Follow up (10/13)

Machine Assignment

Industry Control:

a machine shop is required to manufacture j jobs on m alternate machines. There may be a different time t_{mj} for doing each job on each machine.

$t_{m,j}$ can be precomputed for each machine as a function of job tool regts, initial machine states, running time, and job setup time.

If we let $x_{m,j}$ = 0 if job j is not to be done on machine m

x_{mj} = 1 if job j is to be done on machine m

$j = 50 \cdot 10^0$
$m = 10$

Then

$$\sum_{j=1}^{JMAX} (x_{m,j})(t_{mj}) \leq TMAX_m \quad \text{for } m = 1, 2, \ldots, MMAX$$

$$\sum_{m=1}^{MMAX} x_{m,j} = 1 \qquad j = 1, 2, \ldots, JMAX$$

Minimize $\sum_{j,m} x_{mj} t_{mj}$

Objective function

$$\text{Minimize} \quad T = \sum_{m=1}^{m=M} T_m$$

all jobs must be assigned
treat all jobs as unit qtys

— this solution does not permit assignment to more than one machine

For example it would be valid if job setup >> tool setup.

First step of computation is to construct time / matrix

Given time matrix - I believe we have a linear programing problem

$T_{j,m}$ = tool setup cost + running time + job setup cost
no of tools missing × unit cost / tool setup
+ qty of units × running time/unit
+ job setup cost

Assume Job setup cost is equivalent on all mach
running time per unit is " " "

(still need in equations to satisfy T_{MAX_m} restraint.

Machine Assignment Problem

if $C_j = 1.5\ C_t$

		C_j	C_t
V → A (1, 3, 4)		1	
Y → C (1, 7, 8)	Setup 1 remove 6	1	1
W → B (2, 5, 6)	Setup 2 Remove 3	1	1
X → C (5, 9, 10)	Setup 10 Remove 7	1	1
Z → A (3, 4, 5)		1	
		5	3

Theoretical Minimum = $1.5 + 1 = 8.5$ $\quad 1.5 + 3 = 10.5$

Given a priori usage probabilities

1	2	3	4	5	6	7	8	9	10
.3	.6	.6	.4	.2	.2	.1	.2	.2	.2

to really determine assignment must consider loading -
if time fixed per stroke + table movement -
then time is proportional to number of holes
plus setup time -

~~Setup~~ Sequencing is also significant.

The General Problem

5 jobs ; 3 tools req'd per job
3 machines
5 tools per mach
10 tools in general

Tools/MACH

	A	B	C	TR
1	1			2
2	1			1
3	1	1		1
4	1	1		1
5	1	1	1	
6		1	1	
7		1	1	1
8			1	1
9			1	2
10				1

TOOLS/Jobs

	V	W	X	Y	Z
1	1			1	
2		1			
3	1				1
4	1				1
5		1	1		1
6		1			
7				1	
8				1	
9			1		
10			1		

MACH/JOBS A B C

V	1,3,4	3,4	-
W	2,5	5,6	5,6
X	5,	5,	5,9 10
Y	1	7	7,8
Z	3,4,5	3,4,5	5,

C_j = Job setup cost

C_t = Tool setup cost

(1) Given restrictions:
choose 1 machine for minimum setup
all setups cost the same
all setups equally likely to be used

Then problem is straightforward -
choose randomly among machines which each satisfy maximum number of job requirements.

(2) If wish to weight choice based on likelihood of setup use
then have a probability function for each tool -
$t_{.15}$, .2 means that 20% of the jobs coming in require use of $t_{.15}$. The significance of this percentage may be affected by number of machines - will it really be re-setup or not.

All this assumes [non-interdependence] independent tool selection

Problem Analysis

System

- m identical machines
- the immediate capability of each is described by $50 t_i$
- $i_{max} < 1000$
- t_i on machine m may be identical to t_i on machine m'
- Input for Order O_j will describe

$$\left\{ \begin{array}{l} Q_i, t_i, L_{i,w}, D \\ Q_{i'}, t_{i'}, L_{i,w}, D \\ \vdots \end{array} \right\}$$

- Decision:
  Combination of machines required to perform all tasks
- Decision criteria

Minimize

$$Cost_{O_j} = \sum S_m + \sum S_j$$

where S_m = machine setup required to provide capability on machine m to perform task t_i.

S_j = setup for job j to begin operation on machine m

Simplest Case

$m = 2$

$j = 2$

$i = 4$

$m_1 \subset t_1, t_2$;

$m_2 \subset t_2, t_3$

$J_1 \supset t_1, t_3$; $R_{MAX_1} = 2$ quantities are not significant

$J_2 \supset t_2, t_4$; $R_{MAX_2} = 2$

$T_N \equiv N^{th}$ Task

$TMAX_j$ = Total tasks for job j

for J_1, $T_1 = t_1$; $T_2 = t_3$ or $T_{1,1} = t_1$; $T_{2,1} = t_3$

J_2, $T_1 = t_2$; $T_2 = t_4$ or $T_{1,2} = t_2$; $T_{2,2} = t_4$

$T_{r,j}$ = r^{th} task on the j^{th} job

6 - Wiedemann
15 tools/Wiedemann

—— jobs/day

Typical

Job setup
tool change
unit running time

} independent of machine + job

Logic –

1. If Jt. requirements matched by a machine description assign Jt to machine

2. If Jt Req'ts matched by a machine except for m out of all m t_i are not avail anymore. Then assign Jt to machine and set up m t_i

3. Is Rest an Inventory problem – multiple availability of tools

4. If Jt Req'ts are matched by a combination of two machines and there are no other "available" tools to setup either machine 100%

5. If Jt Req'ts are matched by a combination of two, or more machines then compare extra job setup with setup of needed tools on each machine – choose minimum alternative –

6. If Jt Req'ts are matched by, or more than one machine then choose randomly among equal alternatives

7. Alternative Combinations – might be settled by a load preference rule – Cannot Optimize –

Assumptions:

Initial state predetermined

no. of tools per job << no of tools per machine

relatively few job/day/machine

2 machine $m = 1, 2$

$$t_{1j} = t_{2j}$$

$$\min T = \frac{\sum a_{mij}}{2}$$

Jewish rules.

0,1

fractional values

$$a_{mj} \cdot a_{m'j'} \geq a_{mj'} \cdot a_{m'j}$$

$$\sum_j x_{mj} \cdot a_{mj} \leq T$$

$$\sum_m x_{mj} \leq 1$$

allow fractional values —

restriction —

for any jth There is a test machine

machine assignment - (1)

j jobs, m machines

each mach can do job j but at a different cost (implied by setup + running time)

- if assume that for each machine for each day a change tool pattern can be evolved which will avoid ~~doing~~ putting back on ~~mach~~ tools which were removed and if assume that there will be no setup reduction because of job sequence pattern

In other words establish set of jobs for each machine for each day - detailed sequence to be determined later

j ↓ | m → | $C_{j,m}$

~~$C_{j,m}$ = Cost of~~

$T_{j,m}$ = Time for doing job ~~mach~~ m of - includes job j on job setup, tool setup

$$T_m = \sum_{j=1}^{J} T_{j,m} \leq TMAX_m \; ; \; m = 1, 2, \ldots, M$$

$T_{j,m}$ is a function of initial state of mach m $\{t_i\}_{m=1,2,\ldots,M}$

and req'ts of job j $\{t_i\}_{j=1,2,\ldots,J}$

In general,

$$T_{j,m} \propto \left[\{t_{i,j}\} - \{t_{i,m}\} \right] \quad \text{for } j \neq m$$

Production Control Rules

(1) Promise Date

(1a) Finished Goods - balance stk, non-stk

(2) Material Ordering - to order
stock

--- Raw, procured parts,
procured assemblies
maintenance & Supply parts

(3) Parts manufacturing ~~or~~ scheduling (or Releasing)
Assembly
- to order
- stock

(4) Factory Dispatching (sequencing)

(5) Labor Allocation

(6) Operational level (prod. rate - layoff or rehire)

ancillary or support operations

(1) demand explosion

(2) demand prediction

(3) Inventory record keeping
receipts
withdrawals
orders on vendors
returns

(4) shipping records

(5) operator instructions, vouchers

An Analytical study of the Pure Assembly Line Model

Definition -

(1) all jobs are processed sequentially through all stations.

(2) Initial job sequence is maintained throughout the line

(3) must be for predetermined batch size (1 up) - nonejectable

(4) deterministic setup, running time - may be dependent on initial ordering.

(5) any number of jobs and stations

Problem - Select an optimum (Best achievable) sequence.

Approach -

While there does not appear to be any available technique for ~~optim~~ obtaining an optimum solution to the above problem (short of exhaustive enumeration), it does seem fruitful to consider whether a theoretical minimum time (TMT) can be derived. It would also be useful to be able to analytically evaluate the ~~processing~~ total time which would be taken by any given initial ordering.

The examples included in this paper are taken from "Simulating Factory Operation on an IBM 407 Tabulator" by John E. McConnell of Advanced Equipment development.

Theoretical Minimum Time Calculation

assumptions:

(1) There are 7 jobs. : 1, 2, ..., 7
4 stations: 1, 2, 3, 4

(2) a setup time (sequence independent) is established for each job on each station

(3) a unit processing time is specified for each job on each station

(4) a job lot size is detailed for each job.

(5) Transportation between stations is instantaneous

(6) A unit of a job may move to ~~the~~ The next station as soon as it completes work at a previous station; i.e. 100% Lap Phasing

(7) Setup may be performed prior to the arrival of the first piece of a given ~~job~~ job.

(8) The shop starts with all stations empty.

analysis:

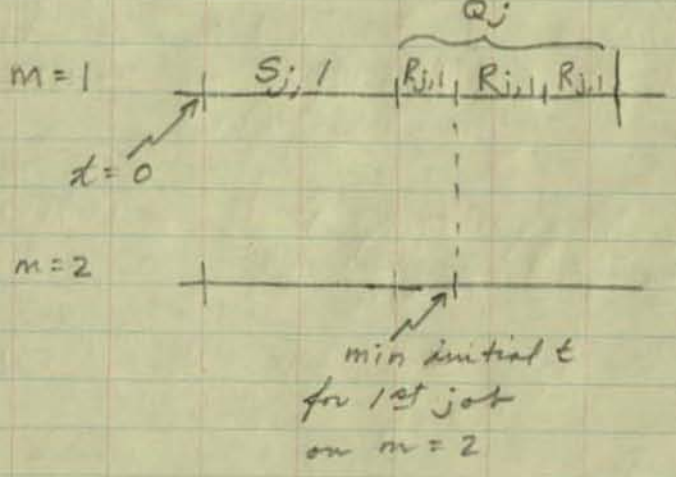
j = job number

m = machine number

$S_{j,m}$ = Setup time of job j on machine m

$R_{j,m}$ = Unit processing (running) time of job j on machine m

Q_j = Lot Quantity of job j



Job j will start at machine 2 at

$$t = (S_{j,1} + R_{j,1}) \quad \text{if} \quad S_{j,2} \leq (S_{j,1} + R_{j,1})$$

~~if we define $S_{j,m}$ = start time of job j on machine m~~

~~if~~ we define an ordering $k = 1, 2, \ldots, 25$ where each unit of each job j is assigned a unique ordering value k

Questions -
(1) notation for ordering
(4) inclusion-exclusion
3 setup time

Then

$S_{k,m}$ = start time of k^{th} unit (in order) on m^{th} machine.

and $t_{k,m}$ = completion time of the k^{th} unit on the m^{th} machine

It can be stated that:

For that j for which $k = 1, 2, \ldots, Q_j$

$S_{1,1} = 0$

$t_{1,1} = S_{j,1} + R_{j,1}$

$S_{1,2} = \max\{t_{1,1}, S_{j,2}\}$

$t_{1,2} = S_{1,2} + R_{j,2}$

$S_{2,1} = t_{1,1}$

$t_{2,1} = S_{2,1} + R_{j,1,1}$

if $Q_j \geq 2$

$t_{2,1} = S_{2,1} + S_{j,1} + R_{j,1}$

if $Q_j < 2$

$S_{2,2} = \max\{t_{2,1}, t_{1,2}\}$

$$t_{2,2} = s_{2,2} + R_{j,2} \quad \text{if } Q_j \geq 2$$

or

$$t_{2,2} = s_{2,2} + S_{j,2} + R_{j,2} \quad \text{if } Q < 2$$

$$s_{1,3} = \max\{t_{1,2}, S_{j,3}\}$$

$$t_{1,3} = s_{1,3} + R_{j,3}$$

$$s_{2,3} = \max\{t_{2,2}, t_{1,3}\}$$

$$t_{2,3} = s_{2,2} + R_{j,3} \quad \text{if } Q_j \geq 2$$

or

$$t_{2,3} = s_{22} + S_{j,3} + R_{j,3} \quad \text{if } Q_j < 2$$

$$s_{3,3} = \max\{t_{3,2}, t_{2,3}\}$$

$$t_{3,3} = s_{3,3} + R_{j,3} \quad \text{if } Q_j \neq 2$$

or

$$t_{3,3} = s_{3,3} + S_{j,3} + R_{j,3} \quad \text{if } Q_j = 2$$

To recapitulate:

$$S_{k,m} = \max \{ t_{k-1,m} , t_{k,m-1} \}$$

$$t_{k,m} = S_{k,m} + R_{j*,m} \quad \text{if} \quad \sum_{j=1}^{j*} Q_j \neq (k-1)$$

or

$$t_{k,m} = S_{k,m} + R_{j*,m} + S_{j*,m}$$

Theoretical Minimum Time (TMT) can be computed as follows.

