An Efficient Extension of Network Simplex Algorithm

Hassan Rashidi[a], Edward P. K. Tsang[b]

[a]*School of Computer Science and Electronic Systems Engineering, University of Essex, Colchester CO4 3SQ, U.K., Email: hrashi@essex.ac.uk*
[b]*School of Computer Science and Electronic Systems Engineering University of Essex, Colchester CO4 3SQ, U.K., Email: edward@essex.ac.uk, Tel: +44 1206 872774*

Abstract:
In this paper, an efficient extension of network simplex algorithm is presented. In static scheduling problem, where there is no change in situation, the challenge is that the large problems can be solved in a short time. In this paper, the Static Scheduling problem of Automated Guided Vehicles in container terminal is solved by Network Simplex Algorithm (NSA) and NSA+, which extended the standard NSA. The algorithms are based on graph model and their performances are at least 100 times faster than traditional simplex algorithm for Linear Programs. Many random data are generated and fed to the model for 50 vehicles. We compared results of NSA and NSA+ for the static automated vehicle scheduling problem. The results show that NSA+ is significantly more efficient than NSA. It is found that, in practice, NSA and NSA+ take polynomial time to solve problems in this application.

Keywords: Scheduling, Container Terminals, Minimum Cost Flow Problem, Network Simplex Algorithm, Optimization Methods.

1 Introduction

The Minimum Cost Flow (MCF) problem is the problem of flowing resources from a set of supply nodes, through the arcs of a network, to a set of demand nodes at minimum total cost, without violating the lower and upper bounds on flows through the arcs (which represent the capacities of the arcs). This problem arises in a large number of industries, including agriculture, communications, defence, education, energy, health care, manufacturing, medicine, retailing, and transportation [16]. This paper has been motivated by a need to schedule Automated Guided Vehicles (AGVs) in container terminals. The container terminals components that are relevant to our problem include quay cranes (QC), container storage areas, rubber tyred gantry crane (RTGC) or yard crane, and a road network [26][24]. A transportation requirement in a port is described by a set of jobs, each of which being characterized by the source location of a container, the destination location and its pick up or drop-off times on the quay side by the quay crane. Given a number of AGVs and their availability, the task is to schedule the AGVs to meet the transportation requirements.

Network Simplex Algorithm (NSA) is the fastest algorithm to tackle the MCF model[16]. Pricing scheme is certainly an important step in NSA since the total computational effort to solve a problem heavily depends on its choice. This step does two things. It checks whether the optimality conditions for the non-basic arcs are satisfied, and if not it selects a violated arc to enter the spanning tree structure [16]. The selected arc has a potential of improving the current solution. According to the theory [16] the NSA terminates in a finite number of iterations regardless of which profitable candidate is chosen if degeneracy is treated properly. Some well-known schemes in NSA are *the steepest edge scheme* (by Goldfarb and Reid [39]), the *Mulvey's list* (by Mulvey [39]), the *block pricing scheme* (by Grigoriadis [8]), the *BBG Queue pricing scheme* (by Bradley, Brown and Graves [39]), the *clustering technique* (by Eppstein [6]), the *multiple pricing schemes* (by Lobel [23]), the *general pricing scheme* (by Istvan [21]). In this paper we present a new pricing scheme, which significantly reduces the CPU-time required to tackle the MCF model. By using the new pricing scheme, we obtain an efficient extension of NSA, which called Network Simplex plus Algorithm (NSA+).

The structure of this paper is as follows. Section 2 reviews the scheduling problem of Automated Guided Vehicles (AGV) in container terminals. Section 3 presents two algorithms to tackle the MCF-AGV model, namely Network Simplex Algorithm (NSA) and Network Simplex plus Algorithm (NSA+). Experimental results from applying the two algorithms to tackle the model are compared in Section 4. Section 5 is considered to summary and conclusion.

2 The scheduling problem of Automated Guided Vehicles (AGV) in Container Terminals

In order to test that the new extension of Network Simplex Algorithm is efficient, we choose the most challenging problem in container port. The problem is the AGV scheduling problem in the container terminals and it is the same as the problem presented in [13]. Here, we have an overview on the problem. For more detail, readers can refer to [42]. The most important reason to choosing this problem is that the efficiency of a container terminal is directly related to use the AGVs with full efficiency.

3.1 The Assumptions

The following assumptions are considered to define the AGV scheduling problem in the container terminals:

Assumption-1: It is assumed that the problem involves only one ship. For the ship, n containers jobs must be transported from the quay-side to the year-side or vice versa. The source and destination of the containers jobs as well as their appointment time on the quay-side are given. To load/unload the containers from a vessel or in the yard, a QC or RTGC is used.

Assumption-2: The RTGCs or yard crane resources are always available, i.e., the AGVs will not suffer delays in the storage yard location or waiting for the yard cranes.

