A HEURISTIC FOR CAPACITY CONSTRAINED PRODUCTION SCHEDULING

by

MARGARIT BITON

B.S. in I.E., Bogazici University, 1984

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Master of Science

in

Industrial Engineering

Bogazici University

1986

A HEURISTIC FOR CAPACITY CONSTRAINED PRODUCTION SCHEDULING

APPROVED BY

Yard. Doç. Dr. Mahmut Karayel ....
(Thesis Supervisor)

Doç.Dr. Gündüz Ulusoy ....

Doç.Dr. Seyhan Tuğcu ....

DATE OF APPROVAL
6/6/1986

ACKNOWLEDGEMENTS

I greatfully acknowledge my gratitude to my thesis supervisor Yard. Doc. Dr. Mahmut Karayel with whose patience and understanding this work is realized. I appreciate his guidance and productive criticisms. It has been a pleasure to work with an advisor of an endless source of inspiration, foresight and enthusiasm.

I would like to express my deep admiration for Doc. Dr. Gunduz Ulusoy. I thank him for his interest and evaluation as a member of the thesis commitee.

I would also to thank my mother, Frida Zilbergleit, who has given me all the necessary help in my everydaylife, for her encouragement and support.

Finally, I should like to mention my appreciation and gratitude to my husband, Yusuf Biton, for supplying me with the necessary computers, assisting me in their usage and for taking over the painstaking typing process. Without him none of this would have been possible.

A HEURISTIC FOR CAPACITY CONSTRAINED PRODUCTION SCHEDULING

ABSTRACT

In recent years, due to high interest rates and strict monetary policy, reducing inventory holding costs have become a crucial issue. Minimizing inventory holding costs has been an emphatic objective in search for increased productive efficiency. In the determination of efficient production schedules, possibilities of time substitution of a limiting resource with th aim of having minimum inventory levels plays an important role.

This study proposes a model for scheduling production in single-stage multi-item capacity constrained production systems. Our algorithm schedules production as late as possible so as to minimize inventory holding costs. Items are independent and have external demands to be met. Two main classes of capacity constraints are considered. The algorithm deals with superoptimal (infeasible) solutions and moves backward in time in order to schedule production within the capacity bounds.

The exactness of the algorithm was tested and the computational results seem to be very favourable.

The algorithm developed in this thesis functions in determining efficient production schedules and in testing and evaluating scheduling designs and strategies as well.

SIĞASI KISITLANMIŞ ÜRETİM ÇİZELGELEMESİ İCİN BULGUSAL BİR YÖNTEM

ÖZET

Monetarist istikrar yöntemleri ile yönetilen günümüz Türk Ekonomisinde envanter maliyetlerinin önemi giderek artmaktadir. Üretkenlik veriminin artişinin sağlanmasi icin yapilan çalişmalarda envanter maliyetlerinin enazlanmasi, üzerinde önemle durulan bir amaç olagelmiştir. Verimli üretim çizelgelerinin belirlenmesinde sinirli kaynaklarin kullaniminin zamanla değiştirilmesi olanaklari, düşük envanter düzeylerinin amaçlanmasi ile birlikte etkin bir rol oynar.

Bu çalişma tek aşamali, cok ürünlü, siğasi kisitlanmiş üretim sistemlerinde üretimin çizelgelenmesi için bir model önermektedir. Oluşturulan algoritma envanter maliyetlerinin enazlanmasi amacina hizmet edecek şekilde üretimi mümkün oldugu kadar geç çizelgeler. Ürünler birbirlerinden bağimsiz olup, karşilanmalari gereken piyasa talepleri vardir. Iki temel siğa sinifi gözönünde tutulmuştur. Algoritma üsteniyi olursuz çözümlerden başlayarak zaman ekseninde geri giderek üretimin siga sinirlari içinde çizelgelenmesini sağlar.

Algoritmanin pekinligi denenip, uygun sonuçlar elde edilmiştir.

Bu tezde oluşturulan algoritma ile verimli üretim çizelgelerinin saptanmasinin yanisira cizelgeleme tasarim ve gengüdümleri de denenebilir ve değerlendirilebilir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION AND BACKGROUND

This study proposes a heuristic algorithm for scheduling production in multi-item single-stage capacity constrained systems.

The objective is to minimize the total holding cost. It is assumed that holding cost is nonnegative for each item and the demand is deterministic.

Two main classes of capacity constraints are considered. One class is time varying preallocated capacity bounds on the production rates of each item, namely bounds on the production rates.

The second class of capacity constraints involves environments in which items compete for limited production capacity, namely shared resource constraints.

The changeovers are modeled as set-up times that absorb capacity, and set-up costs are assumed to be negligible.

The algorithm begins with a late schedule which schedules production as late as possible without satisfying shared resource contraints, and starting at the last period of the planning horizon, moves backward in time to schedule production within the capacity bounds.

When set-ups are negligible, the problem reduces to linear programming and the algorithm is tested for exactness by solving 100 randomly generated problems of this kind. Computational tests showed that our proposed algorithm is a very effective tool in making capacity allocation decisions in production planning. It is very easy to understand and implement and seems to be very close to the exact optimal and much faster and simpler than solving large linear programming problems.

The algorithm is the same for both cases, namely where set-ups are negligible and not. The only difference betweeen these two cases occur in capacity consumption, so only a few calculations differ.

This problem can also be formulated as a minimal cost network flow problem. Such a representation is given and discussed in the last chapter.

Background

The production planning and scheduling literature is vast, and we review some of the relevant previous work.

In the single-stage, single-item lot sizing area Wagner and Whitin [1958] presented a shortest path solution for the single-stage uncapacitated dynamic lot sizing problem.Although their algorithm is efficient, other heuristic algorithms are developed by the practitioner. Florian and Klein [1971] considered the problem with constant capacity.Using the characterization of the extreme point schedules, they developed an efficient dynamic programming algorithm. Lambrect and Vander Eecken [1978] considered the variable capacity problem, and they developed an algorithm conditioning on the number of periods with zero production. Bitran and Yanesse [1982], and Karayel [1984] identified some special cases that are solvable by an efficient algorithm.

In the single-stage lot sizing with shared capacity area Manne [1958] provided a represantation of the individual schedules as columns of a linear program. This approach was improved upon by Lasdon and Terjung [1971], who employed large scale optimization techniques. Newson [1975] developed a dual heuristic which at first decomposes the problem into separate uncapacitated single-item problems. Recently, aggregation of items has been considered by many authors.

The hierarchical approach is becoming increasingly popular among researchers. In this approach the production planning and scheduling problem is partitioned into a hierarchy of subproblems. Graves [1982] employed duality and relaxation principles to incorporate feedback between the aggregate planning model, which determines the aggregate capacity and inventory levels, and the detailed scheduling model that determines lot sizes. Bitran,Haas, and Hax [1981] showed that in certain cases the hierarchical approach gives near optimal solutions.

In multi-stage lot sizing area Love [1972] showed that in the serial production system, if the costs are nonincresing in time, then the optimal solution must have the so-called nested property.
That is, if there is no demand (dependent plus external) in a given period, then there should be no production in that period. Using this property, he devised an alternate dynamic programming solution.

In the material requirements planning area McLain et al. [1981] developed a decomposition procedure to solve the capacity constrained MRP problem with fixed time lags between activities.

In the hierarchical production planning area Billington et al. [1983] proposed the method of product structure compression in order to reduce the problem size and partially aggregate the many items that are linked. Gabbay [1979] devised an aggregation/disaggregation procedure for serial production lines in which items have to go through the same set of production facilities which are capacitated. Bitran et al.[1982] analyzed a two-stage production system using hierarchical planning concepts.

II. A HEURISTIC APPROACH TO A MULTI-ITEM SINGLE-STAGE SCHEDULING PROBLEM

2.1. Formulation and Notation

In this section the production scheduling problem of this thesis is formulated, and the basic assumptions of our approach are stated.

The production activity has a dynamic structure represented by a finite number of time periods, $t=1,2,\ldots,T$.

The production process is defined by many activities, each with a distinct output, $i=1,2,\ldots,N$. All of these items are finished goods with external demands only. Thus we consider the problem of scheduling the production of N independent items over a time horizon of T periods with given demand levels to be filled. The items are coupled through the use of the same limiting resource. (e.g.man-hours or machine-hours).

At time period t, z_{it} units of item i are produced. Together with the incoming inventory of item i, I_{it-1}, z_{it} is distributed among the ending inventory of item i, I_{it} and the external demand for item i, u_{it}.
It is assumed that $I_{i0}=I_{iT}=0$ for all i=1,2,.....,N. Another assumption is that the demand data are deterministic.

Each time an item is produced some preparation (set-up) is necessary before production starts. The model assumes that whenever there is production, there is a set-up. Set-ups are represented by logical variables δ_{it} where

$$\delta_{it} = \begin{cases} 1 & \text{if } z_{it} > 0 \\ 0 & \text{otherwise} \end{cases}$$

The set-up and the production of one unit of item i require d_i and b_i units of resource respectively, for which items compete (for their set-up and production) and which has a capacity limit, c_t, through time.

At time period t, item i has a preallocated capacity bound, $\bar{z}_{it}$, on its production rate.

The per unit cost of keeping stock of item i at the end of time period t is denoted by h_{it}. It is assumed that the holding cost is nonnegative for each item, and constant in time, that is $h_{it}=h_i$ for all i=1,2,.....,N in all t=1,2,......,t.

Set-up costs are assumed to be negligible.

Although in most production environments the single-stage assumption does not hold, the production system may be modeled as a single stage if any of the following conditions are satisfied. First, there may be a bottleneck stage in a multi-stage environment. Secondly, by a prior aggregation rule several stages may have been aggregated into a single stage. Finally, deadlines may have been set for intermediate product outputs in a multi-stage environment. Also assembly lines can be modeled as a single-stage.

Formally the single-stage multi-item production scheduling problem may be stated as follows:

$$\text{Min} \sum_{i=1}^{N} \sum_{t=1}^{T} h_i I_{it} \qquad (0)$$

s.t.

$$z_{it} + I_{it-1} - I_{it} = u_{it} \qquad (1)$$

$$\sum_{i=1}^{N} (b_i z_{it} + d_i \delta_{it}) \leqslant c_t \qquad (2)$$

$$z_{it} \leqslant \bar{Z}_{it} \qquad (3)$$

$$z_{it},\ I_{it} \geqslant 0 \quad ,\ \delta_{it} \in (0,1)$$

The objective of the problem is to minimize the total holding cost.

