Modeling Setup Time Minimizing Scheduling Rules in SIMAN

Walter Schwaiger

Forschungsbericht/Research Memorandum No.284

October 1991

Contents

Zusammenfassung

In diesem Forschungsbericht werden drei von der Modellstruktur her gänzlich verschiedene, hinsichtlich des Resultats - also der erzielten durchschnittlichen Umrüstzeit - jedoch völlig gleichwertige Möglichkeiten präsentiert, in der Simulationssprache SIMAN die Scheduling-Regel "Shortest Setup Time" (kürzeste Umrüstzeit) zu modellieren. Des weiteren wird eine "verfeinerte" Heuristik zur Minimierung von Umrüstzeiten (Shortest Relative Setup Time; Kürzeste Relative Umrüstzeit) vorgestellt und eine Möglichkeit zu deren Realisierung in SIMAN gezeigt.

Summary

In this paper three SIMAN-modeling-alternatives for the scheduling rule "Shortest Setup Time First" are presented. Although the three models are completely different with regard to the programming concept, they yield exactly the same result, i. e. the same average setup time. Furthermore a more sophisticated heuristic ("Shortest Relative Setup Time") is introduced, and its realization in SIMAN is also shown.

The Objectives of the Paper

The topic of this paper is the modeling of setup time-minimizing heuristics in SIMAN. The focus is not on the comparison of the performance of different scheduling rules with regard to the average changeover time. Also it is not the objective to form an empirically based answer to the question which of the tested heuristics is the best.

On the contrary, **the main issue of the paper is the description of three basically different modeling techniques for the simulation of the "Shortest Setup Time First"(SSF)-heuristic.**[1]

Modeling this heuristic is much more difficult than modeling the SPT-rule (shortest processing time), for example. For a certain type of workpiece processing time is represented by an attribute of constant value. Whenever a new job[2] arrives, it can be easily assigned its rank in the queue.

Setup time, however, does not depend only on the type of job, that will be selected next, but also on the type of the preceding workpiece, i.e. the type of the job that has just left the machine. Generally speaking changeover time will be

- zero, if the types of the preceding and the following job ("successor") are the same,
- non-zero, if the types of "predecessor" and "successor" differ.

---

[1]) All models that will be presented make use of SIMAN commands only. The possibility to apply a "user-written-subroutine" (C or Fortran) was not to be considered.

[2]) For our topic the words "workpiece" and "job" may be considered as synonyms. "Changeover time" is used as synonym for "setup time".

Therefore it is necessary to "inform" the system about the type of the workpiece currently being processed. **Only in the moment when this "predecessor" leaves the machine, the current setup times may be assigned** to the waiting workpieces, and the corresponding ranking can be effected.

Towards the end of the paper a "refinement" of the SSF-Rule is presented (**Shortest Relative Setup Time**), and its realization in SIMAN is also shown. Finally the performance of both heuristics is compared with the result of an extremely simple rule ("Same Type First").[3]

General Characteristics of the Given Scheduling Problem

(see, e.g., Kusiak 1990, Chapter 13.3: Scheduling a Flexible Forging Machine)

- The workpieces (=jobs) arrive one by one at a jobshop consisting of a single flexible machine.
- At the beginning the queue is empty.
- The jobs can be classified into various types.
- The processing time depends upon the jobtype.
- Whenever a changeover from one type of workpiece to another occurs, the machine has to be set up. The corresponding setup time depends on the respective predecessor/successor-combination.
- Minimizing the downtime of the machine due to setting up represents the exclusive objective of scheduling.

---

[3]) All source codes and summary reports can be looked up in the appendix.

The Experimental Design

Within the scope of the general situation described above the experimental design is specified as follows:

- The interval between job-arrivals is regular and amounts to 7 minutes.

- The assignment of jobtypes and corresponding processing times is determined by a discrete distribution, the parameters being listed in the table below:

	probab.	processt.		probab.	processt
type 1	0.4	5	type 6	0.04	8
type 2	0.2	8	type 7	0.04	6.5
type 3	0.1	12	type 8	0.04	5.5
type 4	0.05	5	type 9	0.04	3
type 5	0.05	9	type 10	0.04	7.5

If the jobs were selected according to the FIFO-principle, the average processing time (weighted by probabilities) would amount to 6.72 minutes.