Assumption-3: There is a predetermined crane job sequence, consisting of loading jobs, or unloading/discharging jobs, or a combination of both for every QC. Given a specified job sequence, the corresponding drop-off (for loading) or pickup (for discharging) times of the jobs on the quayside depends on the work rate of the quay cranes. After the ship docked at the quay-side, the appointment time of the j^{th} job is calculated by the following expression :

$$AT_j = \text{Ship-docked-time} + j \times W.$$

The Ship-docked-time is the time at which the ship is ready for discharge/loading on the quay-side. The time window W is the duration of discharging/loading a container.

Assumption-4: We are given a fleet of $V=\{1,2,..,|V|\}$ vehicles. Each vehicle transports only one container. At the start of the process, the vehicles are assumed to be empty.

Assumption-5: It is assumed the vehicles move with an average speed so that there are no Collisions, Congestion, Live-locks, Deadlocks[36]] and breakdown problem.

Assumption-6: We assumed the container jobs are distributed in the terminal so that each pickup/drop-ff) point is visited once only by a vehicle. In other word, a QC and RTGC are not busy in each node by different container jobs at the same time.

Assumption-7: In this scheduling problem, our goal is to deploy the AGVs such that all the imposed appointment time constraints are met with minimum cost. Our objectives are to minimize (1) the total AGV waiting time on the quay side; (2) the total AGV traveling time in the route of port; (3) the total lateness times to serve the jobs.

3.2 The formulation

Since the vehicles are Single-Load AGVs (see the Assumption-4), the problem can be converted to a Minimum Cost Flow (MCF) problem. For more details on the MCF problem and the scheduling AGV problem, readers can refer to [13], [42]. The MCF is a well-known problem in the area of network optimisation, i.e. the problem is to send flow from a set of supply nodes, through the arcs of a network, to a set of demand nodes, at minimum total cost, and without violating the lower and upper bounds on flows through the arcs. The problem for two vehicles and four jobs is demonstrated in the Figure-2. In the figure the supply nodes are denoted by A1 and A2. Each of these nodes has a one unit supply. There is only a demand node in the MCF problem. This node has -2 units demand. The directed arcs from A1 and A2 to the demand node must be added to the network model. These arcs show that an AGV can remain idle without serving any job. Therefore, a cost of zero is assigned to these arcs. The lower bound, upper bound and cost of each arc are noted by the triplex [Lower Bound, Upper Bound, Cost].

Solving the MCF problem generates 2 paths (the number of vehicles), each of which commences from a vehicle node and terminates at the demand node. Each path determines a job sequence of every vehicle. Suppose that for some values of arc costs, the paths given by a solution are A1→1→5→4→8→9 and A2→2→6→3→7→9. This states that AGV 1 is assigned to serve jobs 1 and 4, and AGV 2 is assigned to serve jobs 2 and 3, respectively.

3 The Algorithms

In this section, two algorithms to tackle the problem, Network Simplex Algorithm (NSA) and Network Simplex plus Algorithm (NSA+) are presented. NSA+ is an extended NSA with three enhanced features.

3.1 Network Simplex Algorithm (NSA)

Every connected graph has a spanning tree [16]. The network simplex algorithm maintains a feasible spanning tree at each iteration and successfully goes toward the optimality conditions until it becomes optimal. At each iteration, the arcs in the graph are divided into three sets; the arcs belong to the spanning tree (T); the arcs with flow at their lower pound (L); the arcs with flow at their upper bound (U). A spanning tree structure (T, L, U) is optimal if the reduced cost for every arc $(i,j)\in L$ is greater than zero and at the same time the reduced cost for every arc $(i,j)\in U$ is less than zero [1]. With those conditions, the current solution is optimal. Otherwise, there are arcs in the graph that violate the optimal conditions. An arc is a violated arc if it belongs to L (U) with negative (positive) reduced cost. The algorithm in Figure-2 specifies steps of the method[39].

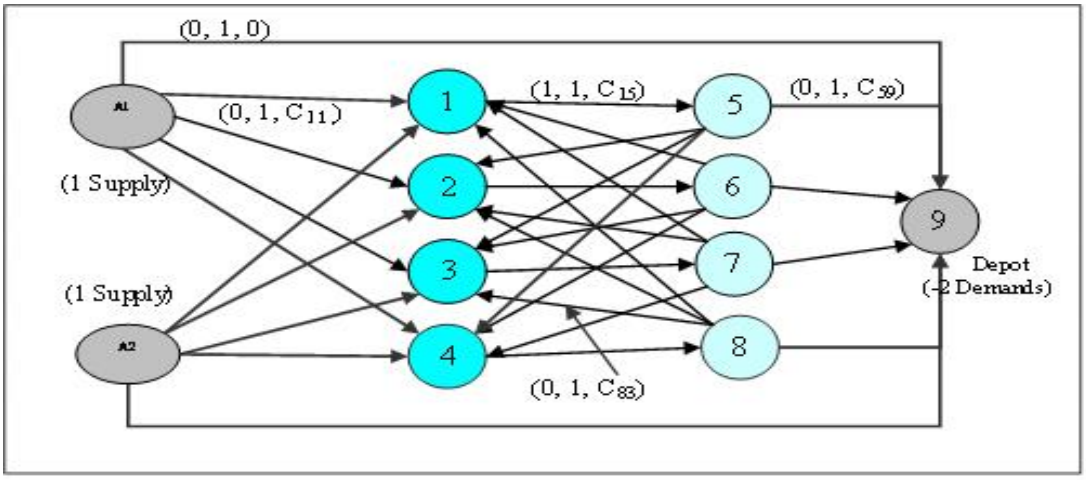

Figure-1: The MCF model for 2 AGVs and four container jobs.

