Scheduling in Container Terminals using Network Simplex Algorithm

Hassan Rashidi
Department of Computer Engineering, Islamic Azad University of Qazvin, Qazvin, Iran
Corresponding author, Hrashi@Gmail.com

Abstract:
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 the Network Simplex Algorithm (NSA). The algorithm is 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. The results show that NSA is fast and efficient. It is found that, in practice, NSA takes 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 [0]. 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 the problem include quay cranes (QC), container storage areas, rubber tyred gantry crane (RTGC) or yard crane, and a road network [see e.g. 5, 6, 10, 16, 17, 0]. 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.
The structure of this paper is as follows. Section 1 reviews definition for Minimum Cost Flow (MCF) model and introduces a formal definition for the MCF model. Section 2 presents the scheduling problem of Automated Guided Vehicles (AGV) in container terminals as a special case of Minimum Cost Flow (MCF) problem. The problem is labelled as MCF-AGV model. Section 3 presents an algorithm to tackle the MCF-AGV model.
Experimental results from applying the algorithm to tackle the model are presented in Section 4. Section 5 is considered to summary and conclusion.

2. Minimum Cost Flow (MCF) model

This section systematically presents a formal definition for the MCF model:

Definition 1 [0]: In an informal description of the MCF model, let graph G = (N, A) be a directed network defined by a set of nodes, N, together with a set of arcs, A. Each arc (i, j) $\in A$ has an associated cost c_{ij} that denotes the cost per unit flow on that arc. It is assumed that the flow cost varies linearly with the amount of flow. The maximum and minimum amount of flow on each arc (i, j) $\in A$ are limited by M_{ij} and m_{ij} ($m_{ij} \le M_{ij}$), respectively. A real number b_i is associated with each node, representing its supply/demand. If b_i is greater (less) than zero, node i is a supply (demand) node; and if b_i =0, node i is a transhipment node. The decision variables in the MCF problem are arc flows, which is represented by f_{ij} for arc $(i, j) \in A$. The standard form of Minimum Cost Flow model is as follows:

$$Min \quad Cost \quad Flow \quad = \sum_{(i,j)\in A} c_{ij}.f_{ij}$$

$$Subject \quad To:$$

$$\sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)\in A} f_{ji} = b_i \quad , for \quad all \quad i \in N$$

$$m_{ij} \le f_{ij} \le M_{ij}, \quad for \quad all (i,j) \in A$$

These constraints state that flows must be feasible and conserve each node. For the feasible flows to exist the MCF problem must also have $\sum_{i\in N} b_i = 0$, which means that
the network is balanced. A special graph for the MCF model is defined as follows:

Definition 2:
A MCF Graph G_{MCF} = (G, NP, AP) consists of a graph G with a couple of properties for the nodes and arcs in G. The NP and AP are the Node's and Arc's Properties, respectively. The node property function NP: N→R (Real numbers; possibly negative) gives the amount of supply/demand of the nodes. This function for each node is defined as follows:

NP(i) = NP_i= b_i where:

$b_i > 0$ if node i is a supply node

$b_i < 0$ if node i is a d*emand* node

$b_i = 0$ if node i is a transshipment node

So that $\sum_{i \in N} NP(i) = 0$

Each arc in A has three properties:
a lower bound, an upper bound and a cost. The arc property function AP maps each arc to these properties, AP: A→R×R×R (Real numbers; nonnegative). For each arc ∈ *A*, the mapping is denoted by AP(i,j), or AP_{ij} in short. The lower bound, upper bound and cost are denoted by m_{ij}, M_{ij} and c_{ij}. Based on Definitions 1 and 2, the standard Minimum Cost Flow (MCF) problem is defined formally as follows:
Definition 3: a MCF model is defined as MCF=(G_{MCF}, *f*, D, CS, FC) where G_{MCF} = ((N,A), NP, AP) is a graph with nodes and arcs specific to the MCF model (Definition 2); f is a finite set of decision variables on A (*f* stands for flow*),f =*{ f_{ij} | (*i, j*) ∈ *A}* ; D = a function which determines a lower and upper bound for f; D: $f \rightarrow$ R×R (to be pulled out from AP); the lower bound and upper bound of f_{ij} are labelled by $D_{f_{ij}}$ as (D stands for Domain); CS is a finite set of Constraint on NP and f; FC is an objective function for the Flow's Cost on AP and f. The task in a MCF model is to assign a value to each f_{ij} that satisfy all constraints in CS with regard to the minimum value of FC. The domains, constraints and objective function in the standard form of the MCF model are as followed:

