Algorithms for Routing AGVs on a Mesh Topology

Ling Qiu and Wen-Jing Hsu

Centre for Advanced Information Systems, School of Applied Science
Nanyang Technological University, Singapore 639798
{P146077466, hsu}@ntu.edu.sg

Abstract: This paper proposes to adapt parallel sorting/message routing algorithms to route Automated Guided Vehicles (or AGVs for short) on mesh-like path topologies. Our strategy guarantees that no conflicts will occur among AGVs when moving towards their destinations; a high degree of concurrency can be achieved during our routing process.

1. Introduction

From a computer science perspective, *Automated Guided Vehicle*s (or AGVs for short) systems are intrinsically parallel and distributed systems that require a high degree of concurrency. Many algorithms for scheduling and routing of AGVs have been proposed, however, most of them are applicable to systems with a small number of AGVs, offering only low degree of concurrency [1]. With drastically increased number of AGVs in recent applications (e.g. in the order of a hundred in a container terminal), efficient scheduling and routing algorithms are needed to resolve the increased contention of resources (e.g. path, loading and unloading buffers) among AGVs. Because they often employ regular route topologies, the new applications also demand innovative strategies to increase system performance.

In this paper, we apply ideas arising from parallel processing and adapt sorting algorithms to route AGVs concurrently on a mesh path. Based on our routing strategy, all AGVs can reach their destinations within $O(n \log n)$ steps of well-defined physical moves in an $n \times n$ mesh. No conflicts, congestion, livelocks or deadlocks among the AGVs will occur, and a very high degree of concurrency can be achieved.

The remaining part of the paper is organized as follows. We describe the problem in Section 2. Section 3 gives the routing algorithm. Section 4 concludes the paper.

2. The Problem

Many AGV applications employ regular path topologies, such as mesh. As a case in point, presently at Nanyang Technological University, Singapore, the application of AGVs in a container handling system is being studied [1 – 4]. The main goal is to schedule and route AGVs within a mesh-like path and container stacking areas (as shown in Fig. 1). In a container port, an AGV could originate from a location near one of the container cranes at a ship, and have a destination at a yard area; similarly, an AGV could also reverse the direction of its trip, i.e., start from a yard location and

A. Bode et al. (Eds.): Euro-Par 2000, LNCS 1900, pp. 595-599, 2000.

end up near a crane (refer to Fig. 1). It is also possible for an AGV to move from a yard area to another. Based on this reality, we formulate the problem as shown in Fig. 2 which is practical and commonly encountered: AGVs are assumed to originate from arbitrary buffers and supposed to end up at the others, how to efficiently route all AGVs such that they can reach their destinations without conflicts? However, to clearly explain the main ideas of our routing process, we will start from an essential scenario as shown in Fig. 3. In the model, all AGVs start from the intersection of the pathways (referred to as "nodes" subsequently) and end their journeys also at the nodes of the mesh. We will first demonstrate how to adapt the principle of *bitonic sorting* to this model, and then extend the result to our primitive problem.