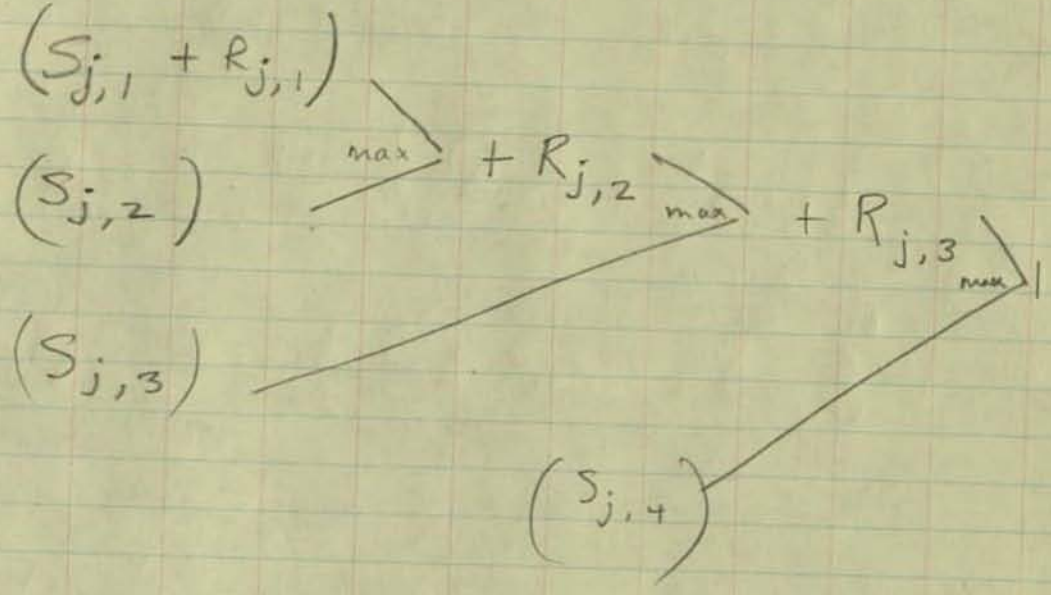
(1) Select machine m ~~with fact~~ to maximize

$$T_m = \sum_{all\ j} \left(S_{j,m} + Q_j R_{j,m} \right); \quad m = 1, 2, 3, 4$$

call this machine m* and the time required T_{m*}

(2) ~~select~~ ~~determine~~ Select job j to minimize

$$S_{1,m*} = \max \{$$



$ENT = [\quad]$

if $\left[(S_{j,1} + R_{j,1}) - S_{j,2}\right] > 0$ Enter entity time

$[\quad] \leq 0$

$ENT = 0$ if $[\quad] \leq 0$

FOR ANY FLOW SHOP MODEL WITH

all jobs thru all stations
First Come First Served Station Assignment
Either 100% batch or unit advance
(maybe for anyplace in between)
deterministic running times
predetermined, non rejectable quantities
(maybe with predictable rejects)
deterministic setup - dependent only on
sequence of arrival

I now "almost" have an algorithm which will compute exact completion time of each "batch" or unit at every station; determine queue delays; compute starve delays

def: Simulator

- Don't McCombie's simply an analytical model - not a simulator -

Same with MSS Simulator
Though in attempt to develop intermediate values and to make it look like a simulator there may be a significant speed sacrifice.

Theoretical Minimum time - three or four machines -

(using McCombie data) -

Complete lap phasing
instantaneous transportation

Mod i takes $S_{i,1}$ setup on station 1

$N_{i,1}$ running time on stn 1 per unit

$S_{i,2}$ setup on stn 2

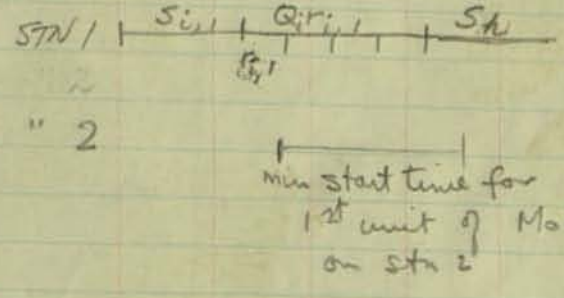
$N_{i,2}$ running time on stn 2 per unit

$S_{i,j}$ setup on stn j

$N_{i,j}$ running " " per unit

$[k \neq i]$

Q_i = quantity in lot of mod i



if $[S_{i,2} \leq (S_{i,1} + r_{i,1})]$ then stn 2 starts at $(S_{i,1} + r_{i,1})$

if not then it start at S_i

$T_{u,j}$ = completion time of unit u at stn j

for $u = 1$

$$T(1,2) = \text{Max}\left(S(i,2), \left(S(i,1) + r(i,1)\right)\right) + r(i,2)$$

for $u = 2$ where $\text{mod}(u=1) = \text{mod}(u=2)$

$$T(2,2) =$$

$$\text{MAX}\left[\left(S_{i,2} + r_{i,2}\right), \left(S_{i,1} + 2r_{i,1}\right)\right] + r_{i,2}$$

for $u = n$ where $\text{mod}(u=n) = \text{mod}(u=n-\ell)_{\ell=1}^{\ell=n-1}$

$$T(n,2) = \text{MAX}\left[\left(S_{i,2} + (n-1)r_{i,2}\right), \left(S_{i,1} + n r_{i,1}\right)\right] + r_{i,2}$$

For the third station

$$T(1,3) = \text{MAX}\left[T(1,2), S_{i,3}\right] + r_{i,3}$$

Recapitulate

$$T_{1,1} = S_{i,1} + r_{i,1}$$

$$T_{1,2} = MAX\left[(S_{i,2}),(T_{1,1})\right] + r_{i,2}$$

$$T_{1,3} = MAX\left[(S_{i,3}),(T_{1,2})\right] + r_{i,3}$$

Hence $T_{1,j} = MAX\left[(S_{i,j}),(T_{1,j-1})\right] + r_{i,j}$

If there were no starve delays on machine j

then $T_{m,j} = T_{1,j} + \sum_{i_1}(S_{i,j} + Q_i r_{i,j}) - r_{1,j}$

for m units of i model.

→ Search for that model which will make $T_{1,j}$ a minimum for that jth machine which has a setup + running time summation such that $T_{i,j} + \Sigma$ would be a maximum

Algorithm -

for $j = 1, 2, \ldots,$ JMAX

calculate $T_{1,j}^{i}$ for $i = 1, 2, \ldots,$ IMAX

	j=1	j=2	j=3	j=4
$T_{1,j}^{1}$	46	58	62	75
$T_{1,j}^{2}$	48	60	66	79
$T_{1,j}^{3}$	46	58	64	74
$T_{1,j}^{4}$	13	17	21	29
$T_{1,j}^{5}$	51	69	73	86
$T_{1,j}^{7}$	13	40	44	57
$T_{1,j}^{11}$	13	82	88	114

compute $\sum_{i=1,I_{max}} (S_{i,j} + Q_i r_{i,j})$ for each j

i	Q_i	$j=$ 1	2	3	4
1	3	78	69	18	59
2	4	102	69	44	72
3	2	62	45	32	40
4	4	37	27	22	47
5	4	114	89	22	72
7	4	37	109	22	72
11	4	37	229	44	132
$\sum_j$		467	637	204	494

"Minimum Theoretical minimum"

$$\text{Min } T_{1,j} + \max \sum_j - N_{1,j} + \text{remaining operation allowance}$$

for $j=2$ $\quad 17 + 637 + ROA - 4$

May want to trade off $T_{1,j}$ against ROA

for $j=2$, $\quad ROA = \min(r_{i,3} + r_{i,4})$

in general

$ROA_{i,j}$

i \ j =	1	2	3	4
1	29	17	13	0
2	31	19	13	0
3	28	16	10	0
4	16	12	8	0
5	35	17	13	0
7	40	17	13	0
11	81	32	26	0

$\min(ROA_{i,2}) = ROA_{4,2} = 12$ but this provides lowest $T_{1,2}$ hence

see next lowest ROA which is $ROA_{3,2} = 16$

hence

$MTM = 17 + 637 + 16 - 4$ (6.66)

for 3 machine case

$\min(ROA_{i,2}) = 4 \quad$ for $i = 1, 5, 7$

hence $MTM = 17 + 637 + 4 - 4 = \underline{658}$

Theoretical
For minimum time

sequence must begin with 4

and end with 1, 5 or 7

To test for starve delays we must analyze as follows:

for lot size Q

for first lot, $u = Q_i$

$$T_{u,i} = T_{i,1} + (Q_i - 1)(r_{i,1})$$

$$T_{u,2} = MAX\left[(T_{u,1} + r_{i,2}), (T_{i,2} + (Q_i - 1)(r_{i,2}))\right]$$

$$T_{u,j} = MAX\left[(T_{u,j-1} + r_{i,j-1}), (T_{i,j} + (Q_i - 1)(r_{i,j}))\right]$$

$T_{u,j}$

i	j = 1	2	3	4
1	78	90/82	94/70	107/101
2	102	114/96	120/84	133/118
3	62	74/70	80/70	90/84
4	37	41/29	45/33	53/53
5	114	132/123	136/85	149/125
7	37	60/109	113/56	126/96
11	37	86/229	235/106	261/192

This needs to be analyzed in view of minimum productive time given a sequence that should enable us to compute completion time

whenever $[T_{u,1} + r_{i,2}] > [T_{i,2} + (Q_i - 1)(r_{i,2})]$

there is a "starve" delay for the machine if the difference for the reverse. There is an Idle time delay for part.

Example -

q_i	4, 4, 4, 3, 4, 2, 4
i	4, 7, 11, 1, 2, 3, 5

$T_{4,4} = 53$

$T_{8,2} = 113$

$T_{8,4} = 130$

$T_{12,2} = 229 + 76 = 305$

$T_{12,4} = 337$

$T_{15,4} = 366$

$T_{19,4} = 387$

$T_{21,4} = 415$

$T_{25,4} = 450$

not right

X recheck

	j =	1		2		3		4	
	Q:	S	N	S	N	S	N	S	N
i									
1		3	30	16	33	12	6	4	20	13
2		4	30	18	21	12	20	6	20	13
3		2	30	16	21	12	20	6	20	10
4		4	5	8	11	4	6	4	15	8
5		4	30	21	17	18	6	4	20	13
7		4	5	8	17	23	6	4	20	13
11		4	5	8	33	49	20	6	28	26

Example

Sequence 1, 2, 3, 4, 5, 7, 11

yields 781 for 4 mach
755 for 3 mach.

---

develop a value for when a machine is ready to start running instead of when it's done the 1st part.

Example 2

$T_{3,4} = 107$

$T_{7,4} = 78 + 133 = 211$

$T_{9,4} = 180 + 90 = 270$

$T_{13,4} = 242 + 53 = 295$
alt $270 + 47 = (317)$

$T_{17,4} = 279 + 149 = 428$

$T_{21,4} = 393 + 126 = 519$
$524 + 13 = (537)$

$T_{25,4} = 430 + 261 = 691$
$755 + 26 = (781)$

$T_{3,2} = 90$

$T_{7,2} = 192$

$T_{9,2} = 254$

$T_{13,2} = 283$

$T_{17,2} = 411$

$T_{21,2} = 393 + 109 = 502$
$411 + 109 = (520)$

$T_{21,2} = 520 + 229 = 749$

$T_{3,3} = 94$

$T_{9,3} = 198$

$T_{9,3} = 260$

$T_{13,3} = 287$

$T_{17,3} = 415$

$T_{21,3} =$
$= 524$

$T_{21,3} =$
$= 749 + 6 = 755$

check out

$$T_{13,4} = MAX\left[\left(T_{9,4} + (S_{i,4} + Q_i r_{i,4})\right), \left(\sum T_{u,1} + T_{i,4}\right)\right]$$

$(270 + 15 + 32) = 317$

$(242 + 53) = 295$

also delay in process (see T_{21} line)

General Systems Representation

analogous to time flow -

might chemical reactions in stages
- refining
- silicone

Study of railroad in terms of switching

Initial state matrix - "Jobs"
Initial state matrix - Facilities
relationships matrix
Jobs vs Facilities

Operational Rules -

Final state matrix - Jobs
Final state matrix - Facilities

Example of McCombie model -
Cafeteria
Car wash
Army processing Center (Exam, Tests, etc.)
Drivers license
Pipe Line

General System Representation

Need to express --

non-productive operation

"null product"

wait, machine idle time — caused by ~ conditions

- monetary flow
- inventory
- traffic flow
- refinery
- airline

Consider "game theoretic" aspects -- is this approach to quality problem.

State variables:

- Initial States
- down time, repair,

Info delay vs physical delay -

shop system analysis

Manufacturing Systems Simulation

~~Heller was in Friday for 2 hours.~~

~~any word from Colley~~?

~~3/25~~

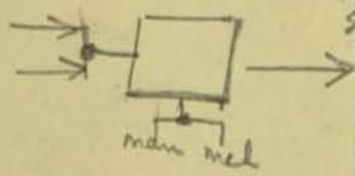
classes
no.
shift
skills
men }
mach } transformation capability
time

ASSIGN RESOURCE FOR T time
TO PROVIDING TRF Z

mat'ls - availability, qty - expendable
information.

to Produce a result R

• and junction



$t_f = if(M_1 \wedge I_1 \wedge A_1 \wedge B_1)\, t_i + Q\sigma_Z$, or $t_i = t_i + 1$

M_1 A_1 P_1, t, l

I_1 B_1

operations performed by
men I_1
on M_1
for time r

$Q_f = [Q_i \cdot N]$, wait at time t_f

$N = p(\text{good})$

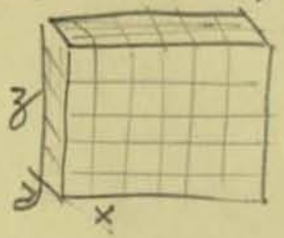
σ_Z = operation time for trf Z

Each physical location can be described by a 3 dimensional representation

(material)
any "product" Then can be represented:

$\{C\}$ parts characteristics (represented by an identification no and spec no.)

$L_{x,y,z}$ use grid or bay location, nearest foot



if z omitted assume on ground floor. column no. may be adequate

nearest foot, 5 foot / - on height can use "floor" Code

t Time value to agreed accuracy must be at least 6 detail 7 minimum increment - may .1 minute is adequate or 1 min - etc - LST-C might be 10 min or .1 hr. often measured from start of simulation time - use days, weeks, shifts - etc

$Q(M)$ typically integral quantity of product - includes unit of measure.
units, lbs, feet,
— cost per unit mass

DWG No | SPEC No | LOC | TIME (HRS) | QUANT (UOM)
121A361-127 , 9 , A15 , 15.26 , 20 (PCS)

Manufacturing Systems Simulation

Processing Facility -
man (non mach)
man - mach
auto mach
process station

$\{\ \}$ skills characteristic
typically machine class -
man labor pool -
16" Milling Mach
10' punch press

$L_{x,y,z}$ Shop location -

$n_{a,b}$ transformation rate for $\{C_a\} \rightarrow \{C_b\}$
consider setup, run time, learning curve
time per unit try - differentiate loaded-in station (drill press) - a constant
man and/or like annealing oven.

M Individual mach plant no -
needed to properly express discreteness
of system elements.

C Capacity (time) - expressed as
start time avail to fin time avail
may be approximated by shift representation

— cost/unit time

Transportation Facility

Identification
from, to

Skills – wt load

f transportation from
$L_a \rightarrow L_b$
consider max, min times

Cost/unit mo

Inspection - Test Facility -

discrimination capability

Branching capability
Cost/unit time

Manufacturing Systems Simulation

consider time delays -

queue
transport
transform
inspect -

} delay function

Consider machine performance concept -
ratio of good to total
stochastic performance

introduce statistical variation for oper, setup times

Input concept -
shop order no.
routing - alternative routing?
time at each station - oper, setup
material reqts - Qty, type
General concept of measures -
Dir. $, time,

output reports -
report generator

Resources which are "absorbed"
"digested"
"used up" → Inventory problems
qty, time, place
identity
- quality

resupply
location

Resources which cannot be accumulated
men, machines
must use at time or not avail
stochastic variation in effectiveness
transformation / unit time
may consider rate of application

quality of transformation → scheduling
loading
balancing
dispatching
class - task capability

transformation
testing
unit movement

Dies, fixtures,

Information resources

control signals -

$$\{m_i\}\{\sigma_1, \sigma_2, \ldots \sigma_s\} \rightarrow P_k$$

equations
qty
location
time

have to write math equations

$$R_{ij} m_i \rightarrow P_j$$

where R may be stochastic

time "load" equations are quite meaningless

DRAFT - 8/25/59

Subject: Systems Language

It's very enlightning to listen to an engineer or a procedures man describe a systems process operation. We typically get into quite a harange concerning the kind of hardware being used such as the kind of analog computer or the nature of the typewriter or items of this sort. These hardware-oriented descriptions are interlaced with the description of the movement of information or signals, the decision criteria for determining courses of action and the input and output formats.