(a) For each element in D and f, $D_{f_{ij}}$ = [m_{ij}, M_{ij}], for $\forall$ (*i, j*) ∈ *A*;

(b) The CS is:

$$\sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(i,j)\in A} f_{iji} = NP_i, \quad \forall i \in N ;$$

(c) The FC is $\sum_{j:(i,j)\in A} c_{ij} f_{ij}$

3. The special case of the MCF model for Automated Guided Vehicles Scheduling

In this section, a scheduling problem of Automated Guided Vehicles (AGVs) in the container terminals is introduced. The problem is to deploy several AGVs in a port to carry many containers from the quay-side to yard-side or vice versa. The main reason to choosing this problem is that the efficiency of a container terminal is directly related to use the AGVs with full efficiency [see e.g. 5, 6, 10, 19, 24]. This problem is formulated as a special case of the MCF model.

3.1 Assumptions

Assumption 1: The layout of a port container terminal is given [24]. According to a specific layout the travel time between every combination of Pickup (P) /Drop-off (D) points is provided.
Assumption 2: It is assumed the vehicles move with an average speed so that there are no Collisions, Congestion, Live-locks, Deadlocks [19] and breakdown problem while they are carrying the containers.
Assumption 3: Every AGV can transport only one container. Also it is assumed that the start location of each AGV at the beginning of process is given.
Assumption 4: The yard is divided to several blocks and RTGCs or yard crane resources are always available [6], i.e., the AGVs will not suffer delays in the storage yard location or waiting for the yard cranes.
Assumption 5: The quay side consists of several Quay Cranes (QCs). For each QC, there is a predetermined job sequence, consisting of loading or unloading jobs, or a combination of both. For each loading (unloading) job, there is a predetermined pickup (drop-off) point in the yard, which is the origin (destination) of the job.
Assumption 6: There are N jobs and M AGVs in the problem. The source and destination of jobs as well as their appointment time on the quay side are given.
Assumption 7: The objectives are to minimize (a) the total AGV waiting time at the quay side (b) the total AGV travelling time in the route of port (c) the total lateness times to serve the jobs. Cheng et al. (2003) minimized the impact of delays and waiting times of the AGVs at the quay side [0].

3.2 Formulation of the problem

Here, a special case of the MCF model is presented for the scheduling problem of automated guided vehicles in container terminal. The problem differs primarily in the arrangement of nodes and arcs with their properties. In this special case, the property function of nodes assigns integer values to the nodes. Additionally, the property function of arcs assigns integer values to the lower bound, the upper bound and the cost of each arc. Here, the special Graph of G_{MCF} for the Automated Guided Vehicles Scheduling ($G_{MCF\text{-}AGV}$) and the special case of the MCF model for the scheduling problem of AGVs (MCF-AGV) are presented.
Based on Definition 2, the following definition is introduced for the G_{MCF} in a special case:
A MCF Graph for AGV, $G_{MCF\text{-}AGV}$ = (GS, NPS, APS), is a special case of G_{MCF} = (G, NP, AP) (Definition 2). The graph GS = (NS, AS) will be defined in the subsections below; the node and arcs properties of GS, NPS and APS, are also special cases of NP and AP, respectively (NPS:

NS→*N* and APS: AS→*N*×*N*×*N*; *N* is the set of Natural numbers). The components of $G_{MCF-AGV}$ are formally described in the two following sub-sections:

