}} + 15n^\alpha$. ∎

Theorem 4.3: *When* $\alpha > k - 1$ *and the relation between* α *and* k *satisfies the inequality in Theorem 4.1, the running time* T *of our routing algorithm is* $2n + o(n)$ *steps.*

Proof: Since $\alpha > k - 1$, we have $n^{k+\alpha} < n^{2\alpha+1}$, Thus,

$$\frac{n^{k+\alpha}}{n^{2\alpha}-n^{1-\alpha}} < \frac{n^{2\alpha+1}}{n^{2\alpha}-n^{1-\alpha}}.$$

Because $\alpha > \frac{1}{3}$, we have

$$\frac{n^{2\alpha+1}}{n^{2\alpha}-n^{1-\alpha}} < \frac{2n^{2\alpha+1}}{n^{2\alpha}} = 2n.$$

Thus,

$$\frac{n^{k+\alpha}}{n^{2\alpha}-n^{1-\alpha}} = o(n).$$

Therefore, $T = 2n + o(n) + 15n^\alpha = 2n + o(n)$. ∎

Theorem 4.4: *For an* $n \times n$ *mesh layout with* n^k *AGVs, as long as* $n^k = n^{2-\epsilon}$, *where* $\epsilon > 0$, *there exists an off-line routing algorithm with the running time of* $2n + o(n)$.

Proof: From the Theorem 4.3, we know that if $k < 2$, we can always find a value of α, which satisfies $0 < \alpha \leq 1$ and $\alpha > k - 1$, to finish the AGV routing in $2n + o(n)$ steps. ∎

V. Discussions and conclusions

In this paper, we try to find the differences between the moving system of the Large Scalable Flexible Manufacturing System (LSFMS)—AGV system, and the packet routing model. After analyzing some packet routing algorithm, we propose an off-line algorithm for AGV routing on the 2D mesh topology with partial permutation. Its time complexity is less than n^2 for $n \times n$ mesh layout. Similar to [15], the time complexity is almost optimal in the worst case.

Although our algorithm is used only for AGV routing on the mesh topology, it can be easily applied to packet routing with slight changes, because the packet routing can use larger buffer to implement it. It has the significance since although lots of research have been done on the 1-1 packet routing, but few works deal with the partial permutation, and the relation between the size of permutation and the time complexity in the routing algorithm.

Our routing algorithm only considers the AGV routing on the 2D mesh layout, but it is easy for us to extend it to the multi-dimensional mesh or other related topologies. For example, torus and 3D mesh has been popularly used in the fixed-connected networks. In some Large Scalable Flexible Manufacturing System, for example, the Automated Storage and Retrieval System (AS/RS), usually the moving system is designed based on the 3D mesh layout. We can use the same principle to design a similar efficient routing algorithm for such a 3D mesh AS/RS.

The buffer size on our mesh layout is three. However, the buffer is only used during step 3, 5 and 7, when there is a slight imbalance of the number of the AGVs among each column (or rows). However, we can even have smaller size of buffer if we have more complicated controlling system. For the controlling system, another thing is that if the local submesh is larger, the controlling price is also higher. So we should try to minimize the size of the submesh n^α, as long as the high time complexity is satisfied.

In [9], a random routing algorithm in the multi robot system is presented. But this random algorithm may lead to deadlocks for multi robot routing. From the details of our routing algorithm, we know that it is deadlock-free for all AGVs.

There are still many open issues for future study. Firstly, in our routing model, we only consider 1-1 routing, namely permutation, in our routing algorithm. But one-to-many or many-to-many routing is more important since they are more common in realistic AGV systems. These cases need more complicated routing algorithm since we have to consider the order for different AGVs to reach the same station. Secondly, our routing algorithm is off-line, and need the global information of the permutation. When the communication mechanism is not good enough to get such a global information, we have to rely on the on-line routing algorithm. Thirdly, our routing model ignores some other differences between the realistic AGV routing and packet routing. In realistic AGV routing, we have to consider the turns made by all AGVs, since they are related to the energy requirement for the AGV routing system. The speed of the AGVs should be another important issue for the AGV routing. Finally, in our study, we did not consider the case when some AGVs break down, or when the communication system malfunctions. A single blockage will cause the failure of the entire system. Therefore, it is important to consider fault-tolerant strategies.