To create the initial or Basic Feasible Solution (BFS), an artificial node 0 and artificial arcs are appended to the graph. The node '0' will be the root of spanning tree (T) and the artificial arcs, with sufficiently large costs and capacities, connect the nodes to the root. The set L consists of the main arcs in the graph, and the set U is empty [16]. Appending the entering arc (k, l), which is a violated arc, to the spanning tree forms a unique cycle, *W*, with the arcs of the basis. In order to eliminate this cycle, one of its arcs must leave the basis. The cycle is eliminated when we have augmented flow by a sufficient amount to force the flow in one or more arcs of the cycle to their upper or lower bounds. By augmenting flow in a negative cost augmenting cycle, the objective value of the solution is improved. The first task in determining the leaving arc is the identification of all arcs of the cycle. The flow change is determined by the equation θ = min { f_{ij} for all (i, j) $\in$ *W*}. The leaving arc is selected based on cycle W. The substitution of entering for the leaving arc and the reconstruction of new tree is called a pivot. After pivoting to change the basis**,** the reduced costs for each arc (i, j) $\notin$ T is calculated. If the reduced costs for all (i, j) $\in$ {L + U} satisfy the optimality condition then the current basic feasible solution is optimal. Otherwise, an arc (*i, j*) where there is a violation should be chosen and operations of the algorithm should be repeated.

Different strategies are available for finding an entering arc for the basic solution. These strategies are called pricing rules. The performance of the algorithm is affected by these strategies. The standard textbook [16] provided a detailed account of the literature on those strategies. Bradley, Brown and Graves (1977), used a dynamic queue, containing the indices of so-called 'interesting' nodes and admissible arcs. Their method is called BBG Queue pricing scheme. An 'interesting' node is a node whose incident arcs have not been re-priced in recent iterations. At each iteration, the entering arc is selected from the queue. Goldfarb and Reid (1977) proposed a steepest edge pricing criterion. Mulvey (1978) suggests a major and minor loop to select the entering arc. A limited number of favourably priced entering arcs are collected by scanning the non-basic arcs in a major iteration. In the minor iteration, the most favourably priced arc in the list is chosen to enter the basis. Grigoriadis (1986) describes a very simple arc block pricing strategy based on dividing the arcs into a number of subsets of specified size. At each iteration, the entering arc is selected from a block with most negative price. Andrew (1997) studied practical implementation of minimum cost flow algorithms and claimed that his/her implementations worked very well over a wide range of problems [3].

Masakazu (1999) used a primal-dual symmetric pivoting rule and proposed a new scheme in which the algorithm can start from an arbitrary pair of primal and dual feasible spanning tree [11]. Eppstein (1999) presented a clustering technique for partitioning trees and forests into smaller sub-trees or clusters [6]. This technique has been used to improve the time bounds for optimal pivot selection in the primal network simplex algorithm for minimum-cost flow problem. Lobel (2000) developed and implemented the *multiple pricing rules* to select an entering arc, a mixture of several sizes for the arc block [23]. A general pricing scheme for the simplex method has been proposed by Istvan [21]. His pricing scheme is controlled by three parameters. With different settings of the parameters, he claimed that it creates a large flexibility in pricing and applicable to general and network simplex algorithms. Ahuja et al. (2002) developed a network simplex algorithm with O(n) consecutive degenerate pivot [1]. He presented an anti-stalling pivot rule, based on concept of *strong feasible* spanning tree. The basis structure (T, L, U) is *strongly feasible* if we can send a positive amount of flow from any node to root along arcs in the spanning tree without violating any of the flow bounds.

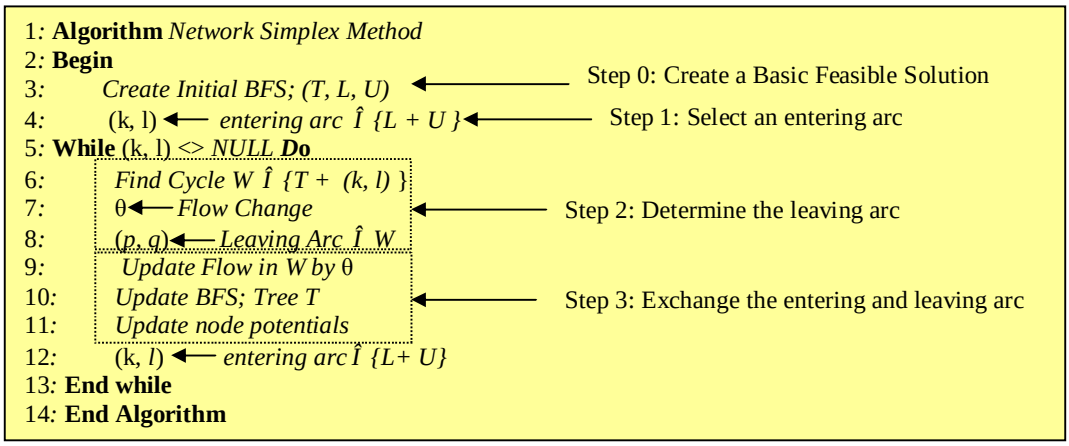

Figure-2: The Network Simplex Algorithm (NSA)