Constraint set (1) represents the flow balance equation for each item i in each period t. As our concern is single-stage systems, there is no interrelation between items (no item is necessary for the production of an other item). It means that we deal only with external demands. Constraint set (1) ensures that the production of item i in period t, together with its incoming inventory is distributed among its ending inventory and its external demand in time period t.

Constraint set (2) represents the time varying shared capacity constraints. In most production systems several activities compete for the use of each resource which is to be shared according to some rule. We denote these systems by the term capacity-shared systems. In capacity-shared systems the allocation of available resources among different activities has to be done explicitely. In the model the existence of only one shared capacity is assumed.

Constraint set (3) represents the preallocated capacity constraints, which are time varying capacity bounds on the production rates of each item. These resources are preallocated to items, and are represented as simple upper bounds on the production rate of each item.

Table 1.1.

Summary of Basic Notation

N	number of items
T	number of periods in the time horizon
z_{it}	quantity of item i produced in time period t
I_{it}	quantity of item i in inventory at the end of time period t
u_{it}	external demand for item i at the end of period t
b_i	quantity of resource needed per unit output of activity i
δ_{it}	logical set-up variable for item i in the period t
d_i	quantity of resource needed for a set-up of activity i
h_{it}	holding cost of keeping a unit of inventory of i at the end of period t
c_t	availability of shared resource in time period t
Ξ_{it}	availability of preallocated resource of item i in period t
u^c_{it}	cumulative demand for item i at the end of period t

2.2. A Fundamental Insight

When set-ups are negligible, then the problem reduces to the below given problem denoted by P1:

P1 $$\text{Min} \sum_{i=1}^{N} \sum_{t=1}^{T} h_i I_{it} \qquad (0)$$

s.t.

$$z_{it} + I_{it-1} - I_{it} = u_{it} \qquad (1)$$

$$\sum_{i=1}^{N} b_i z_{it} \leqslant c_t \qquad (2)$$

$$z_{it} \leqslant \bar{z}_{it} \qquad (3)$$

$$z_{it},\ I_{it} \geqslant 0$$

Note that after a slight transformation in P1, everything can be in units of shared resource (as in P1 preallocated capacities are in units of external demand).

This transformation can be achieved as follows:

1) Multiply constraint set (1) with b_i

=> $b_i z_{it} + b_i I_{it-1} - b_i I_{it} = b_i u_{it}$

2) Multiply consraint set (3) with b_i

=> $b_i z_{it} < b_i \bar{z}_{it}$

3) Divide h_{it} by b_i

Now let's make the following definitions:

$z'_{it}=z_{it}b_i$

$I'_{it}=I_{it}b_i$

$h'_{it}=h_{it}/b_i$

$u'_{it}=u_{it}b_i$

$\bar{\Sigma}'_{it}=b_i\bar{\Sigma}_{it}$

and consider the new decision variables as z'_{it}, I'_{it}

the new inventory holding cost as h'_{it}

the new external demand as u'_{it}

the new bounds of preallocated capacities as $\bar{\Sigma}'_{it}$.

After the transformation problem P1 becomes and is equal to P2, as given below:

P2

$$\text{Min} \sum_{i=1}^{N} \sum_{t=1}^{T} h'_i I'_{it} \qquad (0)$$

s.t.

$$z'_{it}+I'_{it-1}-I'_{it}= u'_{it} \qquad (1)$$

$$\sum_{i=1}^{N} z'_{it} \leqslant c_t \qquad (2)$$

$$z'_{it} \leqslant \bar{\Sigma}'_{it} \qquad (3)$$

$$z'_{it},\ I'_{it} \geqslant 0$$

Now everything is in resource units.

Note that the objective function of P2 can also be written as :

$$\text{Min} \sum_{i=1}^{N} \sum_{t=1}^{T} h_i I_{it}$$

, which is the objective function of P1, as $h'_i I'_i = (h_i / b_i)(I_{it} / b_i) = h_i I_{it}$

<u>Proposition:</u>

After this transformation it can be shown that the problem data of P2 may be changed as follows:

$$c_t = \min\{c_t ; \sum_{i=1}^{N} \bar{z}'_{it}\}$$

<u>Proof:</u>

In an arbitrary period t, the maximum amount of resource allocation to an item i (or resource consumption of item i) can't exceed its upperbound $\bar{z}'_{it}$. Then the maximum amount of cumulative resource consumption, denoted by $\bar{Z}'_t$ is the sum of the individual resource consumption bounds, that is

$$\bar{Z}_t = \sum_{i=1}^{N} \bar{z}_{it}$$

Two cases can occur:

Case I: $c_t \geq \bar{Z}_t$

If $c_t > \bar{Z}_t$, then there is always a slack capacity of value $c_t - Z_t$, as the bound on the cumulative resource consumption is $\bar{Z}_t$ (otherwise there would be preallocated capacity violation).

Thus c_t is no more an upperbound for shared capacity constraint, which will be always inactive. The actual upperbound for shared capacity constraint is $\bar{\bar{Z}}_t$.

So when case I occurs the problem data is changed as: $c_t \longrightarrow \bar{\bar{Z}}_t$.

Case II: $c_t < \bar{\bar{Z}}_t$

In this case total resource consumption is always below c_t so that shared capacity constraints are not violated which are always active and c_t's are the upper bounds for them.

The intersection of the two cases is the general case by which the problem data is changed as:

$$c_t = \min \{ c_t , \sum_{i=1}^{N} \bar{Z}'_{it} \}$$

2.3. Early and Late Schedules

In this section two important schedules, namely early and late schedules are defined and their respective algorithms given. In both algorithms it is assumed that set-ups are negliglible and both algorithms are for systems, having only preallocated capacities.

2.3.1. Early Schedule

Early schedule is the schedule that would schedule production as early as possible, hence results with the highest inventory levels.

The algorithm to find the early schedule solution is summarized below

<u>Early Schedule Algorithm:</u>

Step 0: Let t=1, $Y_i = u^c_{iT}$ for all i=1,2,...,N.

where u^c_{iT} is the cumulative demand of item i up to T.

Step1: For i=1,2,.....,N do $ES_{it}=\min\{Y_i , \bar{z}_{it}\}$

$Y_i = Y_i - ES_{it}$

where ES_{it} is the early schedule production of item i in period t.

Step2: Let t=t+1 . If $t \leq T$ go to Step 1.

2.3.2.Late Schedule

Late schedule is that schedule, that would schedule production as late as possible. In terms of cumulative production this means that with the given capacities a late schedule algorithm that would schedule production as late as possible would result in minimum inventory.

The algorithm to find the late schedule is summarized below.

Late Schedule Algorithm:

Step 0: Let $t=T$, $Y_{it}=0$ for all $i=1,2,\ldots,N$.

Step 1: For $i=N,N-1,\ldots,1$ do

$LS_{it}=\min\{Z_{it},\ u^c_{it}-u^c_{it-1}-Y_{it}\}$

$Y_{it-1}=Y_{it}+LS_{it}$

where LS_{it} is the late schedule of item i in period t.

u^c_{it-1} is the cumulative production of item i up to t-1.

Step 2: Let $t=t-1$. If $t>0$ go to Step 1.

2.3.3. Different View

In this part, using early and late schedules the problem is seen from a different point of view.

Define,

$\sum_{v=1}^{t} SC_v$ = Cumulative production with shared and preallocated capacities

$\sum_{v=1}^{t} ES_v$ = Cumulative production with early schedule

$\sum_{v=1}^{t} LS_v$ = Cumulative production with late schedule

For every t,

$$\sum_{v=1}^{t} ES_v \geqslant \sum_{v=1}^{t} SC_v \geqslant \sum_{v=1}^{t} LS_v$$

Because, as explained before early schedule results in maximum inventory meaning that in maximum cumulative production; late schedule results in minimum inventory, hence in minimum cumulative production. Thus cumulative early schedule constitutes an upper bound and cumulative late schedule a lower bound on the cumulative production of any schedule within the shared and preallocated capacities.

The variation of the cumulative production through time can be seen in Figure 2.1.

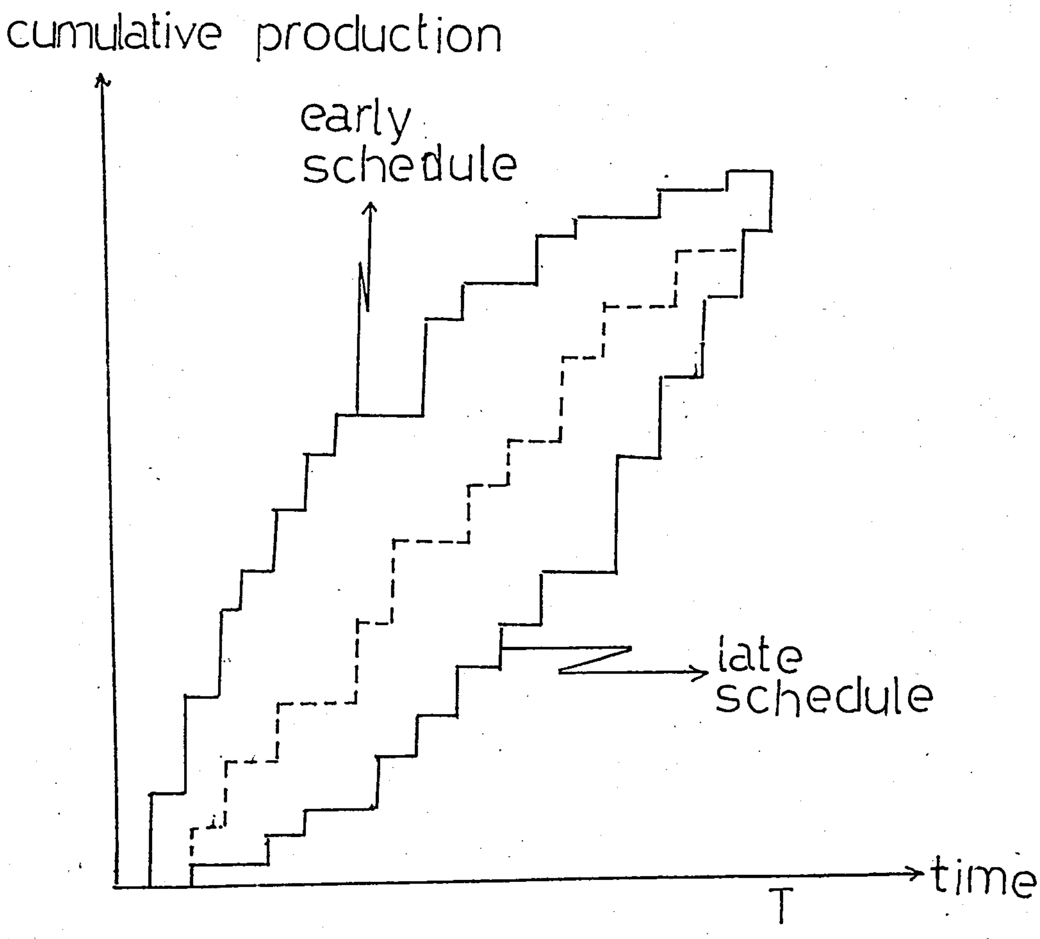

Figure 2.1

So, from another point of view our problem can be stated as follows:
Find a path on the graph showing cumulative production through time (like the dotted one on Figure 2.1.) which is between the cumulative early and late schedules and minimizes

$$\sum_{i=1}^{N} \sum_{v=1}^{t} h_i (z_{iv} - u_{iv})$$

Subject to the shared capacity constraint.