We become particularly concerned with this problem when we try to build a simulator to try to describe the operation of a particular system. We must first make some sort of a flow chart or diagram to describe this systems operation and then we must sequentially determine an algorithm for approximating this systems operation that will be computationally effective on the computer we have available. Even in flow charting the systems problem, we tend to get involved in programming and modeling considerations instead of in just describing the systems operation. The problem becomes even more acute when we consider that there are so many different systems which it is useful to simulate: physical flow, informational flow, inventory processing, etc. Wouldn't it be fruitful to have available a ~~bys~~ business systems language -- a special algebra -- which would enable either the engineer or the procedures man to describe the systems operation and then have the translator organized so as to convert this systems description into an operational computer program for the particular machine which has been chosen?

I have become convinced that there are great areas of similarity between all elemental systems. I believe that our language specialists have emphasized the differences instead of the commoness of patterns. For example, a defense system which has to select a priority of attacking targets and a business system which has to determine which job is to be worked on next are essentially equivalent problems: the assignment of a limited resource to the performance of needed work. This is not to say that the rules are necessarily the same but that the function performed is essentially identical.

There are certain analogies which might be drawn; for example: each initial work on computers was directed toward detailed programming of specific problems. Expensive work was done on numerical analysis in order to develop the most efficient algorithm for solving a particular set of equations. These were then directly programmed for computer solutions. It was extremely difficult and even well nigh impossible to exchange computer programs between different disciplines and different computer groups since the level of the program was so variable. However, what has happened over the past three years? The concept of FORTRAN has been introduced. This program is written in terms of basic calculation elements like multiply, divide, sine, square root, etc. now the engineer who can write out the equations for his problem need no longer concern himself with his method of programming, rather the computer will take over from these algebraic statements and prepare an operable computer program within the framework of the particular machine which the user wishes: 704, 709, 650, Burroughs 220, Transac 2000, etc.

If we in Production Control could discover these same kind of elements in business systems that would correspond to the multiply, divide, sine, etc. of the engineers language, then we would have a corresponding ability to prepare general purpose programs which would translate an algebraic statement of the systems operation into a computer program suitable for simulating the systems behavior.

If this idea works then we can sponsor work on the development of such a basic technique for the computer for which we choose to work with. We can avoid entirely the necessity of detail programming of future computer simulation models. This seems to be a very worthwhile objective and further research analysis might not only relieve our model building chores but could also give us a grasp and insight which would lead to major contributions in the field of production control rules.

B. Grad/pd

P+ Explosion Plan

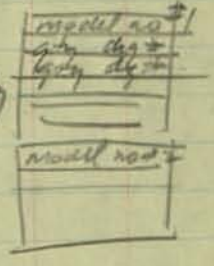
SO
model order cards
S.O. no.
model #
qty

sequence by model no.

S.O.
qty model #

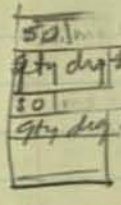
Select model lists 50 × 20 × 20 = 20,000 char

each entry single list
qty + dwg no —
list
20 char/entry



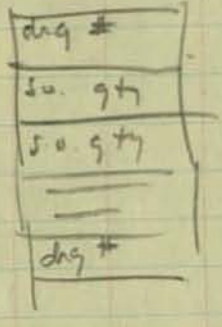
model assem Subassem →

create new tape as follows
model qty × entry qty



40 chars
40 chars
40 000 chars

Sequence by dwg #
consolidate common dwg nos

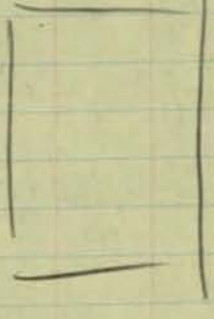


1000 entries
Worst case would be
with zero redundancy.

1/2 would not be assem.
500 assem entries

select assem lists — 500 × 10 × 20 = 100 000 char

create new tape as follows
qty × entry qty.



50 000 Parts

25000 mfd

25000 purch

5000 Subassemblies 5 items - 25000 entries

5000 Assemblies 10 items 50 000 entries

5000 models 20 items 100000 entries -

1 model

20 items : 10 Assem 10 part 10 parts

100 items : 10 assem 90 parts

50 item 50 part

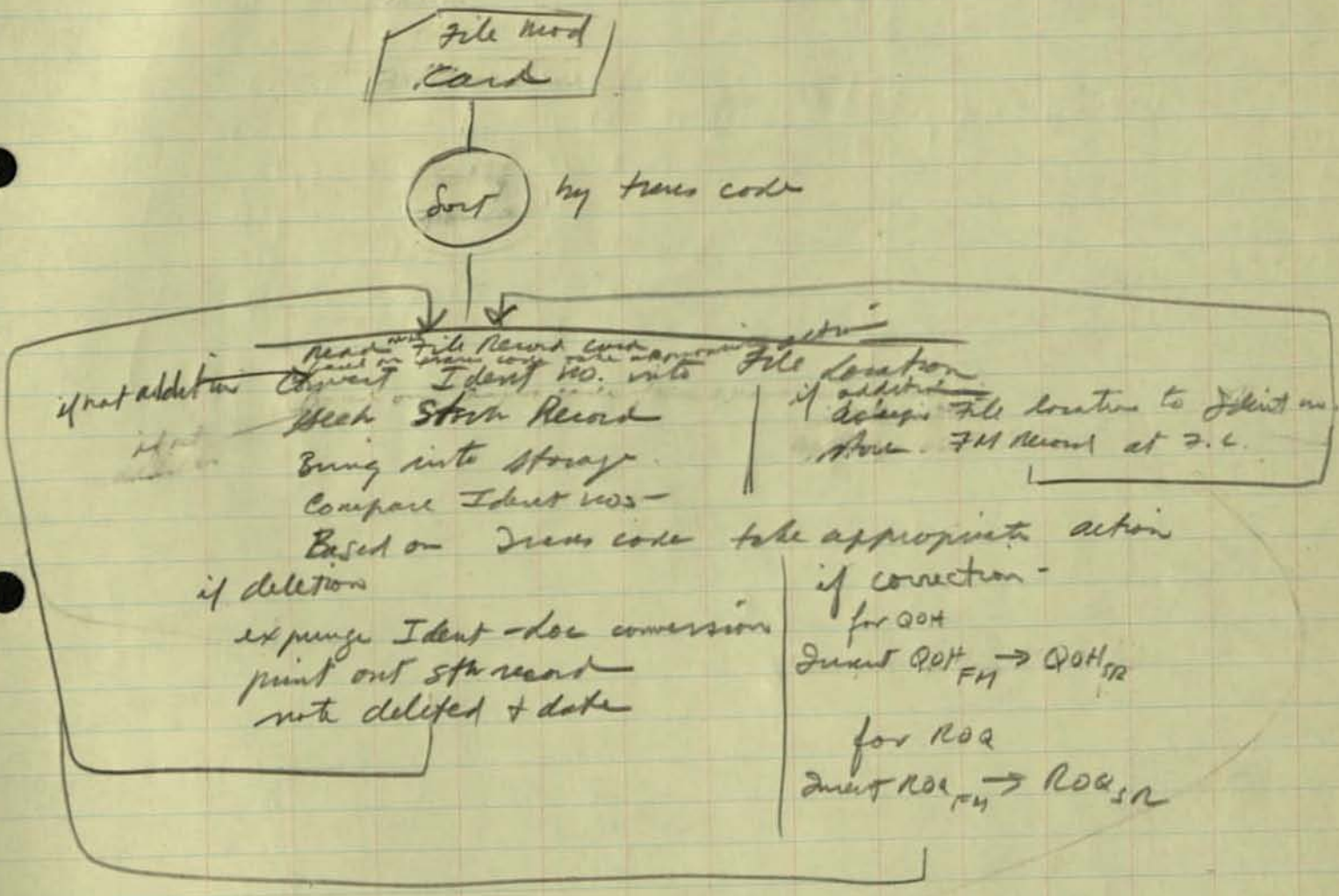
150 parts

Receive 50 ~~lines~~ orders per day 100 purch
50 mfd.

(mono line orders)

5 operations per mfd part

1 material description per part.


File mod Card
Sort
by trans code
Read in File Record card
Convert Ident No. into
Fetch Stock Record
Bring into storage
Compare Idents nos -
Based on Trans code take appropriate action
if not addition
if addition
Assign file location to Ident no.
Store FM record at F.L.
if deletion
expunge Ident - loc conversion
print out stk record
note deleted + date
if correction -
for QOH
Insert $QOH_{FM} \rightarrow QOH_{SR}$
for ROQ
Insert $ROQ_{FM} \rightarrow ROQ_{SR}$

Program

Read Program from Disk -
Set W to Zeros, rest to blanks
{Seek next segment 00000
{Read into Prog track 0
repeat for 1-9
Read 1st File Mod Card
→ track K
from K → W
Control or Transm code:
select next program step through a
character selector.

STOCK RECORD

IDENT No.	0-9	Alpha
DESCRIPTION	11-39	Alpha
Qty on Hand	40-45	numeric
Qty on Order	50-55	"
Reorder Pt	60-65	"
Reorder Qty	70-75	"

- 5000 stock items set aside 100 char per item -
10000 - 14999

Transaction Record - 500/day

IDENT No.	0-9	alpha	
Trans code	10	num	receipt = 5 ; with = 6
Qty	11-15	num	

File modification Record - 100/day - all done 2nd shift -

Ident no.	0-9	alpha	
Trans Code	10	num	addition = 2, deletion = 1, cor to QOH = 3, cor to ROQ = 4
Desc	11-39	alpha	
QOH	40-45		
QOO	50-55		
ROP	60-65		
ROQ	70-75		

Store Programs 00000 - 00999
File Mod Prog 00000 - 00009

Store 100 Zeros in location 01000

COLLINS RADIO COMPANY

CEDAR RAPIDS, IOWA, U.S.A.

CABLE
COLINRAD

December 19, 1957

Mr. Burton Grad, Specialist
Manufacturing Service
Production Control Service (Room 2401)
General Electric Company
270 Lexington Avenue
New York 22, New York

Dear Sir:

On September 6th I wrote to you, describing our use of matrix algebra for paper work flow analysis. Did you find any of the information helpful?

I would appreciate receiving further information about work you may be doing in this field.

Yours very truly,

COLLINS RADIO COMPANY

John L Gable

John L. Gable
Industrial Engineering Div.

JLG/jb

COLLINS

COLLINS RADIO COMPANY

CEDAR RAPIDS, IOWA, U.S.A.

CABLE
COLINRAD

September 6, 1957

Mr. Burton Grad, Specialist
Manufacturing Services
Production Control Service (Room 2401)
General Electric Company
570 Lexington Avenue
New York 22, New York

Dear Sir:

Thank you for your letter of August 21st.

I have been assigned to the project of investigating and possibly developing an Integrated Data Processing and Analysis System which will load and schedule our machine shop and fabrication areas for maximum profit. Our stockrooms presently handle over 40,000 different active part numbers each month. There are usually between 10 to 30 new equipment types started through the shop each month; the lead time between design release by engineering and delivery of the first units to Finished Goods Stock is often only three or four months.

We feel that the development of such a system may take two to three years. We hope to learn enough about Linear Programing and other Operations Research approaches to be able to utilize them in our solution.

During my research early this year, I skimmed through "Electronic Computers and Management Control" by Kozmetsky and Kircher; an appendix to their book introduced me to the matrix method of paper work analysis.

I tried applying the technique to the activities covered by our Manufacturing Order (shop authorization to do work), PD 60 (combination move ticket and location control card), Raw Material Requisition, and Daily Labor Distribution Cards (job time cards). My primary interest was time reporting, I therefore adopted the following ground rule--the introduction of "Written By", "Received By", or "Date" information will not cause a form to be classified as a Source Document. This greatly reduced the size of the matrix. This approach is, I think, the direct opposite of yours. I was primarily interested in the number of times part number, for example, was repeated before a given report was prepared.

Many forms contain information which is used only for what I call "local control" and is not transmitted to a higher report level. Thus the name of the person who counts parts may be necessary for disciplinary purposes if the count should be found to be incorrect; it is not necessary for the weekly Shop Labor reports. As I interpert the matrix system, the final matrix $M_R = \prod M_{R-i}$ shows the total number of times a piece of information would be available if it had been retained at all report levels.

I considered using some symbol--say an asterisk--beside a number which represented local control information. The number of asterisks beside the original entry would tell the number of report levels which would retain the information. Each time a matrix would be multiplied during the process of obtaining the final report level, the number of asterisks beside a number would be reduced by one. Any number with only one asterisk beside it (indicating that the information was not carried to a higher report) would be treated as a zero for the next matrix multiplication. Thus any report level would show only the number of times information was actually repeated.

This problem of pseudo-redundancy could be handled, I suppose, by utilizing an operational matrix of 1's to represent retained information and 0's for dropped or local control information. I found that most local control information is not carried to a higher report level so usually only one asterisk is used; I therefore think that this technique is better than introducing an additional matrix multiplication at each level.

It has been about six months since I did the work on matrix analysis; I hope that these recollections will be of help to you.

I would appreciate receiving any information or ideas which you might have on the problem of machine loading and scheduling for a job shop type of machine shop.

Yours very truly,

COLLINS RADIO COMPANY

John L Gable

John L. Gable
Industrial Engineering Division

JLG/jb

August 20, 1957

Mr. David R. Seidman
IDP Dept. North American Aviation
Columbus 16, Ohio

Dear Mr. Seidman:

During a study conducted early this year we have tried to analyze some of our records in accordance with the suggestions made by Mr. Irving Lieberman in the material he published in Management Sciences. We tried to apply these techniques in a business making fractional motors using, as a consultant, Mr. Richard Canning from Canning, Sisson and Associates. Unfortunately, our results were quite disappointing so we contacted Mr. Lieberman to ask if any others had been able to successfully apply these principles. He was good enough to suggest your name and I wonder if we might exchange information on this subject.

Our difficulties developed around our apparent inability to uniquely describe a field of information. In other words, when a field was a "date" we asked the question of ourselves--"What date?" When we had a "name", we said, "Whose name?" This was done because of the implication in Mr. Lieberman's work that each of these fields should represent a particular class of information such that whenever I speak of that class, I would always get the same value in a specific situation. With this philosophy, we found some 500 to 600 fields were required in analyzing 100 forms used in various stages of the manufacturing process. The matrices were virtually empty and the redundancy measures very low. Even our efforts to work toward a level by level interpretation quickly foundered on this "rock" of field identification.