Istvan reviewed a collection of some known pricing schemes in the original simplex algorithm [21]. They are First improving candidate, Dantzig rule, Partial pricing, Multiple pricing and Sectional pricing. These schemes can be applied to NSA. First improving candidate chooses the first violate arc as the entering arc. It is cheap but it usually leads to a very large number of iterations. In Dantzig rule all non-basic arcs are checked (full pricing) and one which violates the optimality condition the most is selected. This rule is quite expensive but overall is considerably better than the previous method. The Partial pricing scans only a part of the non-basic arcs and the best candidate from this part is selected. In the next step, the next part is scanned, and so on. In Multiple pricing, some of the most profitable candidates (in terms of the magnitude) are selected during one scanning pass. They are updated and a sub-optimization is performed involving the current basis and the selected candidates using the criterion of greatest improvement. The Sectional pricing behaves as a kind of partial pricing, but in each iteration sections or clusters of arc are considered.

3.2 The Network Simplex plus Algorithm (NSA+)

NSA+ is an efficient extension of NSA. Compared with the standard version of NSA by Grigoriadis's blocking scheme [8] and maintaining the strongly feasible spanning tree [16], NSA+ has three new features. These features are concerned with the starting point/block for scanning violated arcs, the memory technique and the scanning method. The pricing scheme of NSA+ is designed based on these features.

There is a function for the pricing scheme to find out an entering arc. The pseudo-code for this function is illustrated in Figure-3. The arcs in the graph of MCF model are divided into several blocks with the same size and each block is identified by a specific number, known as Block-Number. For each problem, the number of blocks is calculated by dividing the number of arcs in the graph into the block's size.

At first iteration, when the initialization is needed and the packet is empty, the number of blocks is calculated and the first one to be scanned for the optimality condition is chosen (see the lines 2-5). The function selects the first block randomly or by a heuristic method (based on location of the biggest cost, for example). Note that at first iteration the lines 6-9 don't perform anything because the packet is empty (these will be activated from the second iteration and when the packet is not empty). Scanning of the arcs for violation among different blocks is chosen circularly. At each scan one violating arc (at most) from each block is put into the packet as long as it has empty place and there is any violated arc (see the lines 10-14). The capacity of the packet is more than the block's size and the most violating arcs are kept at the top of the packet. At the end of function, if the packet is empty, the current solution is optimal (see the lines 15-17). Otherwise the packet will be sorted in descending order, based on the absolute value of the reduced costs, and the most violated arc will be chosen as the entering arc (see the lines 18-19).

The memory technique will be activated from the second iteration. It uses a few elements at the top of the packet of the last iteration. The size of this memory may be a percentage of the block's size. The reduced costs of the most violated arcs in the previous iteration are recalculated (see the line 6). If they violate the optimality conditions again, they are kept in the packet. Otherwise they must be removed from the packet, which can be replaced by new violating arcs (see the lines 7-9). The reaming part of the function acts as before.

As we mentioned, there are two options to choose the first block to be scanned; Randomly and Heuristically. Hence, NSA+ has two extensions: (a) NSA+R: The entering arc function chooses the first block by Random selection; (b) NSA+H: The entering arc function chooses the first block by a Heuristic method (based on location of the largest cost in the graph).

```
1: arc  Entering_Arc_Function
2:          If Initialization is needed Then  // the packet is empty
3:             Calculate the number of blocks
4:             Choose the Block-Number  // Randomly or by a Heuristic method
5:          End if
6:          Recalculate the Reduced Costs of the most violated Arcs in the Packet
7:          If the most violated elements satisfy the optimality conditions Then
8:             Remove the elements from the packet
9:          End if
10:         While the Packet has empty place AND there is any violated arc in the graph Do
11:            Calculate the reduced cost of an arc from the block associated with the Block-Number.
12:            Put the arc into the Packet if it violates the optimality condition.
13:            Increase the Block-Number circularly.
14:         End While
15:         If the Packet is Empty Then
16:            Return Null        // The Current Solution is Optimal
17:         End If
18:         Sort the Packet Descending   // Based on the absolute value of the reduced costs by Quick Sort
19:         Return the first element of the Packet
20:  End Function
```

Figure-3: The **Pseudo-code** of selecting an entering arc in Network Simplex plus Algorithm

3.3 The differences between NSA and NSA+

The main difference between NSA and NSA+ are in the pricing scheme and the entering arc procedure. As we mentioned, the role of the pricing scheme is that how the entering arc to be selected from the violated arcs in the graph. The differences between NSA and NSA+ are as flows:

- At each iteration, a packet of violated arcs from different blocks is collected in NSA+ and the most violated arc is selected as the entering arc, whereas NSA selects the most violated arc from one block.
- There is no memory technique in NSA while NSA+ uses a few elements at the top of the packet for the next iteration. It benefits from the current violated arcs for the next iteration.
- The first block is selected Randomly or by a Heuristic method in NSA+, whereas NSA always chooses the first block for scanning the violated arcs.