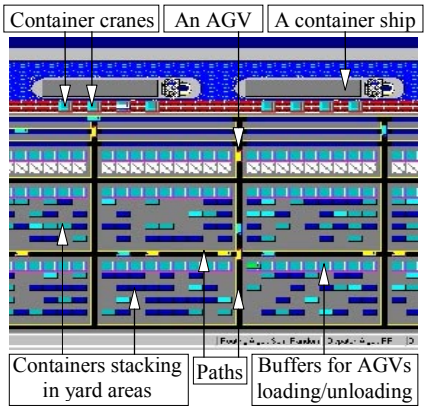

Figure 1. The topology of a container port

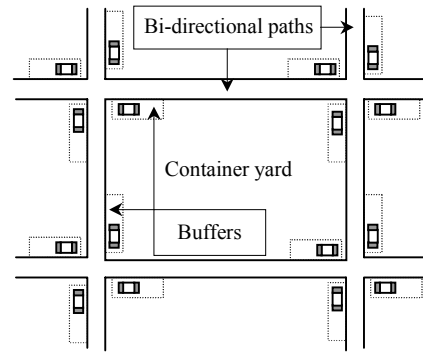

Figure 2. The problem formulation

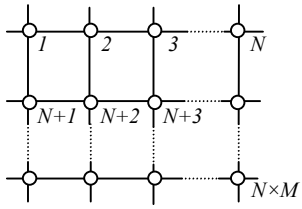

Figure 3. The essential model

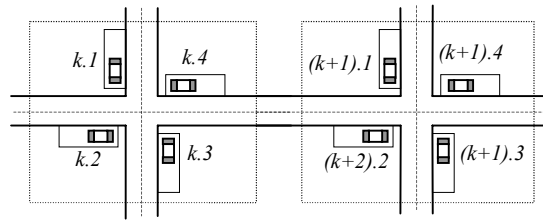

Figure 4. Extended nodes and the numbering of buffers

3. The Routing Strategy

3.1 Routing among Nodes

Referring to Fig. 3, assume that the path topology considered is a mesh of N columns by M ($M = 2^k$) rows. Thus it has $M \times N$ nodes in total, which are numbered from *1* to $M \times N$ in a row fashion. We have $M \times N$ AGVs initially stationed at each node. Every AGV has a unique destination (different from the others). With the numbering of nodes, it should be clear that the routing of the AGVs amounts to sorting their

destination IDs, with the data exchanges corresponding to the moves of AGVs. The major difference is, here we must ensure that the physical moves of AGVs are free of hazards like collisions; the moves may also be constrained in terms of the spatial constraints. The following four steps give a detailed description of the routing algorithm. Readers are referred to [2] for a detailed illustration of the algorithm.

Step 1: Route AGVs in every row concurrently, so that for odd rows AGVs are sorted into increasing order, and for even rows decreasing.

Step 2: Apply *bitonic merging* to route AGVs on row *4i-3* to row *4i*, where $i = 1, 2, ..., M/4$. Finally, AGVs on row *4i-3* and row *4i-2* are sorted into increasing order, while AGVs on row *4i-1* and row *4i* decreasing order.

In this phase, vehicles first move vertically between row *4i-3* and row *4i-2* or row *4i-1* and row *4i*, then move horizontally within each row.

Step 3: Apply *bitonic merging* to route AGVs on row *8i-7* to row *8i*, where $i = 1, 2, ..., M/8$, we get an increasing sequence of AGVs from row *8i-7* to row *8i-4* and a decreasing one from row *8i-3* to row *8i*.

Step 4: Apply operations similar to *Step 3* to route AGVs on row $2^j i - (2^j - 1)$ to row $2^j i$ repeatedly, for $j = 4, 5, ..., k$ and $i=1, 2, ..., 2^{k-j}$ where $M = 2^k$. After this step, every AGV gets to its final destination.

3.2 Routing among Extended Nodes

Now return to the primitive problem as shown in Fig. 2, in which four AGVs initially station at the buffers near a node. In order to apply the previous routing algorithm, we define an *extended node* as a node together with the four associated buffers nearby (cf. the dotted rectangles as shown in Fig. 4). Every buffer is also numbered in the form of *x.y*, where *x* is the ID of the node while *y* is the ID of the buffer in the extended node (cf. Fig. 4). We also assume that every pathway consists of two bi-directional lanes, and that every AGV has distinct destination from the other. If we define that (x.y > u.v) holds iif (x > u) or (x = u and y > v), our routing algorithm can be applied directly to handle this case without any revision. The only difference is that now the number of AGVs is four times as much as the previous one.

Claim 1: Routed by our algorithm, every AGV can reach it destination after limited steps of physical moves. ■

Claim 2: Applying our routing algorithm, no conflicts, congestion or deadlocks will arise amongst AGVs during the routing process. ■

Readers are referred to [2 – 4] for the detailed proofs of both claims.