3.2.1 Nodes and their properties in the special graph

Let N be the number of jobs and M be the number of AGVs in the problem. The nodes of the MCF Graph for the AGV scheduling problem are defined as follows:

a) Supply Nodes: For each vehicle m, a supply node AGVNm with one unit supply is considered. Therefore, the set of supply nodes in the graph are SAGVN = {AGVNm | m=1,2,…,M; NPS(m)=1}

b) Transhipment Nodes: for each job j, a couple of nodes, Job-Input and Job-Output, are considered. Hence, the sets of transhipment nodes in the graph are SJIN U SJOUT where:

c) SJIN = {JINi | i=1,2,…,N; NPS(i)=0} where JINi is a node through which an AGV enters job i.

d) SJOUT = {JOUTi i=1,2,…,N; NPS(i)=0} where JOUTi is a node from which an AGV leaves job i.

e) **SINK:** It stands for a Sink node or a demand node in the $G_{MCF-AGV}$ with M units demand. This node corresponds to the end state of the process, after all container jobs have been served. Hence, for the property of this node, NPS(SINK) = −M.

f) Therefore, there are M+2*N+1 nodes in the $G_{MCF-AGV}$: NS = SAGVN U SJIN U SJOUT U SINK.

3.2.2 Arcs and their properties in the special graph

Below the four types of arcs that join the nodes in $G_{MCF-AGV}$, together with their properties are described:

1) Intermediate Arcs: These arcs are directed arcs from every Job-Output node i to other Job-Input node j.

These arcs with their properties are $ARC_{intermediate}$ ={ (i, j) | i∈ SJOUT, j∈ SJIN, j≠JIN_i, APS(m, j)= [0,1,C_{ij}]}

Where:

$$C_{ij}=\begin{cases} w_1\times(t_j-(t_i+DT_{ij}))+w_2\times DT_{ij} \;\; if \;\; t_j\geq t_i+DT_{ij} \\ P\times(t_i+DT_{ij}-t_j) \;\; Otherwise \end{cases};$$

In the cost, w_1 and w_2 are the weight of waiting and travelling times of the AGVs, respectively; t_j and t_j is the appointment time of job i and j on the quay side (to be unloaded or dropped-off); DT_{ij} is Travelling time from location of job i to location of job j; (see Rashidi's PhD thesis [0] for calculation of the DT_{ij} in different cases). If an AGV can serve job j after serving job i ($t_j \geq t_i + DT_{ij}$), the waiting and travelling times of the AGV are calculated without any lateness time. Otherwise, only the lateness time of serving job j with a penalty (P) is considered for the cost (see Assumption 7).

2) Inward Arcs: a set of arcs from SAGVN to SJIN. These arcs along with their properties are:

ARC_{inward} = { (m, j) | m ∈ SAGVN, j ∈ SJIN, APS(m, j) = [0,1,C_{mj}]}

Where

$$C_{mj}=\begin{cases} w_1\times\left(t_j-\left(RTA_m+TTA_{mj}\right)\right)+ \\ w_2\times\left(RTA_m+TTA_{mj}\right) if \;\; t_j\geq RTA_m+TTA_{mj} \\ P\times\left(RTA_m+TTA_{mj}-t_j\right) Otherwise \end{cases};$$

In the cost, w_1 and w_2 are the weight of waiting and travelling times of the AGVs, respectively; t_j is the appointment time of job j at the quay side (to be unloaded or dropped-off); RTA_m is

Ready time of AGV m at start location, where may be either the quay-side or yard; TTA_{mj} is the travel time of AGV m from the start location to the location of job j on the quay side; (The TTA_{mi} should be calculated in the similar manner like the calculation of DT_{ij}; see Intermediate arcs). If AGV m could arrive on the quay side in the appointment time of job j ($t_j \geq RTA_m+TTA_{mj}$), the waiting and travelling times of AGV m to serve job j are calculated as the cost. Otherwise, the lateness time to serving job j with a penalty (P) is considered.