2.4. The Algorithm

A heuristic algorithm is developed which initially schedules production as late as possible. Thus the initial solution is obtained with the late schedule algorithm. This solution is superoptimal, but may be infeasible as shared capacity costraints were not considered (recall that in the late schedule algorithm only the preallocated capacity constraints ara considered). Then to achieve feasibility, starting at the last period and moving backward in time iteratively, the production is shifted earlier by saving set-ups and increasing inventory levels.

In each iteration only the production of a single item is shifted earlier in a way that preallocated capacity constraints are not violated. And it is that item which has the least cost increase, ΔCost, per capacity change, ΔCapacity, in the current period. If the production of an item is shifted earlier, in the current period capacity consumption decreases, hence capacity availability increases (ΔCapacity) at a cost of increasing inventory levels of that item and hence a higher total cost(ΔCost).

ΔCost/ΔCapacity ratio is affected by:

a)The cost structure: The effect of the cost structure is very obvious as the total cost consists of each individual holding costs.

and

b)The preallocated capacity bounds, because the tightness of them in the later periods necessitates the production shift to earlier periods. Thus time length between production and usage periods will increase, as a result the inventory holding cost will increase also.

and

c)The set-ups, because if the production of an item in a period drops to zero, then set-up savings occur, resulting in extra capacity availability.

Whether set-ups are negligible or not the algorithm is the same. The only change occurs in some calculations used in the determination of ΔCost/ΔCapacity ratio.

The algorithm can be summarized as follows:

Step 0: In this step the production plan is set to producing as late as possible with the given preallocated capacities, that is the solution is initialized to late schedule. Then the shared capacity violations, cv_t, implied by this schedule are calculated. If there are no capacity violations, then the late schedule is optimal. Otherwise current time is set to the last period of the planning horizon and will be continued. Formally:

(0.1): Calculate late schedule (with the late schedule algorithm given in 2.3.2.), z^{late}, and set $z^{current} = z^{late}$

(0.2): Find $cv_t = c_t - \sum_{i=1}^{N} (b_i z_{it} + d_i \delta_{it})$

(0.3): Set $t=T$

Step 1: Moving backward in time,we check if the capacity in this period is violated. If the capacity is not violated, then we move on to an earlier period. Otherwise a ratio test is performed to determine which item to time subtitute in order to resolve infeasibilities. The item that would be is the one that yields the least additional cost per unit of infeasibility that is resolved. Thus, the item with the minimum ratio is found and its schedule updated, by decreasing its production in the current period and increasing it in early periods (of course by the same amount). Formally:

(1.1) If $cv_t \geqslant 0$, do Step 2.

(1.2) Calculate ΔCost/ΔCapacity ratio for each item. Find item j with the minimum ΔCost/ΔCapacity ratio, adjust schedule of item j.

Step2: If there are no more infeasibilities in this time period then we go back in time, otherwise we shift the schedule of some other item to earlier time periods. Formally:

(2.1): If $cv_t < 0$, do Step 1.

(2.2): $t = t-1$; if $t > 1$, do Step 1.

(2.3): If $t=1$ then check whether $cv_t < 0$, if it is less than zero conclude that the problem is infeasible.

2.4.1. Resolving Infeasibilities

In this section, the determination of the ΔCost/ΔCapacity ratios, shifting process of the production and adjustments of schedules or formally Step 1 of the algorithm -where infeasibility resolvement occurs- will be explained.

It would be helpful to start with the explanation of the case with negligible set-ups in order to set the ground for the general case, namely the case when set-ups are not negligible.

2.4.1.1. Set-ups are negligible

Let the current period be t. When an item's, say item i, production in the periods preceding t equals to the preallocated capacity bounds, that is if

$$z_{ik}=\bar{z}_{ik} \quad \text{for all } k=t-1,t-2,\ldots\ldots\ldots,1.$$

then that item's production can't be shifted earlier, because otherwise the preallocated constraints in the previous periods would be violated. And all items with this feature have a ΔCost/ΔCapacity ratio as infinity and all form a set defined as MP.

Also note that if in any period t, the shared capacity violation is cv_t and item i is not an element of the set MP, the minimum amount of production by which the production of item i can be shifted earlier is defined as a_i, where

$$a_i=\min\{z_{it},\ cv_t/b_i\}$$

Because as b_i units of resource are necessary per unit output for item i, cv_t units of resource are used for the production of cv_t/b_i units of item i. So, if the production of item i is shifted by cv_t/b_i units earlier, that is if the production of item i in period t reduces by cv_t/b_i units, then the infeasibility in period t is resolved as a result of the reduced resource consumption (by an amount of cv_t). But,meanwhile the current production should also be considered. The amount by which the production of item i is shifted earlier, can not exceed its current production z_{it}. Otherwise the production of item i in period t will be a negative amount, what is meaningless and also infeasible, as nonnegativity constrains are violated. More than the currently produced amount can not be shifted earlier (as the algorithm goes backward through time, current period's production must be feasible, because the algorithm does only one pass over time).

The algorithm starts with an infeasible solution and aims at achieving feasibility at minimum cost, and feasibility is achieved in each consecutive period. Only if in any period t, the shared capacity constraints are no more violated the algorithm proceds with time t-1. And note that, in earlier periods the shared capacity violations increase, as production is shifted earlier. But from the beginning on (late schedule) preallocated capacities are never violated, therefore the amount to be shifted earlier is determined due to the preallocated capacities in early periods. Thus, the production in any early period can be increased up to the preallocated capacity bound in that period. Then, $de_{i,k}$ is defined as the maximum amount by which the production of item i can be increased in a period k preceding the current period t.

Where $de_{ik} = \min\{a_i, \bar{z}_{ik} - z_{ik}\}$, for all k=t-1.t-2......,1

Recall that the maximum amount by which item i can be shifted earlier was defined as a_i. We aim at increasing the production of item i in any period k (k<t) by a_i; but in order not to violate the preallocated capacities, we are allowed to increase it only by the amount $\bar{z}_{ik} - z_{ik}$, as it is the slack of the preallocated capacities.

The actual amount of production (of item i in period t) that is shifted earlier is defined as te_{it}, where

$$te_{it} = \sum_{k=1}^{t-1} de_{ik}$$

The reason is as follows: The amount of production that is shifted earlier (i.e. is not produced currently), must be produced earlier and therefore carried as inventory, because the demand must be met. Then the amount of the current period's production that is shifted earlier equals to the increase in the incoming inventory, which in turn must be equal to the amount of production that can be increased in periods preceding current period t. And as $te_{i,t}$ is the total production increase before current period t (since it is the sum of the individual production increases, i.e. production increases in each time period from t=t-1 to t=1), then it is also the amount of the current production that is shifted earlier.

After having given the necessary definitions and explanations, the mechanism of the infeasibility resolvement (or step 1 of the algorithm) can be illustrated in detail.

Step 1 in detail will be formally as follows:

(1.1.1) Calculate $cv_t = c_t - \sum_{i=1}^{N} b_i z_{it}$

If $cv_t < 0$ then check whether the number of elements in set MP equals N; if it equals then conclude that the problem is infeasible.

Otherwise continue with Step 1.2.

If $cv_t \geq 0$, do Step 2.

(1.2.1) Set i=1

(1.2.2) If $i \in MP$ then set $(\Delta Cost/\Delta Capacity)_i$ to a very large number and go to 1.2.7. If $z_{it}=0$, again set $(\Delta Cost/\Delta Capacity)_i$ to a very large number and go to 1.2.7.

(1.2.3) Set $a_i = \min\{z_{it}, cv_t/b_i\}$

$te_{it}=0$, $(\Delta Cost)_i=0$

(1.2.4) Set k=t-1

(1.2.5) $\Delta Cost/\Delta Capacity$ ratio of each item is calculated as a result of the following calculations:

(1.2.5.1) If $(\bar{z}_{ik} - z_{ik})=0$ go to 1.2.6

otherwise $de_{ik} = \min\{a_i, \bar{z}_{ik} - z_{ik}\}$

(1.2.5.2) $(\Delta Cost)_i = (\Delta Cost)_i + h_i * de_{ik} * (t-k)$

(1.2.5.3) $te_{it} = te_{it} + de_{ik}$

(1.2.5.4) $(\Delta Capacity)_i = te_{it} * b_i$

(1.2.5.5) Find $(\Delta Cost / \Delta Capacity)_i$

(1.2.5.6) $a_i = a_i - de_{ik}$. If $a_i = 0$ then go to 1.2.7 otherwise continue with 1.2.6.

(1.2.6) $k=k-1$. If $k \geqslant 1$ then go to 1.2.5.

(1.2.7) $i=i+1$. If $i \leqslant N$ then go to 1.2.2.