4 Experimental Results from the implementation and running the algorithms

We implemented the standard version of Network Simplex Algorithm (see Figure-2). As we mentioned, the pricing rule or scheme to choose the entering arc in Step 1 determines the speed of algorithm. In the literature, we reviewed the pricing rules. Actually, there is the trade-off between time spent in pricing at each iteration and the 'goodness' of the selected arc in terms of reducing the number of iterations required to reach the optimal solution. The First improving candidate and Dantzig rule represent two extreme choices for the entering arc. Other pricing schemes strike an effective comprise between these two extremes and have proven to be more efficient in practice [16]. Kelly and Neill [39] implemented several pricing schemes and ran their software for different classes of minimum cost flow problems. In their results, the block pricing scheme provided a better performance compared with others. We therefore chose the block pricing scheme. This scheme is based on dividing the arcs of the graph into a number of subsets of specified size. A block size of between 1% and 8.5% of the size of the arcs in the graph has been recommended by Grigoriadis [39], for large MCF problems. We set the number to 5% by the try and error.

To test the model and make a comparison between NSA and NSA+, a hypothetical port was designed. The parameters in Table-1 were used to define the port.

Table-1: Value of Parameters for the simulation

Description of the Parameters	Values
Number of Vehicles in the port	50
Number of Quay Cranes	7
Number of Blocks in the yard (Storage area inside the port)	32
Time Window of the Cranes	120 seconds
Travelling Time between every two points in the port (see Assumption 1)	Random between 1 and 100 seconds

We implemented our software in Borland C++. Then, it has been run to solve several random problems. The sources and destinations of jobs were chosen randomly. The CPU-Time required to solve the problems by the two algorithms has been drawn in Figure-4 and Figure-5, according to the number of jobs and the number of arcs, respectively. Also the power estimation for those two curves has been shown on the figures.

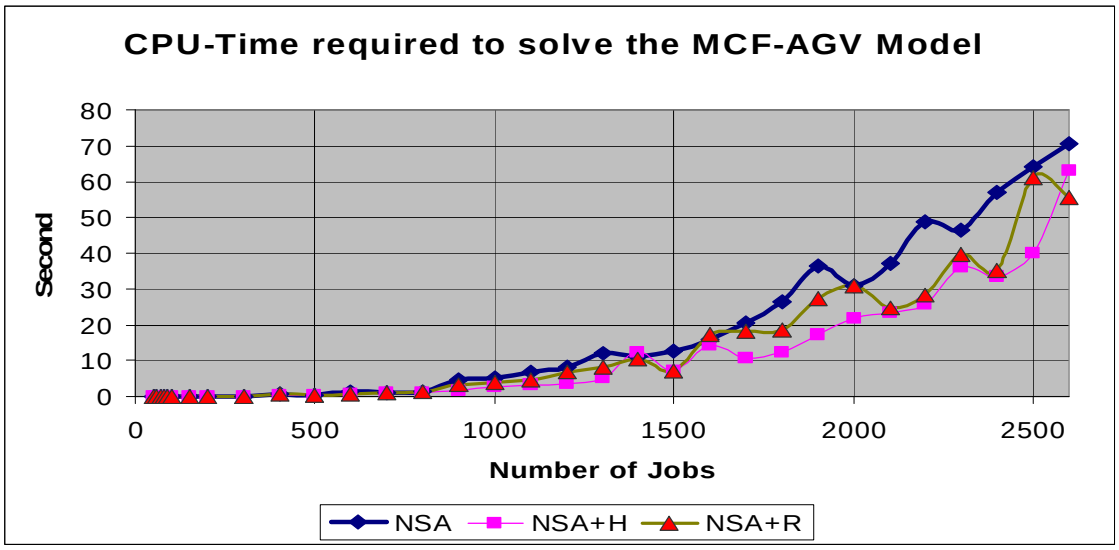

Figure- 4: CPU-Time to solve the static problem by NSA and NSA+, based on the number of jobs