If you are interested in exchanging your experiences in this area with us, I should be pleased to send you copies of the material that we prepared. If this doesn't prove feasible, I should appreciate your informing me so that I might ask Mr. Lieberman for other references.

Burton Grad, Specialist
MANUFACTURING SERVICES
PRODUCTION CONTROL SERVICE
Room 2401

BG/pmr

August 21, 1957

Mr. John L. Gable
Industrial Engineering Division
Collins Radio
Cedar Rapids, Iowa

Dear Mr. Gable:

During a study conducted early this year we have tried to analyze some of our records in accordance with the suggestions made by Mr. Irving Lieberman in the material he published in Management Sciences. We tried to apply these techniques in a business making fractional motors using, as a consultant, Mr. Richard Canning from Canning, Sisson and Associates. Unfortunately, our results were quite disappointing so we contacted Mr. Lieberman to ask if any others had been able to successfully apply these principles. He was good enough to suggest your name and I wonder if we might exchange information on this subject.

Our difficulties developed around our apparent inability to uniquely describe a field of information. In other words, when a field was a "date" we asked the question of ourselves--"What date?" When we had a "name", we said, "Whose name?" This was done because of the implication in Mr. Lieberman's work that each of these fields should represent a particular class of information such that whenever I speak of that class, I would always get the same value in a specific situation. With this philosophy, we found some 500 to 600 fields were required in analyzing 100 forms used in various stages of the manufacturing process. The matrices were virtually empty and the redundancy measures very low. Even our efforts to work toward a level by level interpretation quickly foundered on this "rock" of field identification.

If you are interested in exchanging your experiences in this area with us, I should be pleased to send you copies of the material that we prepared. If this doesn't prove feasible, I would appreciate your informing me so that I might ask Mr. Lieberman for other references.

Burton Grad, Specialist
MANUFACTURING SERVICES
PRODUCTION CONTROL SERVICE
Room 2401

BG/pmr

FILE: ISP - Statistical

4 month Orders Rec'd Summary
AB-DB Switchboard · 1958
excluding E, F, M, T

No. of Items

	Jun	Jul	Aug	Sep	Total
17	21	6	3	25	55
Y	215	167	55	136	573
other	477	318	188	587	1570
Stock	563	601	272	641	2077
Total	1276	1092	518	1389	4275

No of Instruments

17	306	73	26	100	505
Y	303	268	113	233	937
other	1355	731	527	2133	4746
Stock	1914	1961	1076	1839	6790
Total	3878	3053	1742	4305	12978

B Grad
10/2/58

JSP statistical

11/12/58

No. of Direct Labor People — Final Asbly

AB	2-V	9 D.L. + 1 Packer	+ ½ Female Stock sheet writer
DB	2-V	9 DL + 1 combination packer + Mixerman	
AB	3-V	6 D.L. + 35% of one Packer + 35% of one writer	

J A DeFusco

DSP Mfg Cont

Fiscal June -

For instance - 24 working days

Use random number table
2 digit numbers 01 - 24 correspond to working days. 00, 25-99 are discarded.
Shuffle June cards thoroughly.
Assign working days in sequence per random no. table. Sort cards into 5 piles:

① AB Stock } 31 models only
② DB Stock
③ AB A+V
④ AB Other
⑤ DB

Sequence each pile by working day no.

Assign Request date to each card in pile ③, ④ and ⑤

Use random no. table again in following way - 2 digit fields

Random #	value of i
00 - 14	i = 00
15 - 64	i = last digit value 1 - 10
65 - 74	i = 11 for 5,6; 12 for 7,8; 13 for 9,0 14 for 1,2; 15 for 3,4
75 - 89	i = random no. - 59 i.e. i = 16 for 75; i = 17 for 76; 30 for 89
90 - 99	i = 31 for 90; 32 for 91; 34 for "

IBP Mfg Cont

For each day for piles ③, ④, ⑤ summarize total requirements for each value of i. Post on appropriate Input Form.

For each day for pile ① and pile ② accumulate qty for each model no. Assign model code nos in sequence by daily review order. Post 2 typ.

J Hos

prepare paper on automatic program
testing through a Latin Square approach
to structure table values -

Four measures of System Performance

Inventory level, variance

Customer service level, variance

Employment ~~stability~~ level, variance

Cost per unit produced, variance?

Delivery Cycle - mean, variance

Idea

Generalized Test Generator

Concept built on

Alex Star Tables

Basic User-oriented operations

LOAD

SCHEDULE

EXPLODE

Ideas

Concept of Table operations

define entire set of Field operations
in Table context –
all files to be set up as Tables

- concept of order or arrange
- concept of summation of entries in a column
- one single Sequential memory (list-type)
- using dictionaries (index look ups) for
all row identifications

leads naturally to TSB1010 approach
could experiments on this using
Inventory problem or
Feedback controlled Release Plan

idea per H Markowitz

write up Organizational Planning concepts:

Operating elmt -

Operating - vert int unit functional prod line or major comp oriented mgr responsible for short range (day-to-day) operations

Planning - proj mgr - pools of people of various technical skills all work done on project basis. assign project mgr tech ldr with necessary supporting personnel including adm, writing, QA, engrg, etc. There may be an "administrative" organization for personnel purposes - but this is non-"operational"

Services -

entirely a planning organization - reasonable sized pools of manpower in various business function specialties - engrg, acctg -
other pools of manpower in technical specialties - mathematics, computer, writers, etc
all work performed on a project basis - led by a technical leader - provides necessary technical support - independent concept of administrative and "operational" organization

Ideas

~~Field Re~~

Instead of physically moving info from one loc'n to another it would be sufficient to leave it alone & simply give it a new name (extension of Leit concept)

per H Tassenfeld

Single Index concept.

$I = f(a, b, \ldots, z)$

Ideas

to the left most likely to
succeed column

to the top most likely to
fail now

This avoids necessity or intent
in not returning to top of
column.

multiple "OR" in TABJOL

Field name

Text	or constant	0..9

0 = AND

1..9 = no of succeeding blocks to skip if succeeding element is zero

otherwise increment downward.

DRAFT - 12/10/59

A Computational Procedure for Multiple Decision Criteria

Burton Grad

In establishing decision rules to guide the choice between alternative courses of action, it is often necessary to consider multiple decision criteria. Typically, there will be a dominant factor and various secondary considerations. It may be possible to establish explicit relative weights so that a direct equation can be used to determine the relative value of a particular course of action. This is a problem which has been thoroughly covered in the literature (Churchman, et. al. Operations Research).

But in many cases we are only able to rank the decision criteria and state if the value of the first criteria is identical for two or more courses of action then the choice will depend on the relative standing of the second criteria, and so forth.

There seems to be a logical hierarchy in the degree to which we are able to understand and handle automatically a decision making process. At the highest (and least automatable level) we do not know explicitly what parameters should be considered nor are we certain as to what courses of action are available to us or may be successful in accomplishing our objective. At the other extreme we have an explicitly stated choice system with stated criteria each of whose values states are precisely weighted and with alternate courses of action clearly defined and bearing a specific correspondence to relative criteria evaluations.

There are, naturally, many in between states. One useful in between level can be characterized as follows:

We wish to select from among j available jobs that job which will maximize some "pay off" function. However, the pay off function cannot be stated as an explicit equation. There are c criteria each of which to a greater or lesser extent may influence the job selection. Each job has associated with it a set of values, one for each of the significant criteria. The criteria are ranked in terms of relative importance; hence, criteria 1 is most important; criteria 2 next in importance; criteria c least important. Actually criteria 1 is dominant; it is only when there is a standoff (a tie) as regards c criteria 1 that the decision-maker will examine criteria 2, and so on.

It would be convenient to express this decision system so that a priority number could be computed which would cause the decision maker to make the correct selection immediately and directly without the necessity for multiple comparisons.

A solution to this need can be expressed in the following way:

1. Each job is characterized by a criteria value vector $(V_i) = V_1, V_2,$ etc., V_c.
2. It can be stated that V_1 dominates V_2, and V_{i-1} dominates V_i. This precedence domination relationship holds throughout for all criteria.
3. The decision rule is to select the job with the largest V_1; in case of a tie select the job with the largest V_2, etc. The same philosophy would hold for a rule which chose smallest V_1, V_2, etc.
4. We will construct a priority number (P) for each job as follows:

a. For each criteria we will establish a reasonable maximum value which we will call M_i.

Hence for each criteria

$$\left| 0 \leq V_i \leq M_i \right| \text{ all } j$$

b. For each job

$$P = \left[V_1 \left(\prod_{i=2}^{i=c} M_i \right)\right] + \left[V_2 \left(\prod_{i=3}^{i=c} M_i \right)\right] + \ldots + \left[V_c\right]$$

5. Job selection will then be on the basis of the P_j (Priority number for job j) choosing such that $P_{j*} \geq P_j$ for all j.

A simple example will illustrate the principle -- there are 5 jobs waiting to be ~~pxxxx~~ processed at a certain station. There are 3 significant criteria. The most important factor is the customer urgency code which is either 3 (highest), 2 or 1 (lowest). The second factor (which is significant only if there are two or more jobs of the same urgency code) is order value to the nearest thousand dollar. The highest value order is never as great at $10,000. The third criteria which comes into play to break ties if factors 1 and 2 are identical for two or more orders is the operation time at the next station expressed to the nearest minute. No operation time will be as great as 100 minutes.

Assume we have the following vectors for each of the jobs.

Job #	1	2	3	4	5
Criteria 1	3	2	3	3	1
2	8000	9000	3000	8000	6000
3	64	85	70	45	31

$M_1 = 4$

$M_2 = 10000$

$M_3 = 100$

$P_1 = 3 \times 10000 \times 100 + 80000 \times 100 + 64 = 3,800,064$

$P_2 = 2 \times 10000 \times 100 + 9000 \times 100 + 85 = 2,900,085$

$P_3 = 3 \times 10000 \times 100 + 3000 \times 100 + 70 = 3,300,070$

$P_4 = 3 \times 10000 \times 100 + 1000 \times 100 + 45 = 3,800,045$

$P_5 = 1 \times 10000 + 100 + 6000 \times 100 + 31 = 1,600,031$

Clearly

$P_{j*} = 3,800,064$

or $j = 1$

There is, of course, no necissity that the M_i values need to be power of 10. If they are not then the original value terms are not directly recognizable as they are in the previous example. This procedure then carries out the type of multiple decision rule described above.

Another class of multiple decision rules are characterized as follows:

The problem statement is as before in terms of desire to select among j available jobs each of which is characterized by a set of criteria values. The criteria are ranked in terms of relative importance. Criteria 1 still can be thought of as dominating criteria 2. What is changed is the definition of a tie. We state that a larger value for criteria 1 is dominant only if it is at least r units greater than the next largest value of that criteria.

A solution can be stated as follows:

1. Each job is characterized by a criteria value vector $(V_i) = V_1, V_2,$ etc., V_c

2. It can be stated that V_1 dominates V_2, and V_{i-1} dominates V_i.

3. The decision rule is to select the job with the largest V_1 if $(V_{1*} - V_{1\#}) \geq r$.

   Where V_{1*} is the largest value of criteria 1 among the j jobs. $V_{1\#}$ is next largest value of criteria 1. r is the prestated range value.

   In case of a tie, select the job with largest V_2 if it is sufficiently larger than the next largest V_2, etc.

4. We will construct a priority number (P) for each job as follows (This is not quite the same as the rule above, but it is a simplified version of it):

   a. For each criteria we will establish a series of interim range limits ($R_{i,k}$) such that

      $0 \leq R_{i,k} \leq M_i$

      where M_i is the reasonable maxim for criteria i.

      $R_{i,1} < R_{i,2} < \ldots R_{i,k} < R_{i,k+1} \ldots < R_{i,k}$

      where $R_{i,k} = M_i$

      $K = \frac{M_i}{r}$ $\quad K \equiv$ total number of ranges.

      $(R_{i,k+1} - R_{i,k}) = r$

   b. The following decision structure table illustrates the method for developing each element of the priority number.

V_i	≥ 0	$\geq R_{i,1}$		$\geq R_{i,k-2}$	$\geq R_{i,k-1}$
V_i	$< R_{i,1}$	$< R_{i,2}$		$< R_{i,k-1}$	$< R_{i,k}$
F_i	0	1		K-2	K-1

c. P (the priority number) =

$$\left[F_1\left(\prod_{i=2} K_i\right)\right] + \left[F_2\left(\prod_{i=3} K_i\right)\right] + \ldots \left[F_c\right]$$

This same concept can be extended to percent ranges though it is not clear how a direct percent of value difference could be processed; e. g. if two values are within 5% of one another or two values are less than 10 units apart then apply criteria 2. This would require a one for one differencing which does not lend itself to a direct calculation procedure.

Conclusions

Some progress has been made on changing multiple decision criteria rules from a compare and branch procedure to a computational approach. This should result in higher speed operation within a computer and for easier operation by operating personnel. These concepts could be applied directly to factory dispatching, material scheduling, etc.

A Computational Procedure for Multiple Decision Criteria

Burton Grad

In establishing decision rules to guide the choice between alternative courses of action, it is often necessary to consider multiple decision criteria. Typically, there will be a dominant factor and various secondary considerations. It may be possible to establish explicit relative weights so that a direct equation can be used to determine the relative value of a particular course of action. This is a problem which has been thoroughly covered in the literature (Churchman, et. al. Operations Research).

But in many cases we are only able to rank the decision criteria and state if the value of the first criteria is identical for two or more courses of action then the choice will depend on the relative standing of the second criteria, and so forth.

There seems to be a logical hierarchy in the degree to which we are able to understand and handle automatically a decision making process. At the highest (and least automatable level) we do not know explicitly what parameters should be considered nor are we certain as to what courses of action are available to us or may be successful in accomplishing our objective. At the other extreme we have an explicitly stated choice system with stated criteria each of whose values states are precisely weighted and with alternate courses of action clearly defined and bearing a specific correspondence to relative criteria evaluations.

There are, naturally, many in between states. One useful in between level can be characterized as follows:

We wish to select from among j available jobs that job which will maximize some "pay off" function. However, the pay off function cannot be stated as an explicit equation. There are c criteria each of which to a greater or lesser extent may influence the job selection. Each job has associated with it a set of values, one for each of the significant criteria. The criteria are ranked in terms of relative importance; hence, criteria 1 is most important; criteria 2 next in importance; criteria c least important. Actually criteria 1 is dominant; it is only when there is a standoff (a tie) as regards c criteria 1 that the decision-maker will examine criteria 2, and so on.

It would be convenient to express this decision system so that a priority number could be computed which would cause the decision maker to make the correct selection immediately and directly without the necessity for multiple comparisons.