(1.2.8) Find item j with the minimum ΔCost/ΔCapacity ratio.

$$j = \underset{i}{\operatorname{argmin}} (\Delta Cost / \Delta Capacity)_i$$

(1.2.9) Schedule of item j is adjusted as follows:

(1.2.9.1) $z_{jt} = z_{jt} - te_{jt}$

(1.2.9.2) Set $k=t-1$, $f_j=0$

(1.2.9.3) $z_{jk} = z_{jk} + de_{jk}$

$f_j = f_j + \bar{z}_{jk} - z_{jk}$

(1.2.9.4) If $k \geqslant 1$ go to 1.2.9.3 otherwise continue

(1.2.10) If $f_j=0$ then item j joins the MP set.

In 1.1.1. the MP set is checked for its number of elements, because if there is resource violation and all items are in the MP set, then no item exists of which the production can be shifted earlier, as there is no slack in the preallocated capacities in previous periods and the problem is infeasible.

From 1.2.1 to 1.2.6 ΔCost/ΔCapacity ratio of each item is calculated by determining the set of periods (denoted with k) to which the production is shifted and the amount of production increase in these periods. If the production is shifted by de_{ik} amount to period k, then $(\Delta Cost)_i = h_i * de_{ik} * (t-k)$, because an inventory of de_{ik} units is carried (t-k) periods (the production is in k and usage is in t) at a cost of h_i per unit.

$(\Delta Capacity)_i$ in time period t is equal to the product of te_{it} and b_i because if the production of item i is shifted by an amount of te_{it}, then in the current period t, shared resource consumption decreases by $te_{it} * b_i$.

As seen in (1.2.5.6), whenever production increases by an amount de_{ik} in a preceding period k, a_i is adjusted (decreased by de_{ik}), as it is the amount of production that we want to shift. Thus, when a_i reaches zero, we no more need to check former periods for production increase as we shifted production as much as we wanted.

From 1.2.9.1 to 1.2.9.4, the schedule of the item with the minimum ΔCost/ΔCapacity ratio, j, is adjusted. In 1.2.9.1. the production of item j in the current period is decreased by te_{it} and in 1.2.9.2 to 1.2.9.4 its production is increased in the preceding periods. Also it is checked whether item j will be an element of the set MP, by calculating the sum of the slack in the preallocated capacities in the previous periods. If this sum, f_j, is zero, then item j joins the MP set.

After the explanation of the concepts and the main logic, we can proceed with the general case.

2.4.1.2. The General Case

As previously stated, the only change between the general case and the case with negligible set-ups occurs in the ΔCost/ΔCapacity structure, as a result of set-up savings.

First set-up savings are explained and then ΔCost/ΔCapacity structure will be analyzed.

Again let the current period be t and the item under consideration be i.

(1) Set-up Savings

When the production of item i in t is adjusted to zero, then no more resource is needed for the set-up of it in t as set-ups are preparations made for production. So set-up savings occur. Therefore, we need variables which indicate the resource amounts necessary in the set-ups for the productions of items in time periods, and which are updated as any schedule adjustment occurs. And variables s_{it}'s stand for the above mentioned set-up necessities. When the production of item i in period t drops to zero, then s_{it} drops to zero as well. s_{it}'s vary through time, where d_i's are always constant. Note, that at the beginning $s_{it}=d_i$ for all t=1,2,.....,T. When the schedule of an item is adjusted such that its production is reduced to zero, then its variable set-up necessity, s_{it}, is also adjusted to zero.

There are mainly two cases, which can be summarised as follows:

Case 1) $te_{it} < z_{it}$

The amount of production to be shifted earlier is less than the current production.

After the schedule adjustment (recall that schedule of item i is adjusted only if it has the least ΔCost/ΔCapacity ratio), the production in t reduces by an amount of te_{it} and becomes $z_{it} - te_{it}$, that is $z_{it} \longrightarrow z_{it} - te_{it}$.

As there is production, there is need for set-up. Thus s_{it} does not change. Case 1 is stated formally as:

If $te_{it} < z_{it}$ then s_{it} does not change for all $i=1,2,\ldots,N$; and for all $t=1,2,\ldots,T$.

Case 2) $te_{it} = z_{it}$

If item i is chosen to be the item with the minimum ΔCost/ΔCapacity ratio, then its production in time t is adjusted to zero. As no production of it takes place, there is no need for set-up and s_{it} drops to zero.

Case 2 is stated formally as:

If $te_{it} = z_{it}$, then $s_{it} = 0$ for all $i=1,2,\ldots,N$ and for all $t=1,2,\ldots,T$.

(2) ΔCost/ΔCapacity Structure

There are mainly three cases: A, B and C

These can be summarized as follows:

Case A: If $te_{it}=0$ then ΔCost/ΔCapacity= ∞

Case B: If $te_{it}<z_{it}$ then

$(\Delta Cost/\Delta Capacity)_i=(\Delta Cost)_i/(te_{it}*b_i)$

Case C: If $te_{it}=z_{it}$

Case C has 2 subcases: C1 and C2

C1. If $z_{it}*b_i=cv_t$ then

$(\Delta Cost/\Delta Capacity)_i=(\Delta Cost)_i/cv_t$

C2. If $z_i t*b_i<cv_t$

C2 again has 2 subcases: C2i and C2ii

C2i: If $z_{it}*b_i+s_{it}<cv_t$, then

$(\Delta Cost/\Delta Capacaity)_i=(\Delta Cost)_i/(te_{it}*b_i+s_{it})$

C2ii: If $z_{it}*b_i+s_{it}\geqslant cv_t$, then

$(\Delta Cost/\Delta Capacity)_i=(\Delta Cost)_i/cv_t$

Case A is the case where item i is an element of the MP set and therefore its production can not be shifted earlier.

Case B is the case where the production of the current period can not be shifted earlier as a whole (only a portion of it can be shifted) because of the preallocated capacities, that is

$$\sum_{k<t} \Xi_{ik}-z_{ik}<z_{it}$$

The portion of the current period's production can be sufficient to resolve the infeasibility or not. If it is not sufficient, then resource consumption in period t will be reduced only by an amount of $(\Delta Capacity)_i = te_{it} * b_i$, as the production of item i in t can only be reduced by te_{it}

If it is sufficient then in t, the change in the capacity, ΔCapacity, equals to the infeasibility, cv_t.

Case C is the case where the production of the current period can be shifted earlier as a whole, that is

$$\sum_{k<t}^{t} Z_{ik} - Z_{ik} \geq Z_{it}$$

(Or the sum of the slacks in the preallocated capacities in the periods preceding t is larger than the current periods production amount)

In Subcase C1 the violation is resolved by shifting the current production (of item i in period t) earlier, so the change in the capacity, ΔCapacity, equals to the infeasibility cv_t, as resource availability increases by cv_t.

Whereas in Subcase C2, the current period's production shift is not enough to resolve the violation. But, when the amount of production in period t (as a whole) is shifted earlier, that is when item i is not produced i period t, then no resource is necessary for set-up, as set-ups are preparations done for production. As a result, set-up saving occur.

Then, the sum of the resource units used for the current period's production and for its set-up, namely, $b_i te_{it}+s_{it}$ (or $b_i z_{it}+s_{it}$ as $te_{it}=z_{it}$) is either sufficient (Case C2ii) to resolve the violation, i.e. $b_i z_{it}+s_{it} \geq cv_t$ or not enough (Case 2i) to resolve it, i.e. $b_i z_{it}+s_{it}<cv_t$. ΔCapacity in Case C2ii is cv_t, as the change in the capacity availability or consumption is by cv_t; it is $b_i z_{it}+s_{it}$ in Case C2i as only $b_i z_{it}+s_{it}$ units of resource is saved.

Then in the general case Step 1 of the algorithm in detail is formally given as:

(1.1.1) Set $s_{it}=d_i$ for all $i=1,2,\ldots\ldots,N$; $t=1,2,\ldots\ldots,T$

(1.1.2) Calculate $cv_t=c_t-\sum_{i=1}^{N}(b_i z_{it}+s_{it}\delta_{it})$

If $cv_t<0$ then check whether the number of elements in the set MP equals to N; if it equals then conclude that the problem is infeasible. Otherwise continue with Step 1.2.

If $cv_t \geq 0$, do step 2

(1.2.1) Set i=1

(1.2.2) If $i \in MP$ then set $(\Delta Cost/\Delta Capacity)_i$ to a very large number and go to Step (1.2.7).

Also, if $z_{it}=0$, set $(\Delta Cost/\Delta Capacity)_i$ to a very large number and go to Step (1.2.7)

(1.2.3) Set $a_i = \min\{z_{it}, cv_t/b_i\}$

$te_{it}=0$, $(\Delta Cost)_i=0$

(1.2.4) Set k=t-1

(1.2.5) ΔCost/ΔCapacity ratio of each item is calculated by doing:

(1.2.5.1) If $(\bar{z}_{ik}-z_{ik})=0$ goto (1.2.6); otherwise set $de_{ik}=\min\{a_i, \bar{z}_{ik}-z_{ik}\}$

(1.2.5.2) $(\Delta Cost)_i=(\Delta Cost)_i+h_i*de_{ik}*(t-k)$

(1.2.5.3) $te_{it}=te_{it}+de_{ik}$

(1.2.5.4) Check whether

B) $te_{it}<z_{it}$ or

C) $te_{it}=z_{it}$ is

If (B) occurs then $(\Delta Capacity)_i=te_{it}*b_i$

If (C) occurs then check again whether

C1) $z_{it}*b_i=cv_t$ or

C2) $z_{it}*b_i<cv_t$ is

If (C1) occurs then $(\Delta Capacity)_i=cv_t$

If (C2) occurs then check whether

C2i) $z_{it}*b_i+s_{it}<cv_t$ or

C2ii) $z_{it}*b_i+s_{it}\geqslant cv_t$ is

If (C2i) occurs then

$(\Delta Capacity)_i=z_{it}*b_i+s_{it}$

If (C2ii) occurs then $(\Delta Capacity)_i=cv_t$

(1.2.5.5) Find $(\Delta Cost/\Delta Capacity)_i$

(1.2.5.6) $a_i=a_i-de_{ik}$. If $a_i=0$ then go to (1.2.7), otherwise continue with (1.2.6)

(1.2.6) $k=k-1$. If $k \geqslant 1$ then go to (1.2.5)

(1.2.7) $i=i+1$. If $i \leqslant N$ then go to (1.2.2)

(1.2.8) Find item j with the minimum ΔCost/ΔCapacity ratio

$$j = \underset{i}{\text{argmin}}(\Delta Cost/\Delta Capacity)_i$$

(1.2.9) Schedule of item j is adjusted as follows:

(1.2.9.1)

(1.2.9.1.1) If $te_{jt}<z_{jt}$ go to (1.2.9.1.3)

(1.2.9.1.2) If $te_{jt}=z_{jt}$ then set $s_{jt}=0$

(1.2.9.1.3) $z_{jt}=z_{jt}-te_{jt}$

(1.2.9.2) Set $k=t-1$, $f_j=0$

(1.2.9.3) $z_{jk}=z_{jk}+de_{jk}$

$f_j=f_j+\bar{z}_{jk}-z_{jk}$

(1.2.9.4) If $k \geqslant 1$ go to (1.2.9.3)

(1.2.10) If $f_j=0$ then include item j to the MP set.