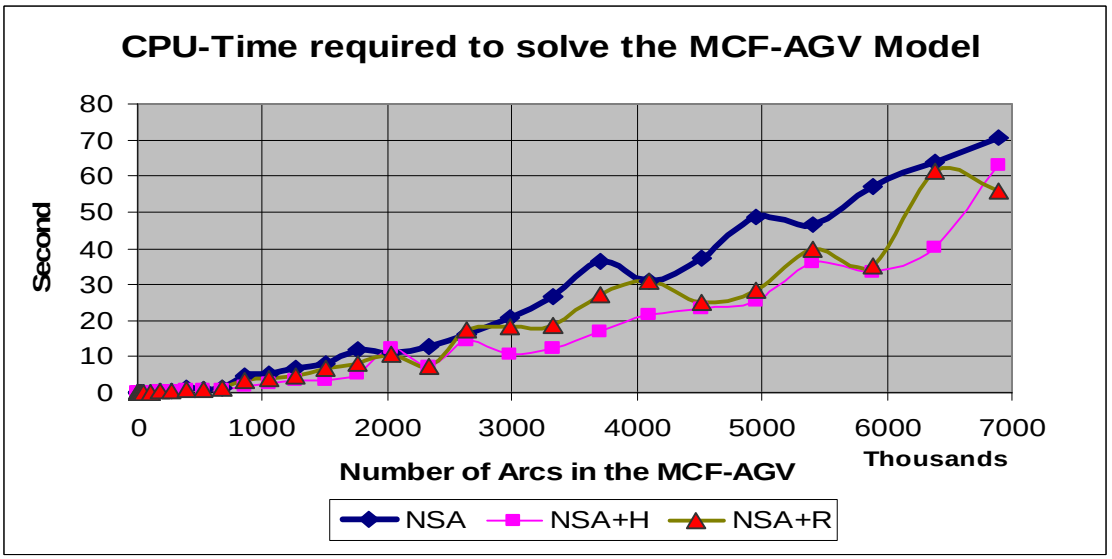

Figure-5: CPU-Time to solve the static problem by NSA and NSA+, based on the number of arcs

All experiments were run on a Pentium 2.2 GHz PC with 1 GMB RAM. From the figures, we can observe that:

Observation 1: NSA and NSA+ are run in polynomial time to solve the MCF-AGV model, in practice.

Observation 2: NSA+ is fast and more efficient than NSA.

There are two different types of iteration in NSA, degenerate and non-degenerate [16]. In every non-degenerate iteration, the value of the objective function is decreased whereas degenerate iterations do not change the objective function's value. In the degenerate iterations, a flow change of zero causes cycling. In the literature, Grigoriadis experienced that cycling is rare in practical application [8]. Observation 1 confirms the experience.

In order to confirm that NSA is run in polynomial time to solve the MCF-AGV model (Observations 2), we estimated complexity of the algorithm. The result shows that the CPU-Time required to tackle the problem, is a function with degree 3 of the number of jobs in the problem [42].

4.1 The percentage of improvement in CPU-Time required to tackle the problem

In order to calculate the average CPU-Time required to solve the problems and to compare performance of the algorithms in this experiment, we introduce the following terms:

T_i^{NSA}: The CPU-Time used to solve the problem i by NSA.

T_i^{NSAH} : The CPU-Time used to solve the problem i by NSA+H.

T_i^{NSAR}: The CPU-Time used to solve the problem i by NSA+R.

PIH_i: The Percentage of Improvement in CPU-time used to solve the problem i by NSA+H compared with NSA.

PIR_i: The Percentage of Improvement in CPU-time used to solve the problem i by NSA+R compared with NSA.

TPIH: The Total Percentage of Improvement in CPU-Time used to solve the problems by NSA+H compared with NSA.

TPIR: The Total Percentage of Improvement in CPU-Time used to solve the problems by NSA+R compared with NSA.

Wi: The Weight of improvement for the problem i. In this experiment we consider the number of arcs in the MCF-AGV model for the weight. Given N jobs and M AGVs in the problem, the number of arcs is M+M×N+N×(N-1)+2×N.

Now we calculate the percentage of improvements in the CPU-Time used for problem i by the following terms:

$$\text{PIH}_i = \frac{100 * (T_i^{NSAH} - T_i^{NSA})}{T_i^{NSA}}$$

$$\text{PIR}_i = \frac{100 * (T_i^{NSAR} - T_i^{NSA})}{T_i^{NSA}}$$

The total percentages of improvement in the CPU-Time used to solve the problems by NSA+H and NSA+R, compared with NSA, are calculated by the following expressions:

$$TPIH = \frac{\sum_{i=1}^{32} W_i \times PIH_i}{\sum_{i=1}^{32} W_i} = -35.16\%$$

$$TPIR = \frac{\sum_{i=1}^{32} W_i \times PIR_i}{\sum_{i=1}^{32} W_i} = -21.28\%$$

4.2 Statistical test for the comparison

The CPU-time required to solve the problems by the two algorithms, NSA and NSA+, were analysed statistically. We tested the null hypothesis that the means produced by the two algorithms were statistically indifferent. Since we cared the change (the difference between the two means) was positive or negative, 'One-tail' test was chosen. The result of Paired T-test along with the critical values of T-distribution for the particular degree of freedom are shown in Table-2. The T-test confirms that NSA+ is significantly better than NSA with 95% degree of confidence.