3.3 Complexity of Concurrent Moves

Now let us analyze the time requirement for AGVs to reach their destinations. For this purpose, assume that the vertical and horizontal path segments are of the *same* length. Moreover, mechanically speaking, making a 90^0-turn (to change from horizontal direction to vertical direction or vice versa) is usually more expensive (in terms of the time, space and energy required) than moving in straight lines. Therefore, we will also analyze the number of 90^0-turns needed in the algorithm.

Definition: A *rectilinear step* is the move required for an AGV to move from a buffer in an extended node to another buffer in an adjacent extended node, exclusive of all 90^0-turns.

Claim 3: Applying our routing algorithm to the scenario shown in Fig. 4, the total amount of all concurrent rectilinear steps for all AGVs to reach their destinations is upper-bounded by $O(M + kN)$ or $O(Max\{M, N \log M\})$. ■

Theorem 1: Given arbitrary initial configurations as on an $n \times n$ mesh as described in Fig. 2 and Fig. 4, all AGVs will be able to reach their destinations using $O(n \log n)$ concurrent rectilinear steps and $O(\log n)$ concurrent 90^0-turns. ■

Readers are referred to [2, 4] for the detailed proofs of the claim and the theorem.

4 Discussions & Conclusions

This paper has proposed to adapt parallel sorting algorithm to route AGVs on a mesh path topology. Routed by the routing algorithm, all AGVs can reach their destinations without conflicts and deadlocks. A high degree of concurrency is also achieved.

In Section 3, we assume that there are two bi-directional lanes in a path. However, even if there is only one lane, the routing algorithm still works. The only difference is that in this case every process of swapping has to be finished in two phases, each of which allows AGVs to move in one direction. Hence the total number of concurrent steps of moves is doubled, but the complexity order remains the same. Usually, the cost of space resource is much lower than that of an AGV that can cost as much as a million dollars per vehicle. From this perspective, it is worthy of obtaining a higher system efficiency and AGV utilization by dedicating space for lanes or buffers. The trade-off among space, cost and time (measured by steps of physical moves) is discussed in detail in [2, 4]. Moreover, if the number of AGVs in an extended node is less than four or in other words, some of the buffers do not have AGVs initially, we regard the vacancies as *virtual* AGVs. Virtual AGVs are numbered with the maximum destination IDs in the extended node. Thus algorithm still works.

Before each step of routing process is carried out, the system has the idea of global traffic information upon which the routing decision is made. From this perspective, if we use the centralized control mechanism, every AGV simply follows the instructions sent by the central controller. Whereas under the decentralized control mechanism, the central controller decides when AGVs begin to compare and swap; while the local controllers of AGVs communicate with each other and coordinate their moves.

For further study, one direction is to relax on the constraints of the problem, in which case, a proper scheduling may be required. For instance, we need scheduling when the destinations of AGVs are not all distinct; similarly, when AGVs have continuous tasks, how to schedule and route them [1]? These outstanding problems have obvious applications.

5. References

1 Qiu, L. and Hsu, W.-J., Scheduling and Routing Algorithms for AGVs: a Survey. *Technical report: CAIS-TR-99-26, Centre for Adv. Info. Sys., Schl. of Applied Science., Nanyang Tech. Univ., Singapore, Oct 1999.* Available at http://www.cais.ntu.edu.sg:8000/.

2 Qiu, L. and Hsu, W.-J., Adapting Sorting Algorithms for Routing AGVs on a Mesh-like Path Topology. *Tech. report: CAIS-TR-00-28, Centre for Adv. Info. Sys., Schl. of Applied Sci., Nanyang Tech. Univ., S'pore, Feb 2000*. Available at http://www.cais.ntu.edu.sg:8000/.

3 Qiu, L. and Hsu, W.-J., Conflict-free AGV Routing in a Bi-directional Path Layout. *Proc. of 5th Int'l Conf. on Comput. Integrated Manu., Singapore, Mar 2000, pp. 392-403.*

4 Qiu, L. and Hsu, W.-J., Routing AGVs by Sorting. Accepted for *2000 Int'l Conf. on Para. and Dist. Processing Tech. and App. (PDPTA'2000), Las Vegas, USA, Jun 26-29, 2000.*