Note, that the general case differs from the case where set-ups are negligible only in step (1.2.5.4) and the general case has 2 additional substeps under (1.2.9.1), namely (1.2.9.1.1) and (1.2.9.1.2).

In (1.2.9.1.2) s_{jt} is adjusted to zero as production in period t is reduced to zero.

2.5.Recapulitation and Optimality Analysis

In this part, the logic of the algorithm is analyzed and reasons for its unexactness are given and shown on an example problem.

Our heuristic is mainly based on the following two ideas:

1) Producing as late as possible minimizes holding costs.
2) In order to achieve feasibility, if it is necessary to decrease the production in a certain period, then shift the production of that item earlier that contributes the least additional holding cost per increase in resource availability resulting from its resource consumption decrease in that period.

Our algorithm is similar to the dual simplex method, because both deal directly with superoptimal solutions. But our algorithm differs from the dual simplex method due to the fact that dual simplex method moves toward an optimal solution by striving to achieve feasibility, whereas our algorithm moves toward a feasible solution; that may not necessarily be the optimal solution.

Our algorithm reaches feasibility by performing necessary adjustments so that feasible production schedules are achieved in each individual time period starting at the last period in the planning horizon and moving backward.

As stated before,the problem is reduced to linear programming when set-ups are neglected. In that case, both solutions, namely the solution found using our algorithm and the optimal solution found using any linear programming method can be compared. The result of this comparison is that our algorithm is not exact. And the reasons for this are the myopic structure of the algorithm and the existence of the preallocated capacity constraints.

Myopicity alone does not contitute a reason for unexactness, but myopicity and preallocated capacities together make the algorithm unexact. Karayel [5] used the same heuristic for the problem in which only the shared capacities were considered (preallocated capacities were not taken into account) and obtained the result that the algorithm is exact.(Of course he used another late schedule algorithm to initilize the solution as preallocated capacities were not considered. His late schedule was obtained by setting the production of each item to their demand by ignoring the capacity constraints. And as his problem was multi-stage, the demand was the sum of dependent and external demands. But the main logic of the algorithm is the same.)

Now, reasons for unexactness are discussed in detail. In our algorithm, in each iteration only a single period is considered. But decreasing the production of an item in the current period can be the most economical choice when previous periods are not considered, but in the long run total holding cost may be lower when another item is chosen.

The decision taken in a certain time period also effects other periods, as a result of preallocated capacities. Because, in a certain period t, the capacity violation may be more than the violation in t-1 and the preallocated capacity bounds may be of such nature that cheaper items have very tight preallocated capacity constraints. In such cases to resolve the violation in t-1, the cheaper items' production shifts are not enough as their productions were shifted earlier to resolve the violation in t and have very little slack capacities in the periods preceding t. As a result,in the next iteration a large amount of production of expensive items must be shifted earlier. But, if in period t, the production of expensive items were shifted (in a less amount as, $cv_t < cv_{t-1}$), of course at a higher cost increase, then we could shift more of the cheaper items in t-1 and at a lower cost increase. Then the net difference between the higher cost in t and the lower cost in t-1 would be a lower cost.

In other words, as the algorithm is myopic, to reduce a less amount of violation the small amount of slack capacities of the cheaper items are used. But if they were used in the reduction of a larger amount of violation, the production of the expensive items were shifted in less amounts and the total cost would be less.

An Example Problem

Analyzing the above discussed matter with an example would be helpful and following single-stage multi-item production scheduling problem is given:

t	u_{1t}	u_{2t}	u_{3t}	$\bar{z}_{1t}$	$\bar{z}_{2t}$	$\bar{z}_{3t}$	C_t
1	1	2	3	3	30	30	80
2	2	4	5	2	10	10	30
3	3	6	7	3	5	5	5
4	4	8	9	4	5	5	10

i	h_i	b_i
1	1	1
2	2	1
3	3	1

(Set-ups are negligible)

The optimal solution found with simplex method is as follows:

t	z^*_{1t}	z^*_{2t}	z^*_{3t}
1	3	9	5
2	2	10	10
3	1	0	4
4	4	1	5

Now, let's solve it with our algorithm. (One iteration is taken as the number of times Step 1 is performed and only Step 1 on this page is shown in detail)

Step 0: Set the current schedule to the late schedule:

t	z_{1t}	z_{2t}	z_{3t}
1	1	2	4
2	2	8	10
3	3	5	5
4	4	5	5

$t=T=4$

$cv_4=14-10=4$

<u>Iteration 1:</u>

Step 1:

(1.2.1) $i=1$

(1.2.2) $i \neq MP$

(1.2.3) $a_1=\min\{z_{14}, cv_4/b_4\}=\min\{4,4/1 =4\}$

$te_{14}=0, (\Delta Cost)_1=0$

(1.2.4) $k=t-1=3$

(1.2.5.1) $\bar{z}_{13}-z_{13}=0$

(1.2.6) $k=2$

(1.2.5.1) $\bar{z}_{12}-z_{12}=0$

(1.2.6) $k=1$

(1.2.5.1) $\bar{z}_{11}-z_{11}=3-1=2,$

$de_{11}=\min\{a_1, \bar{z}_{11}-z_{11}\}$

$=\min\{4,2\}=2$

(1.2.5.2) $(\Delta Cost)_1=(\Delta Cost)_1+h_1*de_{11}*(t-k)$

$=(0+1*2*3)=6$

(1.2.5.3) $te_{14}=te_{14}+de_{11}=0+2=2$

(1.2.5.4) $(\Delta Capacity)_1=te_{14}*b_1=2*1=2$

(1.2.5.5) $\Delta Cost/\Delta Capacity=6/2=3$

(1.2.5.6) $a_1=4-2=2$

(1.2.6) $k=0$

(1.2.7) $i=2$

(1.2.2) $i\notin MP$

(1.2.3)-(1.2.6) The production of item 2 can be increased in time period t=2 by two units and in time period t=1 by 2 units. Because, $\bar{z}_{23}-z_{23}=0$, $\bar{z}_{22}-z_{22}=2$, $\bar{z}_{21}-z_{21}=2$. (There is no slack capacity in period 3 and slack capacity is 2 in time periods 2 and 1). Then item 2's production can be decreased in the current period (t=4) by 4 units as a result violation will be decreased by 4 units. Thus

$$(\Delta Capacity)_2=4$$

$$(\Delta Cost)_2=2*2*2+2*3*2=20$$

because an inventory of 2 units is carried 2 periods (from 2=2 to t=4) at a unit cost of 2 and another inventory of 2 units is carried 3 periods (from t=1 to t=4) again at a unit cost of 2.

$$(\Delta Cost/\Delta Capacity)_2=20/4=5$$

(1.2.7) i=3

(1.2.2) $i \in MP$

(1.2.3)-(1.2.6) The production of item 3 can be increased in t=1 by 4 units, so $(\Delta Capacity)_3=4$ and $(\Delta Cost)_3=4*3*3=36$, as an inventory of 4 units is carried 3 periods (from t=1 to t=4) at a unit cost of 3.

$$(\Delta Cost/\Delta Capacity)_3=36/4=9$$

(1.2.8) j=1 (item 1 has the minimum ΔCost/ΔCapacity ratio

(1.2.9) As item 1 is the item with the minimum ratio, its schedule is adjusted: Its production is decreased in the current period by 2 units $\Rightarrow z_{14}=z_{14}-2=4-2=2$ (1.2.9.1) and increased in the first period by 2 units $\Rightarrow z_{11}=z_{11}+2=3$ (1.2.9.2-1.2.9.4)

The adjusted schedule is as follows:

t	z_{1t}	z_{2t}	z_{3t}
1	3	2	4
2	2	8	10
3	3	5	5
4	2	5	5

Cost increase=$(\Delta Cost)_1=6$
Total cost increase=6

(1.2.10) $f_1=0$, thus item 1 joins the MP set. (it is the first element of the set)

Step 2:

(2.1) $cv_4=2$

Iteration 2:

Step1:

(1.2.1) i=1

(1.2.2) $1 \in MP$, $(\Delta Cost/\Delta Capacity)_1$=a very large number (actually infinity)

(1.2.7) i=2

(1.2.2) $2 \notin MP$

(1.2.3)-(1.2.6) $a_2=2$ (we want to shift the production of item 2 by 2 units earlier, because this amount is enough to resolve the violation: $cv_4=2$ and $b_2=1$)

=>$(\Delta Capacity)_2=2$

$(\Delta Cost)_2=2*2*2=8$, because an inventory of 2 units is carried 2 periods from t=2 to t=4 at a cost 2 per unit.