- The setup times are specified by the following matrix:

type pred.	type of successor									
1	0	1	3	5	7	9	11	13	15	17:
2	2	0	4	6	8	10	12	14	16	18:
3	17	15	0	13	11	9	7	5	3	1:
4	18	16	14	0	12	10	8	6	4	2:
5	1	3	5	7	0	17	15	13	11	9:
6	2	4	6	8	18	0	16	14	12	10:
7	9	11	13	15	17	7	0	5	3	1:
8	10	12	14	16	18	8	6	0	4	2:
9	7	5	3	1	9	11	13	15	0	17:
10	8	6	4	2	10	12	14	16	18	0:
Start	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5

As can be seen, the matrix is assymmetric. This implies, for example, that changing over from jobtype 1 to jobtype 2 does

not cause the same setup time as vice versa.

In the FIFO-case the average setup time (weighted by the probabilities of the respective predecessor/successor-combinations) would come to 6.12 minutes.

- The length of the simulation-run is 10.000 minutes.

If one takes into account that the average processing time is only slightly lower than the interval between job arrivals, it becomes evident that **only a good scheduling in regard to setup times can prevent the queue from growing longer and longer and avoid the collapse of the system.**

Common Features of the Three SSF-Modeling-Techniques

All modeling variants which will be presented below ascertain the type of the workpiece having just been processed by means of a variable named "currjob", to which the value of the attribute "jobtype" is assigned immediately before the machine is released.

Further the matrix of setup times is stored up in a TABLES-element in the experimental file. The assignment of the current setup times is realized by means of a "table-function". This function can "seize" a certain value from the matrix using the type of the preceding (variable "currjob") and of the following (attribute "jobtype") workpiece as parameters defining row and column.

The three modeling-techniques are, however, completely different in respect to the method of finding out which one of the waiting jobs (or jobtypes, respectively) incurs the shortest setup time.

Model "Seize-Priority" (Rule: SSF)

This modeling-variant turned out to be the most simple one: There is a separate queue for each jobtype. These queues can be represented by a single station-submodel, and every job is guided to the right queue by means of the special-purpose-attribute "ns".

Each queue is followed by a SEIZE-block. Using the table-function "tf" each SEIZE-block is assigned a priority depending on the current setup time for the jobs waiting in the respective queue. If the setup time matrix contains integers only, these can be applied directly as priority-levels. Otherwise a priority matrix has to be derived from the setup time matrix. These priorities also have to be stored up in the TABLES-Element.

If the queue, which has first priority, is empty, the next job will be selected from the queue with second priority, and so on.

The advantages of this modeling-technique are its simplicity, clearness and low consumption of computing-time. The necessity of a separate matrix of priorities can be considered as drawback.

Model "Searching Algorithm" (Rule: SSF)

For this technique, too, a priority matrix is needed. After the completion of a job a systematic search in the priority matrix starts. The value of "currjob" (type of the workpiece just completed) determines the row in which the search has to take place. Now this row is tested column by column, until the first priority (=jobtype with lowest setup time) is found.

In its next step the searching algorithm checks if there is a job of the desired type in the queue. If there is such a workpiece, it will be removed from the queue and brought to the machine. If not, the searching in the priority matrix has to start again, this time with the objective to find priority 2.

The disadvantage of this modeling-variant is again the necessity to derive a priority matrix. Considering the computing time it is remarkably more "expensive" than "Seize-Priority".

Model "Queue-Tandem" (Rule: SSF)

This model selects the next job by a LVF (Lowest Value First)-

specification in the QUEUES-Element. At first all unassigned jobs have to wait in "Mainqueue", in this case without special ranking. As soon as the predecessor is completed, it sends a signal which allows the waiting workpieces to leave Mainqueue. They are assigned their current setup times and enter a queue named "Springboard". There they are ranked LVF(Setup time), and the job being ranked on the first position seizes the machine. The remaining jobs in Springboard have to be sent back to Mainqueue.

The great advantage of this model is that it does not require a priority matrix. Its disadvantage is the extremely high consumption of computing time, which is caused by the fact that the workpieces not selected have to be moved back to the mainqueue one by one. The necessity of some dummy-delays which enforce the change of activity-status from one entity to another could be considered as less elegant.