Table-2: The result of T-Test for the comparison between two algorithms, NSA and NSA+

Statistical Parameters	NSA+H vs. NSA	NSA+R vs. NSA
Number of Samples/Observations	32	32
T-Test (Paired Two Sample For Means)	-4.1799	-3.3617
Degree of Freedom	31	31
Critical T-Value	-1.6955	1.6955

4.3 Complexity of NSA+

Given N jobs and M AGVs in the problem (N>>M), the complexity of the NSA+ is calculated as follows: Assume that the maximum flow, MF, in each of the m arcs, at maximum cost, C, for the minimum cost flow model. So there is an upper bound on the value of the objective function. This upper bound is given by m•C•MF. There are two different types of pivots in the algorithm, non-degenerate and degenerate pivots. The former is bounded by m•C because the number of non-degenerate pivots in the algorithm is bounded by m•C•MF (MF=1 in the MCF-AGV model). The number of degenerate pivots is determined by the sum of nodes potential and maintaining the strongly feasible spanning tree. Given n as the number of nodes in the graph model, the sum of nodes potential is bounded by n^2•C. It is decreased at each iteration when the spanning tree is strongly feasible [2]. A series of degenerate pivots may occur between each pair of non-degenerate pivots, and thus a bound on the total number of iterations is m•n^2•C^2. Find the entering arc is O(m) and sorting the packet is O(K•LogK) operation (K is size of the packet, K=225). Finding the cycle, amount of flow change, leaving arc and updating the tree are O(n) operations. Hence the complexity of each pivot is O((m + n) K•LogK). Based on the complexity of the number of iterations and the complexity of each pivot, the total complexity of this algorithm is determined by the following equation:

$$O((m+n)mn^2C^2KLogK)$$

Since m=O(N^2) ; n=O(N), the total complexity of NSA+ to tackle the MCF-AGV model is O(N^6).

5 Concluding Summary

In this paper, two algorithms, NSA and NSA+, were applied to the automated guided vehicles scheduling problem in container terminals. Our experimental results suggested that NSA could find the global optimal solution for 2,600 jobs and 7 millions arcs in the graph model within 70 seconds by running on a 2.2 GHz Pentium PC. NSA+ has enhanced features over NSA and it is faster. The most effective feature of NSA+ is a memory technique and scanning method, which can be applied to Original Simplex Algorithm in Operation Research.

References

Journal Paper:

[1] Ahuja R.K., Orlin J. B., Sharma P., Sokkalingam P.T., "A network simplex algorithm with O(n) consecutive degenerate pivots". Operations Research, Vol 30(3), pp 141-148, 2002.

[2] Ahuja R.K., Orlin J.B., Giovanni M.S., Zuddas P., "Algorithms for the simple equal flow problem", Management Science, Vol 45(10), pp 1440-1455, 1999.

[3] Andrew V.G., "An efficient implementation of a scaling minimum-cost flow algorithm". Journal of Algorithms, Vol 22(1), pp 1-29, 1997.

[4] Chiang, Wen-Chyuan and Robert A. Russell. "Simulated Annealing Metaheuristic for the Vehicle Routing Problem with Time Windows," Annals of Operations Research, Vol. 63, pp 3-27, 1996.

[5] Cunningham W.H. "Theoretical properties of the network simplex method". Mathematics of Operations Research, 4(2):196–208, 1979.

[6] Eppstein D, "Clustering for faster network simplex pivots", In Proc. 5th ACM-SIAM Symposium. Discrete Algorithms, pp 160–166, 1999.

[7] Goldberg A.V., Kennedy, R, **"**An Efficient Cost Scaling Algorithm for the Assignment Problem**".** Technical Report, Stanford University, 1993**.**

[8] Grigoriadis, M.D. 'An Efficient Implementation of the Network Simplex Method', Mathematical Programming Study Vol. 26, pp 83-111, 1986.

[9] Grunow M, Günther H.O, Lehmann M., "Dispatching multi-load AGVs in highly automated seaport container terminals", OR Spectrum, Volume 26 (2), pp 211-235, 2004.

[10] Helgason R., Kennington J., "Primal Simplex Algorithms for Minimum Cost Network Flows," Handbook on Operations Research and Management Science Volume 7, North-Holland, Amsterdam, pp 85-133, 1995.

[11] Masakazu M., "On network simplex method using primal-dual symmetric pivoting rule", Journal of Operations Research of Japan, Vol 43, pp 149-161, 1999.

[12] Murty K.G., Jiyin L., Yat-Wah W, Zhang C, Maria C.L. Tsang, Richard J. Linn., "DSS (Decision Support System) for operations in a container terminal". Decision Support System, Vol 39, pp 309-332, 2002.

[13] Rashidi H., "Scheduling in container terminals using Network Simplex Algorithm", Journal of Industrial Engineering, Volume 1, pp 9-16, Iran. 2008.

[14] Tsang E.P.K., "Scheduling techniques -- a comparative study", British Telecom Technology Journal, Volume 13 (1), pp 16-28, Martlesham Heath, Ipswich, UK. 1995

[15] Wook B.J., Hwan K.K.,"A pooled dispatching strategy for automated guided vehicles in port container terminals", International Journal of management science, Vol 6 (2), pp 47-60, 2000.

Books or Chapter based Booklets:

[16] Ahuja R.K., Magnanti T.L., Orlin J.B.,, "Network Flows: Theory, Algorithms and Applications". Prentice Hall, 1993.

[17] Goldberg D.,"Genetic Algorithms in Search, Optimization and Machine Learning". Addison-Wesley, Reading, 1989.