$(\Delta Cost/\Delta Capacity)_2=8/2=4$

(1.2.7) i=3

(1.2.8) $3 \notin MP$

(1.2.3)-(1.2.6) $(\Delta Capacity)_3=2$

$(\Delta Cost)_3=2*3*3=18$

$(\Delta Cost/\Delta Capacity)_3=18/2=9$

(1.2.8) $j=2$

(1.2.9) $z_{24}=z_{24}-2=3$

$z_{22}=z_{22}+2=10$

The adjusted schedule is as follows:

t	z_{1t}	z_{2t}	z_{3t}
1	3	2	4
2	2	10	10
3	3	5	5
4	2	3	5

Cost increase=$(\Delta Cost)_2=8$
Total cost increase
in t=4 = 8+6=14
Total cost increase=14

Step 2:

(2.1) $cv_4=0$

(2.2) $t=3$

<u>Iteration 3:</u>

Step 1:

(1.1.1) $cv_3=8$

(1.2.1) $i=1$

(1.2.2) $1 \in MP$, $(\Delta Cost/\Delta Capacity)_1$=a very large number (actually infinity)

(1.2.7) i=2

(1.2.2) $2 \notin MP$

(1.2.3)-(1.2.6) The production of item 2 in the current period(t=3) is 5 units, so we can shift its production by only 5 units earlier=>a_1=5. The sum of the slack capacities in the previous periods (30-2=28) exceeds 5, therefore

$(\Delta Capacity)_2=5$

$(\Delta Cost)_2=5*2*2=20$

$(\Delta Cost/\Delta Capacity)_2=20/5=4$

(1.2.7) i=3

(1.2.2) $3 \notin MP$

(1.2.3)-(1.2.6) $(\Delta Capacity)_3=5$ (The reasoning is the same as that with item 2)

$(\Delta Cost)_3=5*2*3=30$

$(\Delta Cost/\Delta Capacity)_3=30/5=6$

(1.2.8) j=2

(1.2.9) $z_{23}=z_{23}-5=0$

$z_{21}=z_{21}+5=7$

The adjusted schedule is as follows:

t	Z_{1t}	Z_{2t}	Z_{3t}
1	3	7	4
2	2	10	10
3	3	0	5
4	2	3	5

Cost increase=$(\Delta Cost)_2$=20

Total cost increase=14+20=34

Step 2:

(2.1) $cv_3=3$

<u>Iteration 4:</u>

Step 1:

(1.2.1) i=1

(1.2.2) $1 \in MP$, $(\Delta Cost/\Delta Capacity)_1$= a very large number (actually infinity)

(1.2.7) i=2

(1.2.2) $z_{23}=0$ => $(\Delta Cost/\Delta Capacity)_2$= a very large number because its production in the current period is zero and therefore can't be shifted.

(1.2.7) i=3

(1.2.2) $3 \notin MP$

(1.2.3)-(1.2.7) $(\Delta Capacity)_3=cv_3=3$

$(\Delta Cost)_3=3*2*3=18$

$(\Delta Cost/\Delta Capacity)_3=6$

(1.2.8) j=3

(1.2.9) $z_{33}=z_{33}-3=2$

$z_{31}=z_{31}+3=7$

The adjusted schedule (and also the final one as will be seen below) is as follows:

t	z_{1t}	z_{2t}	z_{3t}
1	3	7	7
2	2	10	10
3	3	0	2
4	2	3	5

Cost increase=$(\Delta\text{Cost})_3$=18
Total cost increase
in t=3 =20+18=38
Total cost increase=34+18=52

Step2:

(2.1) $cv_3=0$

(2.2) t=2

Iteration 5:

Step 1:

(1.1.1) $cv_2>0$

Step 2:

(2.2) t=1

Iteration 6:

Step 1:

(1.1.1) $cv_1>0$

The algorithm terminates and the final schedule is the last given adjusted schedule.

Now we can analyze what in the structure makes the algorithm unexact:
In t=4, if we don't shift the production of item 1 earlier and shift the production of item 2 by 4 units (instead of 2 as done in the algorithm) then total holding cost would increase by 2*2*2+2*3*2=20 (as an inventory of 2 units is carried 2 periods and an other inventory of 2 units is carried 3 periods). Our algorithm found the cost increase in period 4 to be 14, so the new decision has a higher cost increase and this difference is 20-14=6 (in t=4). Then in t=3, if 2 units of item 1, 5 units of item 2 and 1 unit of item 3 are shifted earlier to period 1, then total holding cost increase in time period 3 would be 2*1*2+5*2*2+1*3*2=30 (as 2 units item 1, 5 units item 2 and 1 unit item 3 are carried 2 periods, at a cost of 1,2 and 3 respectively)
Recall that our algorithm found the cost increase in t=3 to be 38; then the difference between the cost increases (in t=3) is 38-30=8.
Then the net difference (between the approaches) in the cost is 2, as our algorithm had in t=4 a lower cost increase by 6, but in t=3 a higher cost increase by 8.

That difference arises from the myopic choice in t=4. If in t=4 only the production of item 2 were shifted earlier, instead of shifting 2 units of item 1 and 2 units of item 2, holding cost in t=4 would increase more, but then in t=3 less units of item 3- which is the item with the highest holding cost- were necessary to shift earlier. Thus, the decision taken in time period 4 effected previous time periods.

But, if the preallocated capacities of item 1 in the early periods were not so tight, then the myopic choice had no effect on exactness and the solution would be optimal. Also, if the shared capacity constraint bounds are increased, the effect of the tight preallocated capacities of item 1 would vanish and the solution again would be optimal.

2.6. Test and Evaluation

In order to find out the frequency the algorithm reaches optimal solutions and the percent deviation from optimal solutions, we have set up a test in which 100 randomly generated problems (in which N ranges from 3 to 30 and T ranges from 3 to 12) are solved by using our algorithm and linear programming packages; after which the final solutions have been compared. And the following results are obtained (K-B stands for Karayel-Biton Algorithm and problems for which optimal and K-B solutions are equal are marked):

Problem No	N	T	Optimal Solution	K-B Solution	deviation from optimal solution
1	4	3	717	728.25	0.001569
2*	4	5	1118	1118	-
3*	7	3	1633	1633	-
4*	3	7	1773	1773	-
5*	6	4	677	677	-
6	4	8	1351	1363	$8.88231*10^{-3}$
7*	3	11	5034.5	5034.5	-
8	3	11	5192.5	5484	0.05614
9*	7	5	1577	1577	-
10*	6	6	3993.5	3993.5	-
11*	12	3	2855	2855	-
12	12	3	1947.25	1953	$2.9529*10^{-3}$
13*	3	12	6393.75	6393.75	-
14*	3	12	7878	7878	-
15*	3	12	5348	5348	-
16*	3	12	5466.199	5466.199	-
17	4	9	4516	5363	0.18756
18	8	5	4663.2	4720.6	0.01231
19	10	4	1949.5	1974.75	0.01295
20	4	10	5493.5	5531.5	$6.91727*10^{-3}$
21	10	4	509	509	-
22	5	8	843	859.4	0.01945

Problem No	N	T	Optimal Solution	K-B Solution	deviation from optimal solution
23	6	7	3847.333	4225.5	0.09829
24	6	7	3748.2	4017	0.071714
25	5	9	2924	2952	$9.57592*10^{-3}$
26*	5	9	2924	2924	–
27*	5	9	3531.8	3531.8	–
28	12	4	2522.5	2580	0.0228
29*	8	6	4723	4723	–
30	6	8	6145	6369	0.0364
31	7	7	4243.25	4370.8	0.03006
32	5	10	6378	6534	0.02446
33	10	5	4205	4388	0.04352
34	10	5	1418	1435.2	0.01213
35	13	4	1052.5	1230	0.16865
36*	18	3	3216	3216	–
37	5	11	5989.75	6321	0.05530
38	16	4	2202.3	2231	0.01295
39	13	5	2489	2863	0.15026
40	17	4	2956	3288	0.11231
41*	23	3	3249	3249	–
42	24	3	3402	3402	–
43	9	8	5446	5898	0.0829
44*	6	12	11249	11249	–
45	15	5	5364	5692	0.06115
46*	19	4	3953	3953	–
47	19	4	2764	2951.5	0.0678
48	20	4	8306.5	8582	0.03317
49*	16	5	5755	5755	–
50	10	8	6023	6197	0.0289
51	27	3	7133	7445.5	0.0438
52	11	8	5846	6012.75	0.02852
53*	30	3	4006	4006	–
54*	9	10	5681.4	5681.4	–
55*	9	10	5842	5842	–
56	16	6	8388	8434	$5.48402*10^{-3}$
57	25	4	5578.25	5606.5	$5.0643*10^{-3}$
58	20	5	7426.6	7631	0.027523
59*	12	9	10355	10355	–
60*	27	4	8347	8347	–
61	12	9	11264.75	11849.25	0.05189
62	11	10	10856	11208.25	0.03245
63	11	10	11561.25	11703.45	0.0123
64	11	10	11795.5	11821.5	$2.20423*10^{-3}$
65*	16	7	9347	9347	–
66	17	7	9578	9772.5	0.02031
67	15	8	5774	6157	0.06633

Problem No	N	T	Optimal Solution	K-B Solution	deviation from optimal solution
68	24	5	3497	3782	0.0815
69	11	11	14007	14799	0.0565
70	18	7	12629.75	13182.25	0.04375
71*	12	11	13486	13486	–
72	12	11	28934	30780	0.0638
73*	28	5	10809	10809	–
74	18	8	13992	14648	0.04689
75*	17	9	12517	12517	–
76*	13	12	22370	22370	–
77	27	6	14112	14272	0.01134
78*	24	7	14842	14842	–
79*	17	10	12531.25	12531.25	–
80	20	9	21497	22543	0.04866
81	27	7	16924.25	17031	$6.30752*10^{-3}$
82*	28	7	18737	18737	–
83	18	11	47158	47377.25	$4.7558*10^{-3}$
84	23	10	41139	41899	0.01847
85	21	10	28641	28893.5	$8.816*10^{-3}$
86	24	9	26748	27411.5	0.025
87*	28	8	13846.5	13846.5	–
88	19	12	26429.4	26576.2	$5.2139*10^{-3}$
89	23	10	41139	41899	0.01847
90	29	8	33289	33425	$4.08543*10^{-3}$
91*	30	8	26873.5	26873.5	–
92	30	8	30356	31627.5	0.04189
93	26	10	41843	41941.25	$2.34806*10^{-3}$
94	28	9	37175.25	37177.75	$6.7249*10^{-5}$
95	25	11	32770.6	32936.25	$5.054836*10^{-3}$
96*	23	12	69803	69803	–
97*	27	12	63385.5	63385.5	–
98	27	12	59743	59922	$2.9962*10^{-3}$
99	29	12	48232	48244.5	$2.59164*10^{-4}$
100*	29	12	56345	56345	–

Our algorithm is coded in BASIC and 100 problems are run on Commodore 128 Personal Computer. As linear programming packages LINDO and MPOS were used. Small size problem were solved on Olivetti M24 PC and Commodore PC-20 using linear programming package LINDO. But LINDO has a very limited capacity of 60 constraints. Therefore large size problems are solved on CDC system at Bogazici University using MPOS package.

Our algorithm's response time is considerably shorter than MPOS. For example, a problem having N=28, T=12 had a response time of fourty three seconds when it was solved with our algorithm on Commodore 128, whereas it had a response time of twentyone minutes when it was solved with MPOS package on CDC.