3) Outward Arcs: These are directed arcs from every Job-Output node i and AGV node m to SINK. These arcs along with their properties are $ARC_{outward}$ ={ (i, j) | i ∈ SAGVN *U* SJOUT, j=SINK; APS(m, j) = [0,1,0] }. These arcs show that an AGV can remain idle after serving any number of jobs or without serving any job. Therefore, a cost of zero is assigned to these arcs.

4) Auxiliary Arcs: There is a directed arc from every Job-Input node i to its Job-Output node. These arcs along with their properties are ARCauxiliary ={ (i, j) | i ∈ SJIN, j=an unique Job-Output node in SJOUT, correspond to the Input-Node i; APS(i, j) = [1,1,0]}.These arcs guarantee that every Job-Input and Job-Output node is visited once only so that each job is served.

There are M·N+N·(N-1)+M+2·N arcs in the graph (AS=ARC_{inward} U $ARC_{intermediate}$ U $ARC_{outward}$ U $ARC_{auxiliary}$)

$$CS=\left\{\begin{array}{ll} 1) & \sum_{j:(i,j)\in AS} f_{ij}=1; \quad \forall i\in SAGVN; \text{Sending one unit flow into the network from each node in SAGVN} \\ 2) & \sum_{j:(j,i)\in AS} f_{ji}=M; \quad for \;\; i=SINK; \text{Receiving M units flow (the flows sent from the nodes in SAGVN set).} \\ 3) & \sum_{j:(i,j)\in AS} f_{ij}-\sum_{j:(j,i)\in AS} f_{ji}=0; \quad \forall i\in SJIN\cup SJOUT; \text{Flow balance at every Job - Input and Job - Output node.} \end{array}\right\}$$

3.2.3 The MCF-AGV model for the Automated Guided Vehicles Scheduling

Now the model for AGV Scheduling in container terminals is presented by the following definition:

$$\begin{cases} 1)\ D_{f_{ij}} = [0,1]; \text{for} \ \forall\ (i,j) \in (ARC_{inward} \cup ARC_{intermediate} \cup ARC \\ 2)\ D_{f_{ij}} = [1,1]; \text{for} \ \forall\ (i,j) \in ARC_{auxiliary} \end{cases}$$

Definition 4: A MCF-AGV model is defined as MCF-AGV=($G_{MCF-AGV}$, f, D, CS, FC) where $G_{MCF-AGV}$ = (GS, NPS, APS) is a graph for the MCF-AGV problem; f = a finite set of integer decision variables on AS, $f = \{ f_{ij} \mid (i, j) \in AS \}$; D = a function which determines a lower and upper bound for f; D: f→$N \times N$ (to be pulled out from APS); each element in D has :

and FC = $\sum_{j:(i,j)\in AS} c_{ij} f_{ij}$

The MCF-AGV model can be illustrated by Figure1 for two AGVs and four jobs. Solving the MCF-AGV model generates M paths, each of which commences from a vehicle node and terminates at the sink 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 1→3→4→9→10→11 and 2→5→6→7→8→11. 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.