A solution to this need can be expressed in the following way:

1. Each job is characterized by a criteria value vector $(V_i) = V_1, V_2,$ etc., V_c.
2. It can be stated that V_1 dominates V_2, and V_{i-1} dominates V_i. This precedence domination relationship holds throughout for all criteria.
3. The decision rule is to select the job with the largest V_1; in case of a tie select the job with the largest V_2, etc. The same philosophy would hold for a rule which chose smallest V_1, V_2, etc.
4. We will construct a priority number (P) for each job as follows:

a. For each criteria we will establish a reasonable maximum value which we will call M_i.

Hence for each criteria

$$\left| 0 \leq V_i \leq M_i \right| \text{ all } j$$

b. For each job

$$P = \left[V_1 \left(\prod_{i=2}^{i=c} M_i \right) \right] + \left[V_2 \left(\prod_{i=3}^{i=c} M_i \right) \right] + \ldots + \left[V_c \right]$$

5. Job selection will then be on the basis of the P_j (Priority number for job j) choosing such that $P_{j*} \geq P_j$ for all j.

A simple example will illustrate the principle -- there are 5 jobs waiting to be processed at a certain station. There are 3 significant criteria. The most important factor is the customer urgency code which is either 3 (highest), 2 or 1 (lowest). The second factor (which is significant only if there are two or more jobs of the same urgency code) is order value to the nearest thousand dollar. The highest value order is never as great at $10,000. The third criteria which comes into play to break ties if factors 1 and 2 are identical for two or more orders is the operation time at the next station expressed to the nearest minute. No operation time will be as great as 100 minutes.

Assume we have the following vectors for each of the jobs.

Job #	1	2	3	4	5
Criteria 1	3	2	3	3	1
2	8000	9000	3000	8000	6000
3	64	85	70	45	31

$M_1 = 4$

$M_2 = 10000$

$M_3 = 100$

$P_1 = 3 \times 10000 \times 100 + 80000 \times 100 + 64 = 3,800,064$

$P_2 = 2 \times 10000 \times 100 + 9000 \times 100 + 85 = 2,900,085$

$P_3 = 3 \times 10000 \times 100 + 3000 \times 100 + 70 = 3,300,070$

$P_4 = 3 \times 10000 \times 100 + 1000 \times 100 + 45 = 3,800,045$

$P_5 = 1 \times 10000 + 100 + 6000 \times 100 + 31 = 1,600,031$

Clearly

$P_{j*} = 3,800,064$

or $j = 1$

There is, of course, no necissity that the M_i values aeed to be power of 10. If they are not then the original value terms are not directly recognizable as they are in the previous example. This procedure then carries out the type of multiple decision rule described above.

Another class of multiple decision rules are characterized as follows:

The problem statement is as before in terms of desire to select among j available jobs each of which is characterized by a set of criteria values. The criteria are ranked in terms of relative importance. Criteria 1 still can be thought of as dominating criteria 2. What is changed is the definition of a tie. We state that a larger value for criteria 1 is dominant only if it is at least r units greater than the next largest value of that criteria.

A solution can be stated as follows:

1. Each job is characterized by a criteria value vector $(V_i) = V_1, V_2,$ etc., V_c

2. It can be stated that V_1 dominates V_2, and V_{i-1} dominates V_i.

3. The decision rule is to select the job with the largest V_1 if $(V_{1*} - V_{1\#}) \geq r$.

   Where V_{1*} is the largest value of criteria 1 among the j jobs. $V_{1\#}$ is next largest value of criteria 1. r is the prestated range value.

   In case of a tie, select the job with largest V_2 if it is sufficiently larger than the next largest V_2, etc.

4. We will construct a priority number (P) for each job as follows (This is not quite the same as the rule above, but it is a simplified version of it):

   a. For each criteria we will establish a series of interim range limits ($R_{i,k}$) such that

   $$0 \leq R_{i,k} \leq M_i$$

   where M_i is the reasonable maxim for criteria i.

   $$R_{i,1} < R_{i,2} < \cdots R_{i,k} < R_{i,k+1} \cdots < R_{i,k}$$

   where $R_{i,k} = M_i$

   $$K = \frac{M_i}{r} \qquad K \equiv \text{total number of ranges.}$$

   $$(R_{i,k+1} - R_{i,k}) = r$$

   b. The following decision structure table illustrates the method for developing each element of the priority number.

V_i	≥ 0	$\geq R_{i,1}$		$\geq R_{i,k-2}$	$\geq R_{i,k-1}$
V_i	$< R_{i,1}$	$< R_{i,2}$		$< R_{i,k-1}$	$< R_{i,k}$
F_i	0	1		K-2	K-1

c. P (the priority number) =

$$\left[F_1\left(\prod_{i=2} K_i\right)\right] + \left[F_2\left(\prod_{i=3} K_i\right)\right] + \ldots \left[F_c\right]$$

This same concept can be extended to percent ranges thought it is not clear how a direct percent or value difference could be processed; e.g. if two values are within 5% of one another or two values are less than 10 units apart then apply criteria 2. This would require a one for one differencing which does not lend itself to a direct calculation procedure.

Conclusions

Some progress has been made on changing multiple decision criteria rules from a compare and branch procedure to a computational approach. This should result in higher speed operation within a computer and for easier operation by operating personnel. These concepts could be applied directly to factory dispatching, material scheduling, etc.

Pat - pls type draft copy - 1+1

[redacted]
TIS - 307

A computational procedure for multiple decision criteria

By Burton Grad

In establishing decision rules to guide the choice between alternative courses of action it is often necessary to consider multiple decision criteria. Typically there will a dominant factor and various secondary considerations. It may be possible to establish explicit relative weights so that a direct equation can be used to determine the relative value of a particular course of action. This is a problem which has been thoroughly covered in the literature (Churchman, et al Operations Research). But in many cases we are only able to rank the decision criteria and state if the value of the first criteria is identical for two or more courses of action then the choice will depend on their relative standing of the second criteria, and so forth.

There seems to be a logical hierarchy in the degree to which we are able to understand and handle automatically a decision making process. At the highest (and least automatable level) we do not know explicitly what parameters should be considered nor are we certain as to what courses of action are available to us or may be successful in accomplishing our objective. At the other extreme we have an explicitly stated choice system with stated criteria and value states are precisely weighted and with alternate courses of action clearly defined and bearing a specific correspondence to relative criteria evaluations.

There are, naturally, many in between states. One useful in between level can be characterized as follows:

We wish to select from among j available jobs that job which will maximize some "pay off" function. However the pay off function cannot be stated as an explicit equation. There are c criteria each of which to a greater or lesser extent may influence the ~~connection~~ job selection. Each job ~~on the~~ has ~~it~~ associated with it a set of values, one for each of the significant criteria. The criteria are ranked in terms of relative importance; hence criteria 1 is most important; criteria 2 next in importance; criteria c least important. Actually criteria 1 is dominant; it is only when there is a standoff (tie) as regards criteria 1 that the decision-maker will examine criteria 2, and so on. It would be convenient to express this decision system so that a priority number could be computed which would cause the decision maker to make the correct selection immediately and directly without the necessity for multiple comparisons.

A solution to this need can be expressed in the following way:

(1) Each job is characterized by a criteria value vector $(V_i) = V_1, V_2, \text{etc.}, V_c$

(2) It can be stated that V_1 dominates V_2, and V_{i-1} dominates V_i. This precedence domination relationship holds

throughout for all criteria

(3) The decision rule is to select the job with the largest V_1; in case of a tie select the job with the largest V_2, etc. The same philosophy would hold for a rule which chose smallest V_1, V_2, etc.

(4) For each job we will construct a priority number (P_j) as follows:

(a) for each criteria we will establish a reasonable max, min value which we will call ~~Max~~ M_i.
Hence for each criteria
$$0 \leq V_i \leq M_i \quad \text{all } j$$

(b) for each job
$$P_j = \left[V_1 \left(\prod_{i=2}^{i=c} M_i\right)\right] + \left[V_2 \left(\prod_{i=3}^{i=c} M_i\right)\right] + \ldots + \left[V_c\right]$$

(5) Job selection will then be on the basis of the P_j (Priority number for job j) choosing such that $P_{j*} \geq P_j$ for all j

A simple example will illustrate the principle. There are 5 jobs waiting to be processed at a certain station. There are 3 significant criteria. The most important factor is the customer urgency code which is either 3 (highest), 2 or 1 (lowest). The second factor (which is significant only if there are two or more jobs of the same urgency code) is order value to the nearest thousand dollar. The highest value order is never as great as $10,000. The third criteria which comes into play to break ties if factors 1 and 2 are identical for two or more orders is the operation time at the next station expressed to the nearest minute. No operation time will be greater than 100 minutes.

Assume

~~If~~ we have the following ~~for~~ vectors for each of the jobs.

Job #	1	2	3	4	5
Criteria 1	3	2	3	3	1
2	8000	9000	3000	8000	6000
3	64	85	70	45	31

~~$P_j \cdot V$~~

notation where we could say that

$V_{1,1} = 3$

$V_{1,3} = 3$

$V_{3,1} = 64$

$M_1 = 4$

$M_2 = 10000$

$M_3 = 100$

$P_1 = 3 \cdot 10000 \cdot 100 + 8000 \cdot 100 + 64 = 3,800,064$

$P_2 = 2 \cdot 10000 \cdot 100 + 9000 \cdot 100 + 85 = 2,900,085$

$P_3 = 3 \cdot 10000 \cdot 100 + 3000 \cdot 100 + 70 = 3,300,070$

$P_4 = 3 \cdot 10000 \cdot 100 + 8000 \cdot 100 + 45 = 3,800,045$

$P_5 = 1 \cdot 10000 + 100 + 6000 \cdot 100 + 31 = 1,600,031$

Clearly

$P_{j^*} = 3,800,064$

or $j^* = 1$

There is, of course, no necessity that the M_i values need to be powers of 10. If they are not then the original value terms are not direcly recognizable as they are in the previous example. This procedure then carries out the type of multiple decision rule described above.

Another class of multiple decision ~~making~~ rules are characterized as follows:

The problem statement is as before in terms of desire to select ~~to~~ among s available jobs each of which is characterized by a set of criteria values. The criteria are ranked in terms of relative importance, ~~Hence~~ criteria 1 ~~is~~ ~~not completely~~ dominating criteria 2. What is changed is the definition of a tie. We state that a larger value for criteria 1 is ~~only~~ dominant only if it is at least r units greater than the next largest value of that criteria.

A solution can be stated as follows:

$r_i = \frac{M_i}{r}$

(1) ~~Each~~ Each job is characterized by a criteria value vector $(V_i) = V_1, V_2, \text{etc}, V_c$

(2) It can be stated that V_1 dominates V_2, and V_{i-1} dominates V_i.

(3) The decision rule is to select the job with the largest V_1 if $(V_{1*} - V_{1\#}) \geq r$.

where V_{1*} is the largest value of criteria 1 among the j jobs. $V_{1\#}$ is next largest value of criteria 1. r is the permitted range ~~of value~~.

In case of a tie, select the job with largest V_2 if it is sufficiently larger than the next largest V_2, etc.

(4) We will construct a priority number (P) for each job as follows (This is not quite the same as the rule above, but it is a simplified version of it):

(a) for each criteria we will establish a series of interim range limits ($R_{i,k}$) such that

$$0 \leq R_{i,k} \leq M_i$$

where M_i is the reasonable maximum for criteria i.

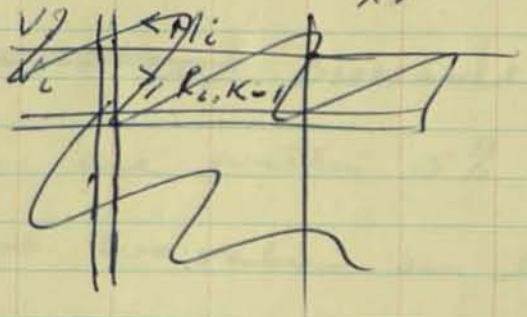
$$R_{i,1} < R_{i,2} < \ldots R_{i,K} < R_{i,K+1} \ldots \text{etc} < R_{i,K}$$

where $R_{i,K} = M_i$

$K = \frac{M_i}{r}$ K = total number of ~~state~~ ranges.

$$(R_{i,k+1} - R_{i,k}) = r$$

(b) The following decision structure table illustrates the method for developing each element of the priority number



V_i	≥ 0	$\geq R_{i,1}$	$\cdots$	$\geq R_{i,K-2}$	$\geq R_{i,K-1}$
V_i	$< R_{i,1}$	$< R_{i,2}$	$\cdots$	$< R_{i,K-1}$	$< R_{i,K}$
F_i	0	1	$\cdots$	K-2	K-1

(c) P(the priority number) =

$$\left[F_1\left(\prod_{i=2}^{i=c} K_i\right)\right] + \left[F_2\left(\prod_{i=3}^{i=c} K_i\right)\right]$$

$$+ \cdots \left[F_c\right]$$

This same concept can be extended to percent ranges though it is not clear how a direct percent or value difference could be processed; e.g. if two values are within 5% of one another or two values are less than 10 units apart then apply criteria 2. This would require a one for one differencing which does not lend itself to a direct calculation procedure.

Conclusion:

Some progress has been made on changing multiple decision criteria rules from a computer and branch procedure to a computational approach. This should result in higher speed operation within a computer and for easier operation by operating personnel. These concepts could be applied directly to factory dispatching, material scheduling, etc.

1956 - Reg'n Transmission
34 frame Systems Project

1955 Scheduling Research
Production
Warehouse Stk control - W+C

1953 - NDA - Otis Elevator
Telemeter - Hood Rubber

1954 Gen Tb Stkroom
D+O Scheduling
Stk Cont - LST-G
D.P.S.
EOQ

Introduction — BGrad — 15mins.

Welcome · ISP Pilot Workshop

Purpose -

- . depth exploration of ISP rec. techniques
- . aimed at encouraging use through understanding and discussion
- . to develop a framework for future courses

Introduction of people:

(use place cards, roster)

name, Section, Department

note team members, GA Walker : arrangements

Study Framework

- . relatively brief presentations by appropriate team member
- . Extensive discussion, problem analysis
- . Comparison with other approaches
- . Application to other problems
- . Somewhat unstructured as regards to time and introduction of certain subjects - but there are general timing objectives and a general plan as to when we wish to introduce each topic.
- . We'll be going at very high speed -- the course is aimed primarily at those who have done work in this field previously - DON'T GET DISCOURAGED IF YOU MISS SOME POINTS - They'll probably be repeated later in a different context - or you can ask at breaks, lunch or in the

con't'd Introduction -

evenings ... see the team members

- You will be given copies of various reports each day. These describe in detail the subjects to be covered that day. We would recommend reading the material that same night while it's still fresh in your mind - this may suggest questions to you or topics on which you would like to have additional discussion.
- Please keep notes - especially of discussions since much of this may not be in the manuals.
- MOST IMPORTANT: Your participation, presentation of examples, asking questions.