On the other hand, a company facing a problem where N*T>60 can not solve it on its PC and of course systems such as CDC are usually not available in companies. Furthermore, since production planning problems are solved again and again to answer "what if" type of questions, using a mainframe becomes cumbersome.

Even for small size problems ($N*T \leq 60$) the response time is longer using LINDO on PC's than using our algorithm on Commodore 128, although PC's operate with a clock rate of 4.77 MHz and Commodore 128 operates with a clock rate of 2 Mhz.

Also, since memory limitations is a rather crucial issue on microcomputers (users are highly restricted because of memory limitations), the small memory occupation of our program is worth close attention.

Furthermore, 39 out of 100 problems have optimal solutions and the mean deviation from optimal solution is only 1.96 per cent.

III. A NETWORK FLOW APPROACH

In this chapter, it will be shown that the single stage multi-item capacity constrained scheduling problem can be formulated as a minimal cost network flow problem when set-ups are negligible. Also some suggestions are given for the use of such a formulation.

Before starting, recall our problem (P1):

P1 $$\text{Min} \sum_{i=1}^{N} \sum_{t=1}^{T} h_i I_{it} \quad (0)$$

s.t.

$$z_{it} + I_{it-1} - I_{it} = u_{it} \quad (1)$$

$$\sum_{i=1}^{N} b_i z_{it} \leqslant c_t \quad (2)$$

$$z_{it} \leqslant \bar{z}_{it} \quad (3)$$

$$z_{it},\ I_{it} \geqslant 0$$

First, we want to illustrate the minimal cost network flow problem. Then, it will be shown that after performing necessary changes, i.e. variable transformation and adapting the shared capacity constraints to network format, P1 can be formulated as a minimal cost network flow problem and the graph of the network will be illustrated.

3.1 Minimum-Cost Network Flow Problem (MCNFP)

Consider the problem of sending a specified amount of flow value θ from source s to sink d in a network G in which every edge (arc) (k,l) has a capacity c_{kl} as well as a nonnegative cost d_{kl} associated with it.

Our purpose is to find the flow pattern which minimizes the total cost. This problem is called the minimal cost network flow problem and may be stated more formally as follows:

$$\text{Min} \sum_{(k,l)\in G} \sum d_{kl} f_{kl} \qquad (0')$$

s.t.

$$\sum_k f_{mk} - \sum_k f_{km} = \theta \qquad (1')$$

$$\sum_k f_{dk} - \sum_k f_{kd} = -\theta \qquad (2')$$

$$\sum_k f_{lk} - \sum_k f_{kl} = 0 \qquad (3')$$

$$f_{kl} \leqslant c_{kl} \quad \text{for every } (k,l)\in G \qquad (4')$$

$$f_{kl} \geqslant 0 \quad \text{for every } (k,l)\in G \qquad (5')$$

3.2. Variable Transformation

Recall that in Chapter 2.2. transformation of variables has been presented. After the transformation the problem P2 was obtained equaling to P1.

P2 was stated as:

P2 $$\text{Min} \sum_{i=1}^{N} \sum_{t=1}^{T} h'_i I'_{it} \quad (0'')$$

s.t.

$$z'_{it}+I'_{it-1}-I'_{it}= u'_{it} \quad (1'')$$

$$\sum_{i=1}^{N} z'_{it} \leqslant c_t \quad (2'')$$

$$z'_{it} \leqslant \bar{z}'_{it} \quad (3'')$$

$$z'_{it},\ I'_{it} \geqslant 0 \quad (4'')$$

3.3 Adaptation of Shared Capacity Constraints

On figure 3.1. the network representation of (P2) is given. On each arc (i,j) the flow on it, the lower bounds of flow, upper bounds of flow and the cost of sending one unit of flow are given respectively. When analyzing carefully, it can be seen that it is the network for the minimal cost network flow problem. Node s is the source node, which has a supply of the sum of all external demands i.e.

$$\sum_{i=1}^{N} \sum_{t=1}^{T} u'_{it}$$

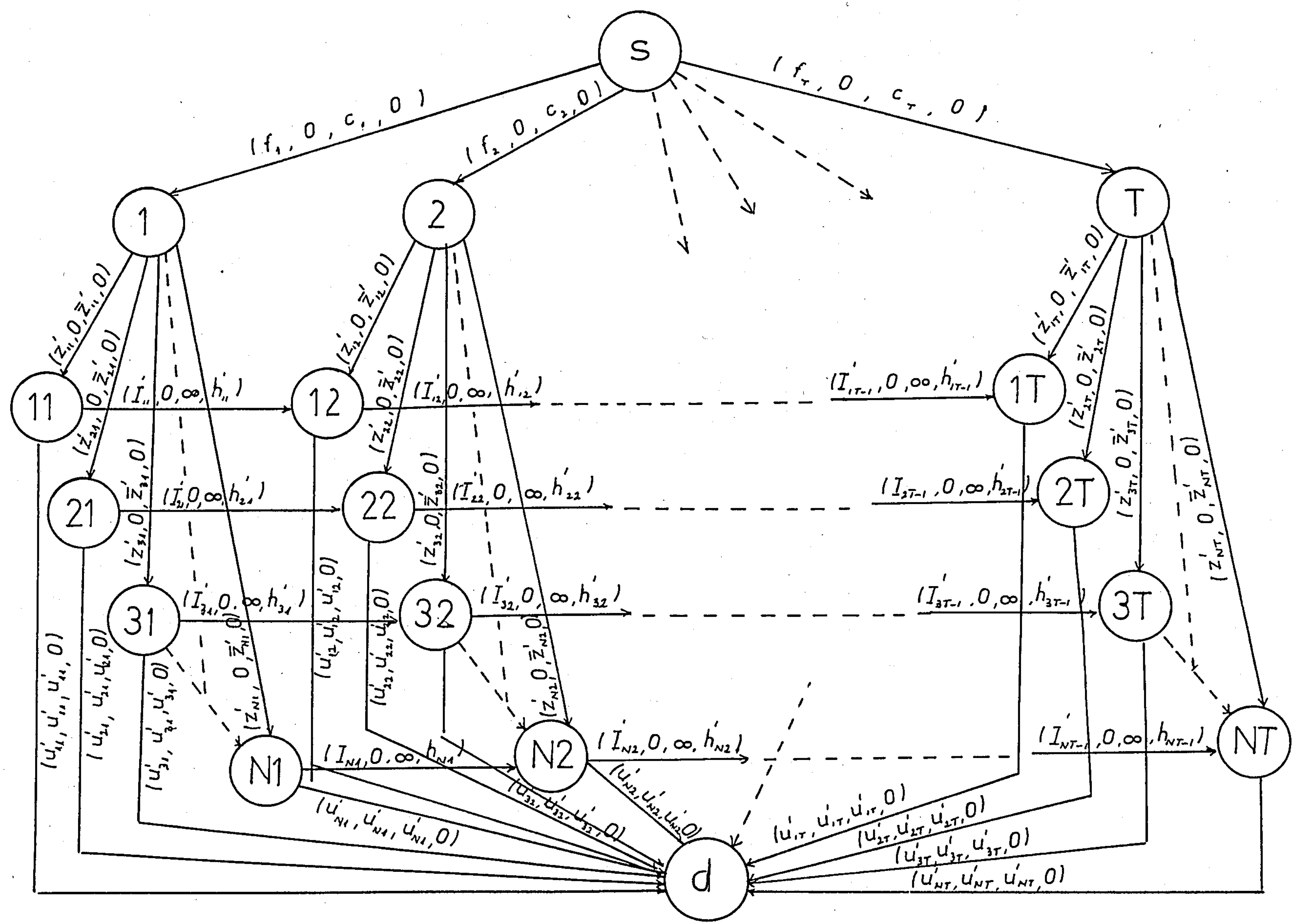
s
1
2
T
11
12
1T
21
22
2T
31
32
3T
N1
N2
NT
d
(f₁, 0, c₁, 0)
(f₂, 0, c₂, 0)
(f_T, 0, c_T, 0)
(I'₁₁, 0, ∞, h'₁₁)
(I'₁₂, 0, ∞, h'₁₂)
(I'₁T-1, 0, ∞, h'₁T-1)
(I'₂₁, 0, ∞, h'₂₁)
(I'₂₂, 0, ∞, h'₂₂)
(I'₂T-1, 0, ∞, h'₂T-1)
(I'₃₁, 0, ∞, h'₃₁)
(I'₃₂, 0, ∞, h'₃₂)
(I'₃T-1, 0, ∞, h'₃T-1)
(I'N1, 0, ∞, h'N1)
(I'N2, 0, ∞, h'N2)
(I'NT-1, 0, ∞, h'NT-1)
(u'₁T, u'₁T, u'₁T, 0)
(u'₂T, u'₂T, u'₂T, 0)
(u'₃T, u'₃T, u'₃T, 0)
(u'NT, u'NT, u'NT, 0)

Figure 3.1

Node d is the sink node which has a demand equaling to the above sum. Arcs between nodes s and t=1,2,.....,T carry the total production of periods t=1,2,.....,T i.e. they carry

$\sum_{i=1}^{N} z'_{it}$ for every t.

Arcs between nodes t (t=1,2,.....,T) and it (for i=1,2,......N; for t=1,2,.....,T) carry the production of each individual item in t, z'_{it}. Arcs between nodes it and it+1 have inventory flows, I'_{it}.

Finally arcs between nodes it and d carry the demand of items i=1,2,....,T, u'_{it}.

With the presence of the nodes t=1,2,....,T and arcs leaving source node s and entering into these nodes, we meanwhile adapted the shared capacity constraints to network format, because the flow entering any node t is the total production in period t, $\sum_{i=1}^{N} z'_{it}$. Then the arc connecting source s and t must have the capacity of c_t, because the flow value on arc (s,t) is $\sum_{i=1}^{N} z'_{it}$ which has a capacity of c_t.

3.4. Equivalence of the two approaches

In the last section, it was illustrated that the network represantations of both P2 and MCNFP are equal. Now, it will be shown that P2 is a MCNFP by equating MCNFP's constraints to constraints of P2.