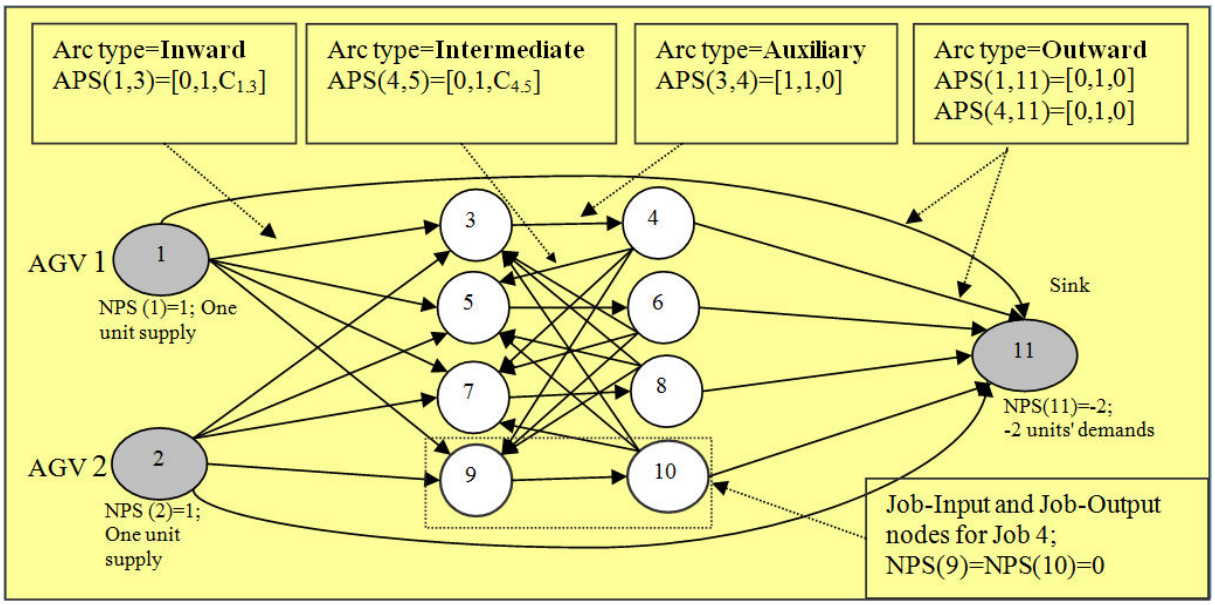

Figure 1. An example of the MCF-AGV model for 2 AGVs and 4 jobs

The MCF-AGV model has a huge search space and the solution should provide the optimal paths for each AGV from every node in SAGVN to SINK. As it was mentioned before, there are M+2×N+1 nodes and M+M×N+N×(N-1)+2×N arcs in the graph model where N and M specify the number of jobs and the number of AGVs in the problem, respectively. The number of paths in the search space is determined by the following equation:

$$Number\ of\ \ path = M + \binom{M}{1} \times N \times 1 + \binom{M}{2} \times (N-1) \times 2 + \ldots + \binom{M}{M} \times (N-M+1) \times M$$

The equation calculates every possible path in the search space. The first term represents paths from every node in SAGVN to SINK. The remaining terms shows the number of paths when 1, 2,.. , M (M ≤ N) AGVs, respectively, is selected to serve the jobs.

3. The Algorithm

The Network Simplex Algorithm (NSA) was chosen to tackle the problem. The main reasons to choose NSA are as follows:

- The area of development algorithm to tackle the MCF model by NSA is under-researched and offers fertile research opportunities for large scale problems. Several researches have been devoted on this matter [1, 0, 0, 0, 0, 0, 0] in the recent years.
- NSA is based on simple network operations. With simple network operations, the MCF model can be solved more than 100 times faster than equivalently sized Linear Programs. It is the fastest algorithm for solving the generalized network flow problem in practice [1].

This section reviews the algorithm. Every connected network has a spanning tree [1]. 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)∈L is greater than zero and at the same time the reduced cost for every arc (i,j)∈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 [0,0].