Subject matter

(see Program)

The plan for Monday:

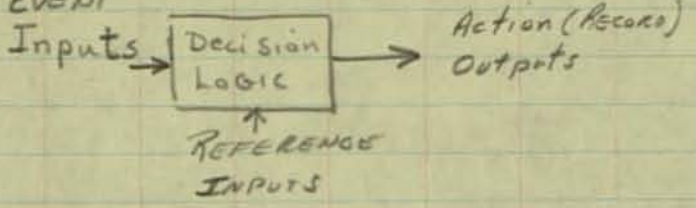
Systems Concept - Bryan
Engineering - Dan Langenwalter
Decision structure Tables - Don Kavanagh

THE SYSTEMS CONCEPT

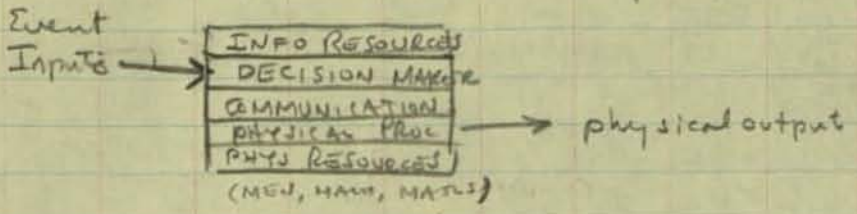
Concept of Vertical Integration ---
Compare this to "flow shop" factory concept ... combined information + physical processing system

Definition of a System ---
Convenient to think in terms of actions or results rather than functional kinds of work. --- multiple function

Information processing System:



Mixed Info + physical processing systems



MAIN LINE SYSTEM
That set of activities which convert a customer's order into a finished product

UNDER STAND BEFORE YOU DESIGN
Product, Market
Facilities: physical, information processing
DECISION LOGIC

cont'd Systems Concept

TECHNIQUES FOR UNDERSTANDING:
DATA Gathering + Analysis (Book 2 of ASP)

Generic Model List
Customer Specifications
Information Process charting

Concept of aggregation + segregation

ABC product, parts, materials

Careful record, file analysis to establish current information fields and use of information

Establish today's costs

SYSTEMS APPROACH ---
See the problem as a whole
don't try to automate today's procedures
Reconceive the system with the new tools + principles

Engineering - of Langenwalter

1. Systems Approach.
2. Engineering relation to department objectives
3. Product design structure - causal relation
   Bottoms up approach
4. ISP. Results in Engineering
5. Requisition Edit Sheet. - customer specs
6. Model Numbers -
7. Structure Tables. <TFK: Str. Tables>
8. Drawings -
9. Applications + comparisons
   Power Transformer
   Medium Transforme.
   Medium A.C. Motor and Generator.
   Low Voltage Switchgear.
   Industry Control
   Large Steam Turbine.
10. Summary.

Decision Structure Tables

(1) work problem -
Medium Transformer -
Base Problem

Cost Accounting - SA MacMullen

1. Cost Accounting's Interest in the Integrated Systems Project
2. Relationship of Cost with New Techniques Developed by ISP
3. Present Method of Developing Costs
4. New Method of Determining Costs (Cost Structure Tables)
5. Usefulness of New Information to Cost Accounting

Span
12/29/59

Systems Oriented and Computer Oriented Languages

Format (physical layout)

Language (instructions, conventions restraints, symbolism)

Jargon (Specific terminology)

Classes of operations -

Input - Output

Formula Evaluation

Comparison + Branching, Relational Logical

Information Rearrangement

History -

	Instr Addr	Instr	operand address	Read by Comp	1-1 mod Instr
machine language	Abs.	Coded	absolute	Yes	Yes
Assembly language	Rel	Mnem.	Rel	No	Yes
FORTRAN	Rel	Implied	Named	No	No

Deductions problem

Input fields:

Salary current week

~~No of Dependents~~

No of Dependents

Accum Salary YTD

Accum FICA YTD

Accum WTAX YTD

BOND DEDUCTION AMT

MIN BOND PURCH AMT

ACCUM BOND ACCRUAL

PERCENT PARTICIPATION SVG-SEC

PERCENT OF S-S FOR STOCK

ACCUM SVGS-SEC TOTAL

Output:

~~Updts~~ YTD fields

BONDS

current week Deductions

005/

ACCUM SALARY YTD	< 4800	≥ 4800
GO TO	010	025

010/015

FICA DED	(Salary current wk) * .03
ACCUM SALARY YTD	(~~Accum Salary YTD~~) + (Salary current wk)
ACCUM FICA YTD	(~~Accum FICA YTD~~) + (FICA DED)

015/

ACCUM FICA ~~YTD~~	≤ 144	> 144
GO TO	025	020

020/025

FICA DED	(ACCUM FICA YTD) − 144
ACCUM FICA YTD	144

025/030

WTAX DED	(Salary current wk) − (NO OF DEPENDENTS * 12)) * .20
ACCUM WTAX YTD	~~Accum~~ + WTAX DED

030/

BOND DED AMT/month	0	> 0
GO TO	050	035

035/040

ACCUM BOND ACC	+ BOND DED AMT/month

040/

ACCUM BOND ACC	< (MIN BOND PURCH AMT)	≥ (MIN BOND PURCH AMT)
GO TO	050	045

045/040 ~~ACCUM BOND ACC~~

~~(ACCUM BOND ACC) – MIN BOND PURCH AMT~~	
ACCUM BOND ACC	~~(MIN BOND PURCH AMT) – (ACCUM BOND ACC)~~ ACC – (MIN BOND PURCH AMT)
NO OF BONDS PURCH	+1

~~045~~ 049

desired features

easy to write (logical, easy to learn)

easy to debug

easy to change or modify

easy to understand or communicate (problem-wise)

flexible, unique, consistent

natural jargon (English, formulas)

common to many problem areas

convenient to use with a variety of computers --- make it easy to switch from one computer to another.

Easy to write Converters, Generators, interpreters

Provide reasonable memory + speed (input volume)

What's available ?

(1) FORTRAN, COMTRAN
ALGOL, COBOL

(2) Mathmatic, Flowmatic

(3) Autocoder, SOAP, SAP, CAGE

(4) Special Routines - Sort, Merge,
Report Generator, Linear Programming

Our basic interest is in languages
for Integrated Systems types of problems -
--- information processing --

Special features ---

Operation capability

indexing (subscripting)

word length (total, variable)

try out its applicability to Acctg
type problems

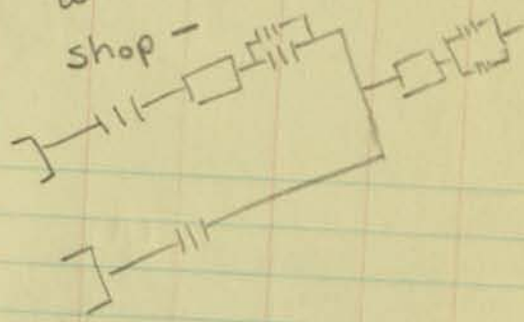
Logical "Or"

FORTAB

Herb Hindsley
MY 57
MOS 010 FORTAB converted into FORTRAN

```
1   IF (WDGIT - 15) 2, 6, 6
2.  ~~CONTINUE~~ NØØP CONTINUE
3   TIME2 = WDGIT
4   TIMG  = TIME2
5   GØ TØ 20
6   IF (WDGIT - 15) 11, 7, 7
7   IF (WDGIT - 100) 8, 8, 11
8   TIME2 = 125 * WDGID * WDGIT
9   TIMG = TIME2
10  GØ TØ 20
11  IF(WDGIT-100) 16, 16, 12
12  ~~CONT~~ NØØP
13  TIME2 = 1000 * WDGID * SQRT(WDGIT)
14  TIMG = TIME2
15  GØ TØ 20
16  ~~ERROR~~ GØ TØ Error Routine Location
```

a simple shop -



Can Mills example?

have them develop problem by asking questions -

Production Control Planning

Rep Unit Class

Control Group.

Time Control

Setback Cycle Chart

Unit

Load Impact Table

Load - Capacity Analysis -

- Stock Plug -
  - distrust
  - whse
  - mfd pt + asms
  - purch pts
  - raw mats
  - cont. parts

-

demand analysis, mean, variance by model
  ABC
Order Qty (Order Interval) , cost of ordering, cost of carrying
Safety Stock Analysis
  Lead Time + its variance
  demand variance
  Service level
  ~~inventory carrying cost~~

Outline - Course material

Production ~~Manufacturing~~ Control

1. Introduce the basic elements of a production control system.
   - cost promise
   - material ordering
   - schedule release
   - prod level
   - employee assignment
   - dispatching

2. Explore action of these elements ~~using~~ on a paper simulator of a simple process - à la Mills

3. Example of simple stock control problems showing protective stock and economic lot size concept

4. Example of the leveling problem showing normal information lag effect - reference Forrester work

Study problems

1. Raw material ordering

2. Make a paper simulator of Coal city area

3. Set up simple scheduling rule

4. Try out rule on simulator

Discussion

1. Bring out interrelationships of decisions
2. Emphasize inherent complexity, lack of intuition
3. Availability of new tools and techniques

Production Control wrap-up

1. Cycle charts
2. Use of control groups and sep. ^substitution^ unit classes
3. Load-Capacity Analysis
4. Effects of Flexible automation on inventory control and scheduling practices
5. Time ~~[illegible]~~ ^trigger versus^ ~~[illegible]~~ reorder point.
6. Mechanized Feedback

Electro Coil - Production Control

materials : wire -
coil support

parts : armature coil

facilities : storage
manufacturing station

Rep Unit Classes
Setback/cycle time per stn for Rep Unit class
Load per stn for Rep Unit class

Usage data for materials & parts ; variance information

Question ? Should we store certain Armature Coils
materials

Eco Lot Size calc.
Prod Sch calc.

Promise Rule

Control Group concept ; m/f, procurement

Automatic Ordering.

Assembly Release

Production level = labor employment level.

ISP Simulator

Concept of a simulator -

carry through a manual example

based on simple Inventory problem.

discuss ISP Simulator

Shops - 3 lines

rules = Asm Release
cost from
Labor decision.

demand -

measures - from os. request
del & promise
inventory
employment

Evaluation and Criteria — B grad

(Follow TIS R59 MS308)

(1) Basis for evaluation

(2) Summary of Costs, Saving + Benefits

(3) Direct Costs

(4) Indirect Personnel

(5) Information Processing Equipment

(6) Other Indirect Costs

(7) Investment

(8) Implementation Costs

(9) Sales

(10) Criteria for Application

Project Team Organization

to whom to report -

a Manager or a Technical leader (counselor)
-- an insider vs outsider

sources of personnel,
functions, skills, problems.

Personnel - full time vs part time
work for or assigned to
services,
Use of a Mfr, outside consultants,

Progress Reports ---

money, time

Operating Organization

use organization chart from 8/3/08 -

TABFOR

```
(P + A*C)*W**2 * L**2 /

4 * T**2 (W**2 + L**2)
```

---

note: under symbol permutations
there are minimum non redundant sets —
with any type 2 relation —
if all relations are the same
i.e. a*b*c ch n a+b+c
and not exponentials then no parentheses
are needed.

---

differentiate between literal +
non literal operands
Test for max length of 2 lists
check NL after in memory
consider effect of front in approach to
solution

Correct TABFOR tables in development
portion - not correct -
Do I have to know parentheses or not.

$a_3x^3 + a_2x^2 + a_1x + a_0$

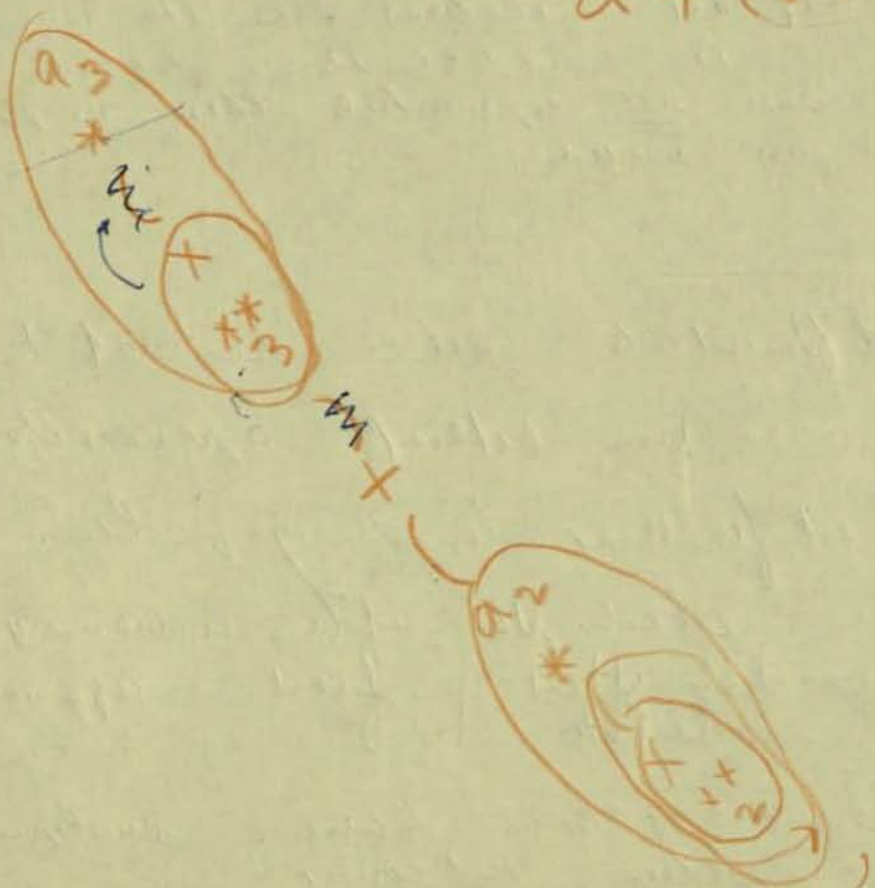
$= a_3 * x + a_2 * x + a_1 * x + a_0$

your language

$a_3 * (x ** 3) + (a_2 * (x ** 2)) +$

$(a_1 * x) + a_0$

$a + (b + c)$



$$\frac{(P+HC)\,w^2\,(L1)^2}{4T^2\,(w^2+(L1)^2)}$$

```
# ((((P+(H*C))*(W**2))*(L**2))/
  ((4*(T**2))*((W**2)+(L**2))))**#
```

```
TEMP 1 | +H
TEMP 1 | *C
TEMP 1 | +P
TEMP 2 | +W
TEMP 2 | **2
TEMP 2 | *TEMP 1
TEMP 3 | +L
TEMP 3 | **2
TEMP 3 | *TEMP 2
TEMP 4 | +T
TEMP 4 | **2
TEMP 4 | *4
TEMP 5 | +W
TEMP 5 | **2
TEMP 6 | +L
TEMP 6 | **2
TEMP 6 | +TEMP 5
TEMP 6 | *TEMP 4
TEMP 7 | +TEMP 3
TEMP 7 | /TEMP 6
```

~~IV-b~~
TABSOL

TABFOR

B. Grad
11/5/58

TABSOL - For Formula Statements

1. It would be desirable for a structure table to have the ability to directly express simple formulas which could be solved by TABSOL rather than requiring explicit programming. Since the bulk of the equations used in Manufacturing Control Decision Rules are relatively simple this would allow the system designer to define the entire program without the necessity of any programming (even the relatively small amount now needed for pseudo-arithmetic.)