ACKNOWLEDGMENT

We acknowledge the Maritime and Port Authority, A*STAR and Nanyang Technological University, all of Singapore, for their support of the research project.

REFERENCES

[1] J. M. A. Tanchoco. *Material flow systems in manufacturing*. Chapman and Hall, 1994.

[2] W.-J. Hsu and S.-Y. Huang. Route planning of automated guided vehicles. *Proceedings of Intelligent Vehicles*, Paris, pp.479-485.

[3] S.-Y. Huang and W.-J. Hsu. Routing automated guided vehicles on mesh like topologies. *Proceedings of International Conference on Automation*, Robotics and Computer Vision, 1994.

[4] L. Qiu. *Scheduling and Routing of Automated Guided Vehicles*. PhD thesis, School of Computer Engineering, Nanyang Technological University, Singapore, 2002.

[5] L. Qiu, W.-J. Hsu, S.-Y. Huang, and H. Wang. *Scheduling and Routing Algorithms for AGVs: a Survey*. International Journal of Production Research, Vol. 40, No. 3, pp. 745-760, 2002.

[6] L. Qiu and W.-J. Hsu. Routing AGVs on a Mesh-like Path Topology. *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (IVS 2000)*, pp. 392-397, Dearborn, Michigan, USA, Oct. 3-5, 2000.

[7] L. Qiu and W.-J. Hsu. Algorithms for Routing AGVs on a Mesh Topology. *Proceedings of the 2000 European Conference on Parallel Computing (Euro-par 2000)*, pp. 595-599, Technical University of Munich, Munich, Germany, Aug. 29-Sep. 1, 2000.

[8] L. Qiu, and W.-J. Hsu. An Algorithm for Concurrent Routing of AGVs in a Mesh. *Proceedings of the 7th Australasian Conference on Parallel and Real-Time Systems (PART 2000)*, pp. 202-214, University of New South Wales, Sydney, Australia, Nov. 29-30, 2000.

[9] S. Preminger. *Complexity analysis of movement in multi robot system*. Master's thesis, Department of Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 1995.

[10] C. D. Thompson and H. T. Kung. Sorting on a mesh-connected parallel computer. *Communicaitons of the ACM*, 20:263-271, 1977.

[11] T. Suel. *Routing and Sorting on Fixed Topologies*. PhD thesis, Faculty of the Graduate School, University of Tesas at Austin,USA, 1994.

[12] L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*,11:350-361, 1982.

[13] F. T. Leighton. *Intorduction to Parallel Algortihms and Architectures: Arrays, Trees, and Hypercubes*. Morgan_kaufmann, San Mateo, CA, 1991.

[14] T. Cormen, C. Leiserson, R. Rivest and C. Stein. *Intorduction to Algorithm*. The MIT Press, USA, 2001.

[15] F. T. Leighton, F. Makedon, and I. G. Tollis. A $2n-2$ step algorithm for routing in an $n \times n$ array with constant queue sizes. *Proceedings of the 1st Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 328-335, July 1989.

[16] C. P. Schnorr and A. Shamir. An optimal sorting algorithm for mesh-connected computers. *Proceedings of the 18th ACM Symposium on Theory of Computing*, pages 255-263, May, 1986.

[17] J. Zeng, W.-J. Hsu and L. Qiu, An Energy-Efficient Algorithm For Conflict-Free AGV Routing On A Linear Path Layout. *Proceedings of the International Computer Symposium 2002(ICS 2002)*, Vol. 1, pp 816-825, Hualian, Taiwan, Dec. 18-21, 2002.