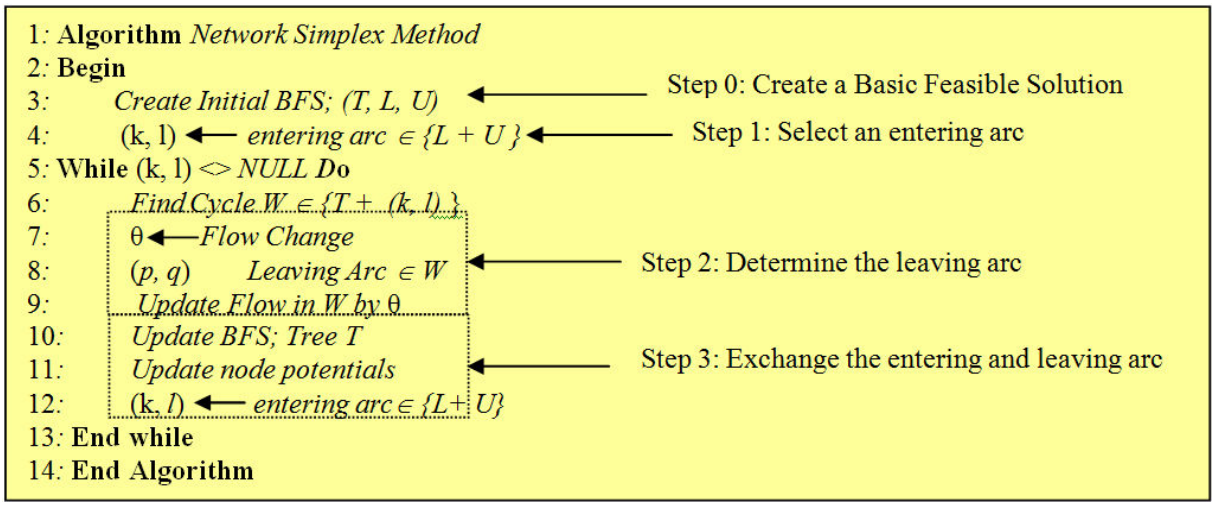

Figure 2. The Network Simplex Algorithm (NSA)

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 [0]. 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 $\theta = \min \{ f_{ij}$ 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 [1] 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 [4]. 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 [14]. Eppstein (1999) presented a clustering technique for partitioning trees and forests into smaller sub-trees or clusters [7]. 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 [13]. A general pricing scheme for the simplex method has been proposed by Istvan [11]. 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 [2]. They 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.

Istvan reviewed a collection of some known pricing schemes in the original simplex algorithm [0]. These schemes 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.

4. Experimental Results

This section presents the Experimental Results from the implementation and running the algorithm. As it was mentioned, the pricing rule or scheme to choose the entering arc in Step 1 determines the speed of algorithm. In the literature, the pricing rules are reviewed. 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 [0]. Kelly and Neill [0] 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. The block pricing scheme therefore was chosen. 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 [0], for large MCF problems. The number was set to 5% by the try and error.

To test the model and the algorithm, a hypothetical port was designed. The parameters in table 1 were used to define the port, the objective function, the number of vehicles and to generate the jobs. The software was implemented in Borland C++. Then, it was 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 algorithm has been drawn in Figure 3 and Figure 4, according to the number of jobs and the number of arcs, respectively. Also the power estimation for the curve has been shown on the figures. All experiments were run on a Pentium 2.4 GHz PC with 1 GMB RAM.

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 second
Travelling Time between every two points in the port (see Assumption 1)	Random between 1 and 100 seconds
Weight of waiting Times for the AGVs in the cost of the objective function	1
Weight of travelling Times for the AGVs in the cost of the Objective Function	5
P as the penalty in the costs of the objective function	10,000