2. The economics of including a formula solver in TABSOL is not answered in this write-up, but only the technical feasibility of representing the arithmetic logic in a simple, usable form.

3. In general, it will be necessary to express the operations of: plus, minus, multiply, divide, square root and integer exponents (square, cube). These are called operators.

4. There are two types of operands required: literal, where the actual value itself is expressed; and non-literal where the name or location of the factor is denoted.

5. We must use and recognize two "delineators": open parenthesis $\langle (\rangle$ and close parenthesis $\langle) \rangle$. This enables us to define related terms explicitly. Two additional symbols will be used to indicate the beginning $\langle @ \rangle$ and end $\langle ¢ \rangle$ of the formula.

6. Finally provision should be made for the incorporation of special formulas where required such as Sine functions, etc.

7. The balance of this paper will develop through examples a particular notation and logic which could be used to solve this problem. A technical summary of this approach is attached as exhibit A.

8. All formulas will be stated as a series of words (or blocks) in the result (or output) rows of a structure table. The first row will be the name or location of the result field. For example, R = a + b would be shown as:

R	@
-	a
-	+
-	b
-	¢

This would be a simple unconditional table stating that computed variable R equals the value of the expression (a + b).

9. To solve this simple expression the computer could use the logic expressed in the following tables combined with a reserved sequential storage area (the Q line).

In the regular TABSOL routine the following test is made:

Is 1st Result Row

a computed variable	yes	no
Go To TABLE	TABFOR (2)	normal TABSOL

(2)

type of present word	operand	operator	operand	@	¢
type of previous word in Q line	≠ operator	---	operator	---	---
store in Q line at Next Location (NL)	(present word)	(present word)	(present word)	~~(present word)~~	-
add to NL	+ 1	+ 1	+ 1	+ 1	-
add to present word location	+ 1	+ 1	+ 1	+ 1	+ 1
Go To TABLE	(2)	(2)	(3)	(2)	(7)

(3)

———	———
Inter Result	(NL - 1) operates per (NL - 2) on (NL - 3) note: NL - 2 is always an operator and the others operands.
Store at NL - 3 in Q line	Inter Result
Add to NL	- 2
Go To TABLE	(2)

(7)

———	———
Store at Result Location	1st Entry in Q line
Go To	TABSOL

10. Let's expand this to handle a more complicated problem:

R = c(a + b) rewritten as R = c*(a + b)
where * means times

———	———
R	@
-	c
-	*
-	(
-	a
-	+
-	b
-	)
-	¢

TABLE (2) would be expanded to handle this as follows:

type of present word	(	)
type of previous word in Q line	-	-
store in Q line at Next Location (NL)	(present word)	-
add to NL	+ 1	-
add to present word location	+ 1	+ 1
Go To TABLE	(2)	(5)

(5)

———	———
store in Q line at location NL - 2	(Q line at location NL - 1)
add to NL	- 1
Go To TABLE	(6)

(6)

NL	≤ 3	≥ 4	≥ 4
Type of word at NL - 2	—	operator	$\neq$ operator
Go To TABLE	(2)	(3)	(2)

Let us follow this through step by step.

-	-
R	@
-	c
-	*
-	(
-	a
-	+
-	b
-	)
-	¢

Present word location equals 1.

Step (1) (present word) = @

Q Line	NL	Present Word Location
	1	2

Step (2) (present word) = c

	Q Line	NL	Present Word Location
	c	2	3
Step (3)	c *	3	4
Step (4)	c * (	4	5
Step (5)	c * (a	5	6

R	@
-	a
-	/
-	(
-	(
-	b
-	-
-	c
-	)
-	**
-	2
-	)
-	¢

The Q Line successively looks as follows:

(1) Q Line

(2) Q Line
a

(3) Q Line
a
/

(4) Q Line
a
/
(

(5) Q Line
a
/
(
(

(6)
a
/
(
(
b

(7)
a
/
(
(
b

(8)
a
/
(
(
[b - c]

(9)
a
/
(
[b - c]

(10)
a
/
(
[b - c]
**

(11)
a
/
(
$[b - c]^2$

(12)
$a/[b - c]^2$

(13)
$a/[b - c]^2 \rightarrow R$

Exhibit A

TABFOR

Table Solution Of Formulas

B. Grad
11/5/58

Definitions

⌜	means operand
r	means operator
(	means front delineator
)	means close delineator
¢	means end of expression delineator
@	means beginning of expression delineator

All words in a result column will carry a tag. This tag will indicate the type of word that it is: operand, operator, front parenthesis, close parenthesis, beginning of expression or end of expression.

The content of an operand word in a result column will either be a literal value (fixed point noted by explicit inclusion or else assumed that the value is an integer) or a non literal which can be interpreted into an address.

The content of an operator word in a result column will designate the particular type of operator:

add or plus	by	+
subtract or minus	by	-
multiply	by	*
divided	by	/
exponent	by	**

The content of a delineator word in a result column will be blank.

T represents the tag of a result block.

T = ⌜, r, (,), ¢, @

PW indicates the kjth block of a result matrix where k is the row and j the selected column; this is called the present word. The present word may have any of the following constructions:

ϬAAAA	≡	operand, non literal, relative address may be alphanumeric
ϬNNNN	≡	operand, literal, numeric only, one location may have a decimal point
rbbb+ rbbb- rbbb* rbbb/ rbb**		five operators, add, minus, multiply, divided by, exponent
(bbbb)bbbb ¢bbbb @bbbb		four delineators
T_{PW} PW		contents of the present word

Q_{NL} represents a queue line in which all information is stored prior to arithmetic manipulation. NL denotes the next location in the queue line.

L_R represents the location where the result is to be stored.

R represents the value of the result.

Initially k is set and j is fixed at the proper column.

In the regular TABSOL routine the presence of the symbol @ will automatically transfer the program to the TABFOR subroutine.

The following are the TABFOR Tables:

(1) Go to (2)

L_R	Result variable kj location

(2)

T_{PW}	@	r	σ	(	σ	)	¢
$T_{Q_{NL}-1}$	-	-	≠ r	-	=r	-	
Q_{NL}	-	[PW]	[PW]	[PW]	[PW]	-	-
NL	1	NL+1	NL+1	NL+1	NL+1	-	-
k	k+1	k+1	k+1	k+1	k+1	k+1	k+1
Go To	(2)	(2)	(2)	(2)	(3)	(5)	(7)

(3) Operate per $Q_{NL} - 2$
on $Q_{NL} - 1$
and $Q_{NL} - 3$
} Inter result
Go to (4)

(4) Go To (2)

$Q_{NL} - 3$	Inter result
NL	NL - 2

(5) Go To (6)

$Q_{NL} - 2$	$(Q_{NL} - 1)$
NL	NL - 1

(6)

NL	≤ 3	≥ 4	≥ 4
TQ_{NL-2}	-	r	$\neq r$
Go To	(2)	(3)	(2)

(7)

R	Q_1

The simplest form is:

R = @ σ r σ ¢

This would be written:

L_R	@
-	σ
-	r
-	σ
-	¢

Initial condition: k = 1; L_R is set up.

Step 1	Q Line	NL	k	L_R
		1	2	

Step 2	Q Line	NL	k	L_R
	σ	2	3	

Step 3	Q Line	NL	k	L_R
	$\overline{\sigma}$	3	4	
	r			

Step 4(a)	Q Line	NL	k	L_R
	$\overline{\sigma}$	4	5	
	r			
	σ			

(b)

Q Line	NL	k	L_R
$\left[\overline{\sigma}r\sigma\right]$	2	5	

Step 5 (a)	Q Line	NL	k	L_R
	$\left[\overline{\sigma}r\sigma\right]$	2	6	

(b)	Q Line	NL	k	L_R
			6	$\left[\overline{\sigma}r\sigma\right]$

We will now develop a wholly general form showing each possible kind of juxtaposition;

There are six symbols $\overline{\sigma}$, r, (,), @, ¢

The possible permutations of six things taken two at a time (including permutations with itself) is 36: (x notes an invalid combination)

x$\sigma\overline{\sigma}$	r$\overline{\sigma}$	($\overline{\sigma}$	x)$\overline{\sigma}$	@$\overline{\sigma}$	x¢$\overline{\sigma}$
σr	xrr	x(r	)r	x@r	x¢r
xσ(	r(	((	x)(	@(	x¢(
σ)	xr)	x()	))	x@)	x¢)
xσ@	xr@	x(@	x)@	x@@	x¢@
σ¢	xr¢	x(¢	)¢	x@¢	x¢¢

1. Any permutation with @ last or ¢ first can be eliminated.

2. Two operands or two operators cannot be adjacent.

3. An expression or a term cannot end in an operator.

4. An open parenthesis cannot be followed by a close parenthesis or a ¢.

5. An operand cannot directly precede an (.

6. An operator cannot directly follow an (.

7. A close parenthesis cannot be followed by an open parenthesis or an operand.

8. An @ cannot be followed by a close parenthesis, a ¢ or an r.

This leaves only twelve possible combinations of two symbols. The following three symbol combinations are possible using the above rules:

σrσ	xrσr	(σr	)r-	@σr
σr(	rσ)	x(σ)	)r(	x@σ)
σ)r	rσ¢	x(σ¢	))r	x@σ¢
σ))	r(σ	((σ	)))	@(σ
σ)¢	r((	(((	))¢	@((

9. An operand cannot be both preceded and followed by an operator;

10. An operand cannot be both preceded by an open parenthesis or an @ and followed by a close parenthesis or ¢.

The following four symbol combinations are possible using the above rules:

				@σrσ
	r σ)r	(σrσ	)rσ)	@σr(
σrσ)	rσ))	(σr(	)rσ¢	@(σr
σrσ¢	rσ)¢	((σr	)r(σ	@((σ
σr(σ	r(σr	(((σ	)r((	@(((
σr((	r((σ	((((	))rσ	
σ)rσ	r(((		))r(	
σ)r(			)))r	
σ))r			))))	
σ)))			)))¢	
σ))¢				

The following five symbol combinations are possible using the above rules.

x (σrσ¢
x @σrσ)

11. There must be a close parenthesis between an open parenthesis and a ¢.

12. There must be an open parenthesis between a close parenthesis and a @.

13. There must be an equal number of open and close parentheses in an expression.

14. There must be only one @ and one ¢ in an expression and they must respectively be in the first and last positions. No other symbol may be in either the first or last position.

15. A close parenthesis cannot occur in an expression unless an open parenthesis has previously occurred.

16. An open parenthesis cannot occur in an expression unless a close parenthesis subsequently occurs.

Under the above rules this complete expression is possible in five symbols:

@ σ r σ ¢

The following nine symbol expressions are possible:

@ σ r (σ r σ) ¢
@ (σ r σ) r σ ¢

The following twelve symbol expression is possible:

@ (σ r σ) r (σ r σ) ¢

The following thirteen symbol expression is possible:

@ (σ r (σ r σ)) r σ ¢

TECHNIQUES

An Algebraic Translator

H. Kanner, Control Data Corporation, Minneapolis, Minn.

1. Introduction

This paper presents an algebraic translator bearing some similarities to the one described by Wegstein.[1] The scheme differs from Wegstein's primarily in the following respects:

(1) A four-level hierarchy is established so that the operations of evaluating a function of one variable, exponentiation, multiplication (division), and addition (subtraction) are performed in the stated order of precedence except where otherwise dictated by parentheses.

(2) An efficient assignment is made of symbols corresponding to temporary storage locations for intermediate results in the object program.

(3) A correct interpretation is made of a plus or minus sign following the symbols (or =.

As in Wegstein, variables are formally represented by single characters, and no provision is made for the treatment of numerical constants. The elimination of these restrictions will be discussed briefly in section 4.

2. Flowchart Notation

χ_i denotes the ith symbol in the string to be compiled. ξ is a generic symbol for those χ_i's representing variables. α is a generic symbol for those χ_i's representing functions of one variable, e.g., $\sqrt{}$, sin. It will be assumed that each such variable is represented by a one-character symbol.

Since the compiler performs a translation between the respective languages of subject and object programs, dealing as a consequence with the alphabets of both languages, care must be taken to provide a notation for describing the compiler which avoids any confusion between symbolic variables and symbolic locations within the compiler itself. The following conventions apply:

"$A' \rightarrow \eta$" is read: The symbol A is to be stored in location η.

$\alpha \rightarrow \eta$ is read: The contents of location α are to be stored in location η.

3. Functional Description

The input strings are written in conventional algebraic notation except for two special symbols: ⊢ and ⊤. The termination of the string is indicated by ⊢, and ⊤ is the exponentiation operator, the usage being A ⊤ B for A^B. The output of the translator is a set of ordered quadruples to be interpreted as instructions in a three-address code.

The elements of an output quadruple are respectively: the antecedent operand, the operator, the consequent operand, and the symbolic location of the result. The null symbol Φ is used as the first element in a quadruple representing a singulary operation and as the first two elements in a quadruple corresponding to a "fetch and store" operation. To illustrate, the input string Y = A × B + C / D ⊢ would produce the output:

$$(A, \times, B, T_1)$$
$$(C, /, D, T_2)$$
$$(T_1, +, T_2, T_1)$$
$$(\Phi, \Phi, T_1, Y)$$

In a more conventional notation, the first of these output quadruples would be written $A \times B \rightarrow T_1$, and the last would be written $T_1 \rightarrow Y$. The symbols such as A, B, T_1, etc. appearing in the output are to be interpreted by a subsequent assembly program as symbolic addresses. The locations associated with T_1 and T_2 are for the storage of intermediate results during computation.

4. Modus Operandi

The algebraic statement to be translated is read once only, from left to right. Symbols representing operators and operands are stored in a prescribed format as they are read; this process is termed "recording." At certain points, recording is interrupted and the material already recorded is wholly or partially converted to output statements, i.e., compiled. The stored items are compiled in the reverse order to that in which they were recorded, so that the last item recorded becomes the first item to be compiled. The processes of recording and compiling alternate until the entire statement has been read, the final step being one of compiling. As soon as explicit definitions of recording and compiling have been given, the algorithm for algebraic translation can be stated as a set of rules for determining when to switch from recording to compiling, and vice versa.

[1] J. H. Wegstein, From formulas to computer oriented language, *Comm. Assoc. Comp. Mach.* 3 (1960), 6.