Technical Reports:

[18] Cheng Y., Sen H., Natarajan K., Teo C., Tan K.,"Dispatching automated guided vehicles in a container terminal", Technical Report, National University of Singapore, 2003.

[19] Czech Z. and Czarnas P., "Parallel Simulated Annealing for the Vehicle Routing Problem with Time Windows. In Proceedings of 10th Euromicto Workshop on Parallel Distributed and Network-Based Processing, Canary Islands, Spain, pp 376-383, 2002.

[20] Huang Y., Hsu W.J., "Two Equivalent Integer Programming Models for Dispatching Vehicles at a Container Terminal". CAIS, Technical Report 639798, School of Computer Engineering, Nan yang Technological University, Singapore, 2002.

[21] Istvan M., "A General Pricing Scheme for the Simplex Method ", Technical Report, Department of Computing, Imperial College, London, 2003.

[22] Galati M., Geng H., and Wu T., "A Heuristic Approach For The Vehicle Routing Problem Using Simulated Annealing", Lehigh University, Technical Report IE316, 1998.

[23] Löbel A., "A Network Simplex Implementation", Technical Report, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), 2000.

[24] Sen H., "Dynamic AGV-Container Job Deployment". Technical Report, HPCES Programme, Singapore-MIT Alliance, 2001.

[25] Zhang L.W, Ye R, Huang S.Y, Hsu W.J, "Two Equivalent Integer Programming Models for Dispatching Vehicles at a Container Terminal". School of Computer Engineering, Nan yang Technological University, Technical Report, Singapore, 2002.

Conferences:

[26] Böse J., Reiners T., Steenken D., Voß S.,, ”Vehicle Dispatching at Seaport Container Terminals Using Evolutionary Algorithms”. Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, IEEE, pp 1-10, 2000.

[27] Chan S.H, “Dynamic AGV-Container Job Deployment”, Master of Science, University of Singapore, 2001.

[28] Rashidi H. & Tsang E.P.K, "Applying the Extended Network Simplex Algorithm and a Greedy Search Method to Automated Guided Vehicle Scheduling", Proceedings, 2nd Multidisciplinary International Conference on Scheduling: Theory & Applications (MISTA), New York, Vol 2, pp 677-693, 2005.

[29] Hasama T, Kokubugata H, Kawashima H., “A Heuristic Approach Based on the String Model to Solve Vehicle Routing Problem with Backhauls”, Proceeding of the 5th World Congress on Intelligent Transport Systems (ITS), Seoul, 1998.

[30] Meersmans P.J.M, Dekker R, “Operations research supports container handling”, Technical Report EI 2001-22, Erasmus University of Rotterdam, Econometric Institute, 2003.

[31] Meersmans P.J.M, Wagelmans A.P.M., “Dynamic scheduling of handling equipment at automated container terminals”, Technical Report EI 2001-33, Erasmus University of Rotterdam, Econometric Institute, 2001.

[32] Meersmans P.J.M, Wagelmans A.P.M., “Effective algorithms for integrated scheduling of handling equipment at automated container terminals”. Technical Report EI 2001-19, Erasmus University of Rotterdam, Econometric Institute, 2001.

[33] Patrick J.M., Dekker R., “Operations research supports container handling”, Technical Report EI 2001-22, Erasmus University of Rotterdam, Econometric Institute, 2003.

[34] Patrick J.M., Wagelmans P.M., “Dynamic scheduling of handling equipment at automated container terminals”, Technical Report EI 2001-33, Erasmus University of Rotterdam, Econometric Institute, 2001.

[35] Patrick J.M., Wagelmans P.M., “Effective algorithms for integrated scheduling of handling equipment at automated container terminals”, Technical Report EI 2001-19, Erasmus University of Rotterdam, Econometric Institute, 2001.

[36] Qiu L., Hsu W.-J., Huang S.-Y and Wang H., “Scheduling and Routing Algorithms for AGVs: a Survey”. International Journal of Production Research, Taylor & Francis Ltd, Vol. 40 (3), pp 745-760, 2002.

[37] Qiu L, Hsu W.J., “A bi-directional path layout for conflict-free routing of AGVs”. International Journal of Production Research, Volume 39 (10), pp 2177-2195, 2001.

[38] Qiu L, Hsu W.J., “Scheduling of AGVs in a mesh-like path topology”. Technical Report CAIS-TR-01-34, Centre for Advanced Information Systems, School of Computer Engineering, Nanyang Technological University, Singapore, 2001.

Dissertations:

[39] Kelly D.J., ONeill G.M., "The Minimum Cost Flow Problem and The Network Simplex Solution Method", Master Degree Dissertation, University College, Dublin, 1993.

[40] Larsen A., “The Dynamic Vehicle Routing Problem”, PhD Thesis, Technical University of Denmark, 2000.

[41] Leong C. Y., "Simulation study of dynamic AGV-container job deployment scheme", Master of science, National University of Singapore, 2001.

[42] Rashidi H., “Dynamic Scheduling of Automated Guided Vehicles in Container Terminals”, PhD Thesis, Department of Computer Science, University of Essex., 2006