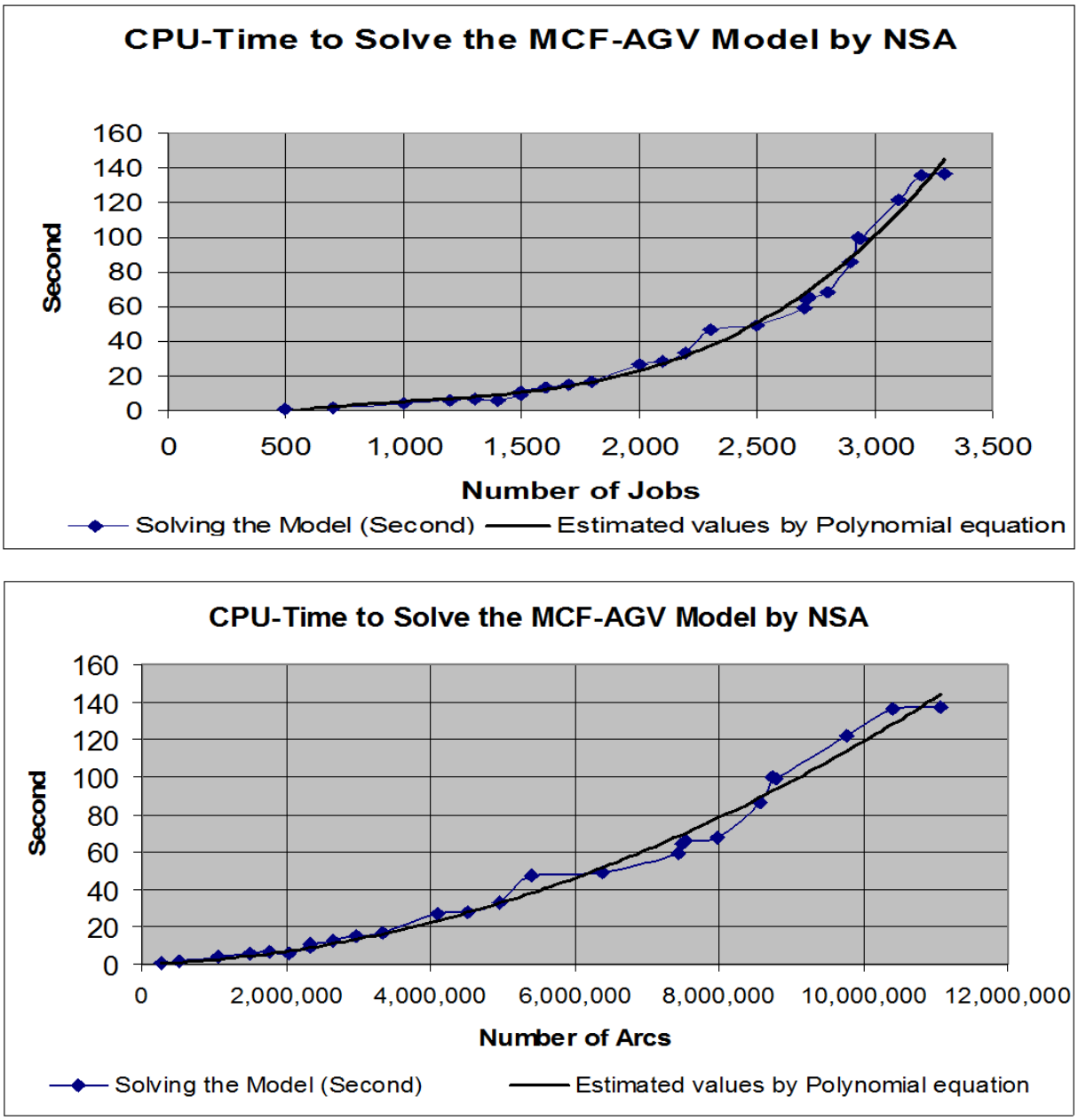

Figure 3. CPU-Time to solve the static problem by NSA, based on the number of jobs

From the experiments, the following observations are gotten:

Observation 1: NSA is fast and efficient.

Observation 2: NSA is run in polynomial time to solve the MCF-AGV model, in practice.

Observation 3: Although NSA is efficient and provides the optimal solution, it can only work on problem with certain limits in size.

There are two different types of iteration in NSA, degenerate and non-degenerate [2]. 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 degenerate iterations, a flow change of zero causes cycling. Observation 1 shows that cycling is rare in this experience.

In order to confirm that NSA is run in polynomial time to solve the MCF-AGV model (Observations 2), the complexity of the algorithm was estimated. The result showed that the CPU-Time required to tackle the problem, is a function with degree 3 of the number of jobs in the problem [0].