The appropriate symbols are recorded in a set of storage locations labelled δ_i, α_i, μ_i, π_i, and η_i, $i \geq 1$. The symbol triple corresponding to a binary operation is stored as follows: the operator in π_i, the antecedent operand in μ_i, and the consequent operand in η_i. A singulary operation is put in the same form by storing Φ, the symbol for a null operand, in location μ_i (22).[1] The value of the index i is always equal to the number of triples recorded but not yet compiled, and is increased by unity immediately prior to recording an operator (32). At the time an operator is recorded in π_i (36), the current temporary storage symbol, T_i, replaces the operand in η_{i-1} (34). An operand is always recorded in both η_i and μ_{i+1} (6). As an illustration of this procedure, the steps in recording the sequence $A + B \times C$ are: 'A' $\rightarrow \mu_1$, '+' $\rightarrow \pi_1$, 'T_1' $\rightarrow \eta_1$, 'B' $\rightarrow \eta_2$, 'B' $\rightarrow \mu_2$, '×' $\rightarrow \pi_2$, 'T_1' $\rightarrow \eta_2$, 'C' $\rightarrow \eta_3$, 'C' $\rightarrow \mu_3$, yielding the array shown in table I. The symbols =, (,), and ⊢ exert control functions and are never recorded as such.

Compilation is defined to be the production of a sequence of output quadruples of the form

$$([\alpha_i], [\pi_i], [\eta_i], [\eta_{i-1}]),$$

where $[\alpha]$ denotes the symbol stored in location α. The value of i is decreased by unity after production of each such statement. Thus, the recorded information in table I, if compiled completely, would yield: (B, ×, C, T_1), (A, +, T_1, T_1) (45).

Following compilation, the step $\eta_i \rightarrow \mu_{i+1}$ is performed (48), where $i + 1$ is the index of the last triple compiled. This provides a proper antecedent for the first triple of the next sequence to be recorded. Thus, after the compilation of the sequence in table I, the step 'T_1' $\rightarrow \mu_1$ would be performed, and T_2 would be used to indicate subsequent temporary storage.

The formation of a set of output quadruples that truly represents the operational sequence given by the input string is accomplished by making the transitions between recording and compiling at the proper points. As a tool for accomplishing this purpose, a set of indices λ, termed hierarchy numbers, are assigned to the various operators according to the schedule shown in table II. The hierarchy number describes the precedence of the corresponding operation; the one with the higher number is to be performed first.

The rules for starting and terminating the steps of recording and compiling can now be stated. The case wherein the expression to be analyzed has no parentheses will be considered first. During recording, the hierarchy number of the current operator is compared to that of the operator just previously recorded. If the current number is higher, the item is recorded; if lower, compilation starts. Special treatment is given to the case of two or more successive operators bearing the same hierarchy number. If the number is 1 or 2, compilation of the previously recorded material is started. If the number is 3 or 4, the current operator is recorded (30, 31). This complication is needed to give the indicated interpretations to strings of the following types:

A − B − C to be interpreted as if written (A − B) − C
A ↑ B × C to be interpreted as if written (A ↑ B) × C
A ↑ B ↑ C to be interpreted as if written A ↑ (B ↑ C)
√√A to be interpreted as if written √(√A).

Compilation is also triggered by the symbol ⊢ (27, 28), and proceeds without interruption until all outstanding recorded triples have been compiled (40).

A similar set of rules governs in part the cessation of compiling. The hierarchy number of the operator whose appearance triggered compilation is compared to that of the operator in the triple about to be compiled. If the latter number exceeds the former, the triple under consideration is compiled. If the reverse is true, the compilation ends, and the operator that has been held in abeyance is recorded. As before, in case of a tie the decision depends on the value of the tied hierarchy number, compilation proceeding if the number is less than three (43, 44).

The following modifications of and additions to the rules provide for the treatment of parenthesized expressions:

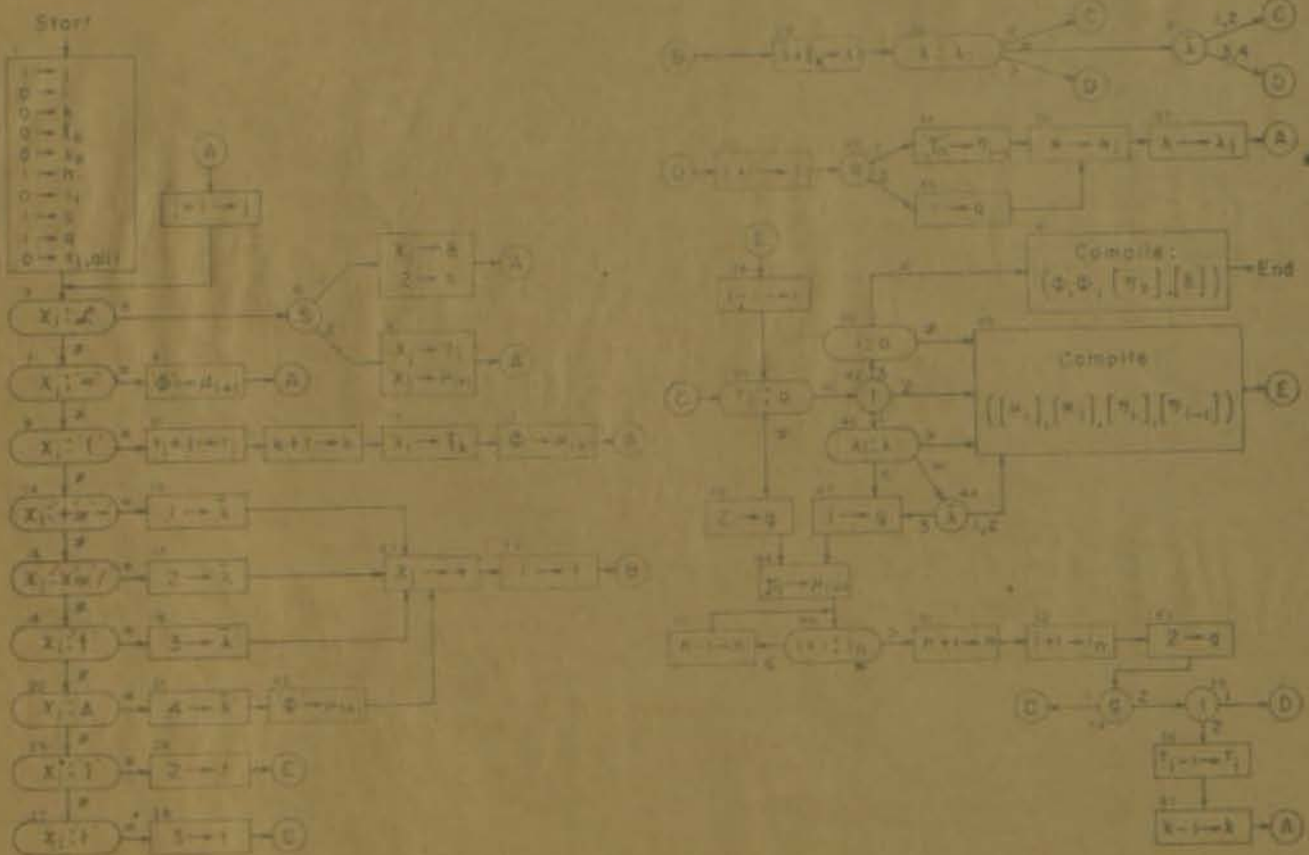
(1) The previously given rules apply with the substitution of the phrase "level number" for "hierarchy number".
(2) The level number of a left-hand parenthesis is defined to be that of the nearest operator to its left unless no such operator exists, in which case it is defined to be zero (1, 12).
(3) The level number of an operator is defined to be the sum of its hierarchy number and the level number of the nearest uncancelled left-hand parenthesis to its left (29).
(4) Compilation is always terminated by an uncancelled left-hand parenthesis (39).
(5) Compilation is always triggered by a right-hand parenthesis (25, 26), and at the end of such compilation, the terminating left-hand parenthesis is cancelled (36, 37).

The following example illustrates the application of these rules by comparing the compilation of an equation without parentheses to that of an equation with the same structure, in which each variable in the former has been replaced by a parenthesized expression.

Table I

Location	Content
η_1	T_1
μ_1	A
π_1	+
η_1	T_1
μ_2	B
π_2	×
η_2	C

[1] Numbers in parentheses refer to boxes of the flow chart shown in figure 1.


Start
Compile:
End
Compile

As a further guide to the flow chart, it should be noted that the connector A re-enters the program with the reading of the next symbol, the connector B enters a subprogram which determines whether recording or compiling is to follow, the connector D enters a recording subprogram, and the connector C enters a compiling subprogram. If compilation has been started by the appearance of an operator ($\lambda = 1$), then compilation continues until a left-hand parenthesis is reached (39), or the proper λ_i value is encountered (43, 44). The operator which caused the compilation is then recorded by switching to connector D (54 or 55). The switch q is always set to 2 after compilation to prevent an improper assignment of temporary storage at the time of recording the next operator (33).

The sub-program 49, 50, 51, 52 provides the proper assignment of temporary storage symbols. The contents of q_i will always be some T_s at the time box 48 is entered. The effect of the sub-program is equivalent to the act of inspecting T_s for its s-value, and increasing s by unity.

4. Extensions

Several obvious extensions of this scheme would be required in order to make it suitable for inclusion in a working compiler. The 3-address format of the compiled program should be considered a canonical form requiring further processing to adapt it to the syntactic structure of any particular computer. In this additional processing, attention should be given to optimum usage of the computer's register arrangement.

It would be highly desirable to make provision for the treatment of multi-character symbols and numerical constants. Probably the most satisfactory means for implementation would be to store symbols and constants in a symbol table similar to that of an assembly program, and to use the relative symbol table addresses in lieu of the symbols during internal manipulation. The original symbols could then be re-inserted at the point of compilation, which would increase the legibility of the compiled but unassembled program, an important consideration if error correction is to be attempted at that level.

Provision should also be made for some degree of "goof control" to warn of the appearance of erroneous symbol strings, e.g.,

$$Y = (/B + C) \vdash \quad \text{or} \quad Y = (\lambda A + B) \div C) \vdash$$

I thank Mr. Herbert Gray of the Argonne National Laboratory for suggesting several simplifications of the original version of the flow chart.

SALE, a Simple Algebraic Language for Engineers

W. R. Breitenham, K. Clark, G. Kuss, and H. Thompson, *A. O. Smith Corp., Milwaukee, Wis.*

A great deal of attention has been given, and rightly so, to the problem of solving large problems economically on digital computing equipment. In order to cope with larger and larger problems, computing machines have become larger, faster, and more expensive. Unfortunately, this also has had the effect of increasing the size of the smallest problem which can be economically run on the equipment.

R. W. Hamming [1] and R. W. Bemer [2] have suggested that this trend can be reversed by providing users with special personal input-output equipment remotely connected to a large, centrally located digital computer. When the scientist needs the assistance of the computer in some phase of his work, he keys in his problem, properly stated. The machine then sandwiches the requested computation into its schedule, reporting the results on the remote output equipment.

Although such specialized equipment is not now available, it nevertheless appears that something along these lines can be realized. Our approach has not entailed the development of any special hardware; present computing equipment is used just as it is. Instead we have developed a specialized automatic coding system, aimed at satisfying the following requirements:

1. It must be easy to learn and use, with a minimum of language complications.
2. The computer time required to convert the problem into machine language must be minimal.
3. Set-up and tear-down time on the computer must be close to zero.
4. Printed results should be available to the user immediately.
5. The overall procedure should be efficient for the handling of small problems, but not necessarily for large

TABFOR

arithmetic operations

2 field instruction

xx	xxx xxx	xx±
OP. code	Field A	Field B

6 digit const
3 digit addr. 2 digit addr + sign

Preclear	Store	Store + Clear

OP CODE : +
−
×
/
SPECIAL - BINARY
" UNARY

Combinations: ×, + ; ×, − } use fixed accum
/, + ; /, − }

Field A const, var (accum)

Field B var (accum)

OP Code	
x4	+
x5	−
x6	×
x7	/
x8	×, +
x9	/, +

OP Code	Field A	Field B
3x	const	var
4x	var	var
5x	const	var
6x	var	var
7x		
8x		

90-99 : Special operations

199 - Accum address

34 A + (B) → Accum
44 (A) + (B) → Accum (Effectively Pre-Clear Accum)
54 A + (Accum) → B, Accum
64 (A) + (Accum) → B, Accum

✓ A + (B) → Accum, B (Effectively 0 → Accum then →)

✓ ✓ (A) + (B) → Accum, B (" " " ")

✓ A + (B) → B; 0 → Accum

✓ ✓ (A) + (B) → B; 0 → Accum

✓ A + (Accum) → B; 0 → Accum

✓ ✓ (A) + (Accum) → B; 0 → Accum

✓ A + (Accum) → Accum

✓ ✓ (A) + (Accum) → Accum

~~If no accum~~

A + (B) → B
(A) + (B) → B

could possibly go to

$(A_{11}) + (A_{12}) \rightarrow B$

$Y1 = (X1 * X1) +$

$$Y_1 = (x_1^2 + 3x_1x_2 - x_1x_2^2 + 4)/(x_1 + x_2)$$

$Y1 = ((X1*X1) + ((3*X1)*X2) - (X1*(X2*X2)) + 4)/$

ASSGV	X1	T1
MULTV	T1	T1
ASSGV	X1	T2
MULTC	3	T2
MULTV	X2	T2
ADDV	T2	T1
ASSGV	X1	T3
MULTV	X2	T3
MULTV	X2	T3
SUBV	T3	T1
ADDC	4	T1
ASSGV	X1	T4
ADDV	X2	T4
DIVV	T4	T1
ASSGV	T1	Y1

OP			
~~MULTV~~	X1	X1	(X1)(X1) → Acc
~~ASSGV~~	Accum	T1	Acc → T1
~~MULTC~~	3	X1	0 → Acc, 3X1 → Acc
~~MULTV~~	Accum	X2	3X1X2 → Acc
ADDV	Accum	T1	3X1X2+(X1)(X1) → Acc
MULT	X1	X1	
ASSG		T1	
MULT	3	X1	
MULTS	X1	T2	
MULT	X1	X2	
MULTS	X2	T3	
ADD	T1	T2	
SUB	T3		
ADDS	4	T4	
ADD	X1	X2	
DIVS	T4	Y1	

$Y1 = X1 + 4X2$

ASSGV	X1	T1
ASSGV	X2	T2
MULTC	4	T2
ADDV	T1	T2
ASSGV	T2	Y1

or

ASSG	X2	T1
MULT	4	T1
ADD	X1	T1
ASSG	T1	Y1

MULT	4	X2
ADDS	X1	Y1

Accum	OPER 1	OP	OPER 2	Accum	Loc A	Loc B
0	A	+	B	0	A	A+B
0	A	+	B	A+B	A	A+B
0	A	+	B	A+B	A	B
( )	Acc	+	A	0	A	Acc+A
( )	Acc	+	A	Acc+A	A	Acc+A
( )	Acc	+	A	Acc+A	A	—

(.) $(A \times B) + Acc \rightarrow Acc$

Inventory Control Decisions

whether to stock an item

Reorder Point Quant

Order Quantity

Review Interval

Service Level

Factors

Demand Forecast

Lead Time Forecast

Forecast Error

Cost of Carrying Inv

~~Cost of set preparation~~

Cost of Work-out

Cost of analysis (review)
[ordering costs]

Setup Cost (discounts)

Limiting Considerations

investment

Production Smoothing

Space

Competitive performance

Model changeover, Engrg design change