3.4.1. Equivalence of the objective functions

In this part we will show that the objective function of P2(O") is equivalent to the objective function of MCNFP(O'). That is,

$$\sum_{(k,l)\in G} \sum d_{kl} f_{kl} \text{ is equivalent to } \sum_{i=t}^{N} \sum_{t=1}^{T} h'_i I'_{it}.$$

d_{kl} is the cost associated with flow f_{kl} on arc (k,l) where h_{it} is the cost associated with item i's inventory carried from period t to period t+1. On figure 3.1. h_i's are the costs associated with the arcs connecting nodes it and it+1 for every i=1,2,....,N; t=1,2,....T. Flows on arcs connecting nodes t and it have zero costs, as only the inventory holding costs are considered (production costs are not included in the objective function).

Then, for every $(k,l)\in G$; i=1,2,....,N and t=1,2,....,T on Figure 3.1.

$d_{kl}=h_i$ for every arc (k,l)=(it,it+1)

$d_{kl}=0$ for every arc (k,l)=(t,it)

$d_{kl}=0$ for every arc (k,l)=(it,d)

3.4.2. Equivalence of flow balances

Constraint set (3') in MCNFP is equivalent to constraint set (1") in P2, because both are flow balance equations for every intermediary node (a node which is neither source nor sink node).

$\sum_k f_{ik}$ is the sum of the flows outgoing from an intermediary node k.
It corresponds on Figure 3.1. to the sum of the following flows: For each node it (i=1,2,...,N; t=1,2,...,T) the outgoing arcs are (it,it+1) and (it,d) which carry flows I'_{it} and u'_{it} respectively. Then a total flow of $I'_{it}+u'_{it}$ leaves that node.

Thus $\sum_k f_{ik}$ is equivalent to $(I'_{it}+u'_{it})$
(As figure 3.1 is also the network of MCNFP)

$\sum_k f_{ki}$ is the sum of the flows entering an intermediary node k.
It corresponds on Figure 3.1. to the sum of the following flows. For each node it, the incoming arcs are (it-1, it) and (t,it) which carry flows I'_{it-1} and z'_{it} respectively. Then a total flow of $I'_{it-1}+z'_{it}$ enters that node.

Thus $\sum_k f_{ki}$ is equivalent to $(I'_{it-1}+z'_{it})$

As a result $\sum_k f_{ik}-\sum_k f_{ki}=0$ (3')
is equivalent to $z'_{it}+I'_{it-1}-I'_{it}-u'_{it}=0$ which is constraint (1") in P2.

3.4.3. Equivalence of Capacities

In this section we illustrate how constraint sets (2") and (3") in P2 are equivalent to constraint set (4') in MCNFP.

(4') stands for the capacity of each individual flow. Our problem has capacity restrictions only for production (thus on figure 3.1. only arcs (s,t) and (t,it) have capacities, as they represent production flows). No restriction is brought for inventories (then arcs (it, it+1) have zero capacities as they represent inventory flows).

Then for every $(k,l) \in G$, c_{kl} (in MCNFP) corresponds to the following capacities on Figure 3.1. (again for i=1,2,...,N; t=1,2,...,T).

<u>Preallocated capacities</u>

If arc (k,l)=(t,it) then $c_{kl} = \bar{z}'_{it}$, because the amount of production of item i in period t is limited by $\bar{z}'_{it}$. Then in Figure 3.1. $\bar{z}'_{it}$ is the capacity of each arc going out from node t and entering it, (t,it), due to the fact that the flows on arcs leaving node t and entering it carry the production of item i in period t, z'_{it} (the capacity of z'_{it} is $\bar{z}'_{it}$).

If arc (k,l)=(it,it+1) then $c_{kl}=0$, because arcs (it,it+1)'s have inventory flows, which are without capacity. They are restricted only to be nonnegative (4" in P2).

<u>Shared Capacity Constraints</u>

If arc $(k,l)=(s,t)$ then $c_{kl}=c_t$.

The sum of the flows entering node t must be equal to the sum of the flows leaving node t. Only the arc between source s and t enters t, which has a flow value of f_t.

The sum of the flows on arcs leaving node t, is the total production in period t, $\sum_{i=1}^{N} z'_{it}$, as each arc (t,it) carries a flow of z'_{it} (and there are N arcs leaving t).

Then, $f_t=\sum_{i=1}^{N} z'_{it}$

As the total production in t, $\sum_{i=1}^{N} z'_{it}$, is restricted by the shared capacity constraint c_t, the arcs leaving sink node s, (s,t)'s, have capacities of c_t's.

(Recall that the same argument was given in Section 3.3.)

To sum up,

$$c_{kl}=\begin{cases} \bar{z}'_{kl}, & \text{if } (k,l)\in G=(t,it) \text{ on Figure (3.1).} \\ c_t, & \text{if } (k,l)\in G=(s,t) \text{ on Figure (3.1).} \\ 0, & \text{if } (k,l)\in G=(it,it+1) \text{ on Figure (3.1).} \end{cases}$$

If,

$c_{kl}=\bar{z}'_{it}$ then (4') in MCNFP is equivalent to (2") in (P2)

$c_{kl}=c_t$, then (4') in MCNFP is equivalent to (3") in (P2)

$c_{kl}=0$, then (4') in MCNFP is equivalent to (5") in (P2)

3.4.5. Nonnegativity of flows

It is very obvious that constraint set (5') in MCNFP is equivalent to set (4") indicating the nonnegativity of flows.

3.4.6. Total flow value and source-sink nodes

Constraint set (1') in MCNFP states that the supply of the source node is θ, or the difference between the sum of the flows leaving source node and entering source node is θ. As no flow enters source node, the sum of the flows entering it is zero.

On Figure 3.1. the sum of the flows leaving s is $\sum_{t=1}^{T} f_t$.

As $f_t = \sum_{i=1}^{N} z'_{it}$, then the sum of the flows leaving s is $\sum_{i=1}^{N} \sum_{t=1}^{T} z'_{it}$

Also, (2') states that the sum of the flows entering the sink node (as no flow leaves it) is zero.

On Figure 3.1. the sum of the flows entering sink node, d, is the sum of the external demands, $\sum_{i=1}^{N} \sum_{t=1}^{T} u'_{it}$.

(1') in MCNFP states that $\sum_{i=1}^{N} \sum_{t=1}^{T} z'_{it} = \theta$

(2') in MCNFP states that $\sum_{i=1}^{N} \sum_{t=1}^{T} z'_{it} = -\theta$

Then, $\sum_{i=1}^{N} \sum_{t=1}^{T} z'_{it} = \sum_{i=1}^{N} \sum_{t=1}^{T} u'_{it}$

The same result will be obtained by summing each flow balance equation in P2, namely 1"'s (as 1" in P2 represents N*T flow balances).

3.5. A Suggestion

In section 3.3. and 3.4. it was shown that our scheduling problem can be formulated as a minimal cost network flow problem (as its objective function and constraints can be transformed to those of a minimal cost network flow problem).

Due to the fact that linear programming methods are slow, the following suggestion is made to find optimal solutions.

As our algorithm provides near-optimal solutions, it is advisable to solve the problem with our algorithm and then transform the problem to MCNFP and solve it by using any MCNFP algorithm in which the solution of our algorithm is taken as an initial solution. Thus, in only a few iterations the optimal solutions are obtained.

IV. CONCLUSION

In the preceding chapters the single stage multi-item capacity constrained problem was analyzed and a heuristic algorithm for its efficient solutions was introduced.

The algorithm provides a very important advantage, by settling the tradeoff between exactness and speed favourably.

One can think of many applications for the scheduling algorithm. It can also be implemented in the multi-stage case. Another application may be using it as an aid in making the decisions of using overtime in a period or not.

The algorithm can be used to generate good initial solutions for the minimal cost network flow problem. The major advantages of our algorithm is its simplicity and implementability on a microcomputer. Large problems can be solved in a short amount of time on a microcomputer. Common problems in industry exhibit the property that for a given item the preallocated capacity usage, the shared capacity demand and the holding costs are closely related in which case our algorithm finds very good (usually optimal) solutions.

BIBLIOGRAPHY

1. Wagner, H.M., & Whitin, T., 'Dynamic version of the economic lot-size model,' Management Science, 5, No. 1,89-96, 1958.

2. Florian, M., & Klein, M., 'Deterministic production planning with concave costs and capacity constraints,' Management Science, 18, No. 1, 12-20, 1971.

3. Lambrecht, M., & Van der Eecken J.,' A capacity constrained single-facility dynamic lot-size model,' European Journal of Operational Research, 2, 132-136, 1978.

4. Bitran, G.R., & Yanesse, H.H., 'Computational complexity of the capacitated lot-size problem,' Management Science, 28, No. 10, 1174-1186, 1982.

5. Karayel M., 'Dual-Based Heuristics For Capacity-Constrained Production Scheduling,' Ph.D. Dissertation, Graduate Division of The University of California, Berkeley, 1984.

6. Manne, A.S., 'Programming of economic lot sizes,' Management Science, 4, No.2, 115-135, 1958.

7. Lasdon, L.S., & Terjung, R.C., 'An efficient algorithm for multi-item scheduling,' Operations Research, 19, No.4, 946-969, 1971.

8. Newson, E.F.P., 'Multi-item lot-size scheduling by heuristic. Part 1: With fixed resources,' Management Science, 21, No. 10, 1194-1203, 1975.

9. Graves, S.C., 'A review of production scheduling,' Operations Research, 29, No.4,646-675, 1981.

10. Bitran, G.R., Haas, E.A., & Hax, A.C., 'Hierarchical production planning: A single-stage system,' Operations Research, 29, No. 4,717-743, 1981.

11. Love, S., 'A facilities in series inventory model with nested schedules,' Management Science, 18, No.5, 327-338, 1972.

12. McClain, J.O., Thomas, L.J., & Weiss, E.N., 'Solving capacity constrained MRP problems by decomposition,' Working paper, 81-18, Graduate School of Business and Public Administration, Cornell University, 1981.

13. Billington, P., McLain, J., &Thomas, J., 'Mathematical programming approaches to capacity constrained MRP systems: Review, formulation, and problem reduction,' Management Science,29, No. 10, 1126-1141, 1983.

14. Gabbay,H., 'Multi-stage production planning,' Management Science, 25, No.11, 1138-1148, 1979.

15. Bitran, G.R., Haas, E.A., & Hax, A.C., 'Hierarchical production planning: A two-stage system,' Operations Research, 30, No. 2 ,232-251, 1982.