An important question related to Observation 3 is that how big (the number of vehicles, the number of jobs) of a problem can be solved by NSA within time t, a minute for example? The answer is that there is no limitation in NSA, theoretically. In practice, the answer is based on the platform and implementation. The limitations due to available memory to put the MCF-AGV model into (see section 2.2 for the number of nodes and arcs). The largest problem, which has been solved by the software, was a MCF-AGV model consists of 11,058,350 arcs (M=50; N=3,300). Based on this maximum number of arcs and the related formula, the number of vehicles (M) and number of jobs (N) can be had different values.

5. Conclusion

In this paper, a scheduling problem in the container terminal was presented and modelled. Then, the standard version of Network Simplex Algorithm (NSA) with the block pricing scheme was applied to the problem. To test the program, Random data were generated and fed to the model for fifty vehicles. The experiment shows that the Network Simplex Algorithm is efficient and run in polynomial time to tackle the problem. Although the algorithm is efficient and provides the optimal solution, it can only work on problems with certain limits in size. When the size of problem goes beyond the limit, incomplete solution methods should be used.

6. Refrences

[1] Ahuja R.K., Magnanti T.L., Orlin J.B., Prentice Hall (1993).

[2] Ahuja R.K., Orlin J. B., Sharma P., Sokkalingam P.T., Operations Research, (2002), Vol 30(3),141-148.

[3] Ahuja R.K., Orlin J.B., Giovanni M.S., Zuddas P, Management Science, (1999), Vol 45(10), 1440-1455.

[4] Andrew V.G., Journal of Algorithms, (1997) Vol 22(1), 1-29.

[5] Böse J., Reiners T., Steenken D., Voß S., Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, (2000), IEEE, 1-10.

[6] Cheng Y., Sen H., Natarajan K., Teo C., Tan K., Technical Report, National University of Singapore (2003).

[7] Eppstein D, In Proc. 5th ACM-SIAM Symposium. Discrete Algorithms, (1999) 160–166.

[8] Goldberg A.V., Kennedy, R. Technical Report, Stanford University (1993),.

[9] Helgason R., Kennington J., Handbook on Operations Research and Management Science Volume 7, North-Holland, Amsterdam, (1995), 85-133.

[10] Huang Y., Hsu W.J., CAIS, Technical Report, School of Computer Engineering, Nan yang Technological University, Singapore 639798 (2002).

[11] Istvan M, Technical Report, Department of Computing, Imperial College, London (2003).

[12] Kelly D.J., ONeill G.M.,Master Degree Dissertation, University College, Dublin (1993).

[13] Löbel A., Technical Report, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB) (2000).

[14] Masakazu M., Journal of Operations Research of Japan, (1999), Vol 43, 149-161.

[15] Murty K.G., Jiyin L., Yat-Wah W, Zhang C, Maria C.L. Tsang, Richard J. Linn. Decision Support System, (2002), Vol 39, 309-332.

[6] Patrick J.M., Dekker R.,Technical Report EI 2001-22, Erasmus University of Rotterdam, Econometric Institute (2003).

[17] Patrick J.M., Wagelmans P.M.,Technical Report EI 2001-33, Erasmus University of Rotterdam, Econometric Institute (2001).

[18] Patrick J.M., Wagelmans P.M.,Technical Report EI 2001-19, Erasmus University of Rotterdam, Econometric Institute (2001).

[19] Qiu L., Hsu W.-J., Huang S.-Y and Wang H. International Journal of Production Research, Taylor & Francis Ltd, (2002), Vol. 40 (3), 745-760

[20] Rashidi, H.R Department of Computer Science, University of Essex (2006).

[21] Rashidi H. & Tsang E.P.K Proceedings, 2nd Multidisciplinary International Conference on Scheduling: Theory & Applications (MISTA), New York, (2005), Vol 2, 677-69

[22] Sen H., Technical Report, HPCES Programme, Singapore-MIT Alliance (2001).

[23] Tsang E.P.K.,British Telecom Technology Journal, (1995) Vol.13 (1), Martlesham Heath, Ipswich, UK.

[24] Wook B.J., Hwan K.K., International Journal of management science, (2000) Vol 6 (2), pp 47-60.