Rule:"Shortest Relative Setup T. First" (by Model "Queue Tandem")

The concept of "relative setup time" is defined as (Setup time)/(Number of jobs of this type in the queue)[4]. The idea of this heuristic can be explained easily: it may be disadvantageous to select a job, if only few workpieces of the same kind are already waiting in the queue - although the respective job incurs a relatively short setup time. It might be better to select a job with a higher setup time if the number of unassigned workpieces of the same type is great.

In other words: One "buys" a series of job-changes without any setup time at the "price" of the setup time for the first job of this group.

The modeling of this rule can be realized by an extension of the "Queue-Tandem"-model: The arriving workpieces and the completed ones have to be counted by jobtype. Now the denominator of the

---

[4]) This heuristic has been "invented" by the author of this paper.

ranking criterion above can be calculated by the formula "Value of the counter at the entrance minus value of counter at the exit".

The respective value of the ranking criterion has to be assigned to an additional attribute, and the LVF-Rule for queue "Springboard" has to be modified.

Comparing the Results with Rule:"Same Type First"

The most simple heuristic reducing setup times would be "Same Type First": When a job is finished, a workpiece of the same type is selected as successor. If there is no such job in the queue, the FIFO-principle takes effect, without taking into account the setup times. The simulation of this heuristic can be realized easily by a simple modification of the "Searching Algorithm".

The three modeling-variants of SSF all yield exactly the same result: The average setup time amounts to 0.76 minutes. The comparison with the performance of "Same Type First" was rather sobering: The result of this extremely simple rule was even slightly better than that of the "higher sophisticated" SSF-heuristic: 0.71 min.

The "Shortest Relative Setup time First"-heuristic, however, reduces the average-setup time down to 0.48 minutes!

Further Aspects of Interest (not Covered in this Paper)

- Certainly the presented modeling-techniques could be improved. Especially for the "Queue-Tandem"-variant a reduction of calculation time should be possible.

- In order to be virtually able to assess the performance of the presented heuristics, they would have to be tested in the context of different experimental designs.

- The dependence of the performance of different rules on

several characteristics of the experimental design could be investigated by an analysis of variance.

- Other interesting rules could be modeled, e.g. a heuristic that does not only take into account the setup time a job causes when seizing the machine, but also the setup time that will probably occur when the next changeover to another jobtype takes place.

  Another sensible method would be to transform the matrix of setup times by subtracting in each column the lowest non-zero value from all other setup times in this column. Such a procedure could avoid the unavailing deferring of jobs that require relatively high changeover times in any case.

Appendix

SEIZE-PRIORITY

```
BEGIN,          no,seize_prior;
;     This is the model "Seize Priority" with:
;      - a separate queue for each jobtype
;      - representation of all queues by 1 generic station-submodel
;      - selection of next job according to the priority currently
;        assigned to the seize-blocks;

          CREATE:     7;                      !interval between arrival of jobs
          ASSIGN:
                      jobtype=DP(1,2):        !assigning jobtypes and processing-
                      processtime=dp(jobtype+2   !times
                      ):
                      ns=jobtype;             !assigning sequence
          ROUTE:      ;                       !directing to the jobtype's queue
type      STATION,    type1-type10;
          QUEUE,      m;                      !representing all queues
          SEIZE,      tf(m+11,currjob):       !priority of queue
                      forg_mach;
          ROUTE;                              !directing selected job to machine
setup_proc STATION,   setup_proc;
ass setup ASSIGN:
                      setuptime=tf(currjob,   !assigning setuptime
                      jobtype);
          DELAY:      setuptime:              !setting up
                      MARK(arrsetup);
          TALLY:      setup_time,int(arrsetup  !measuring setuptime
                      );
          DELAY:      processtime;            !processing

          ASSIGN:     currjob=jobtype;        !assigning value to "currjob"
;                                             !corresponding to type of the
;                                             !finished job
          RELEASE:    forg_mach:
                      DISPOSE;
END;
```

```
BEGIN,          no;
;   this is the exp-file for "Seize_priority";
PROJECT,        seize_prior,walter;
ATTRIBUTES:     arrsetup:
                jobtype:
                processtime:
                setuptime;
VARIABLES:      currjob,11;
RESOURCES:
                forg_mach;
QUEUES:
                1:
                2:
                3:
                4:
                5:
                6:
                7:
                8:
```

```
                9:
                10;
PARAMETERS:     1,0.4,1,0.6,2,0.7,3,0.75,4,      !distribution of jobtypes
                0.80,5,0.84,6,0.88,7,0.92,8,
                0.96,9,1.0,10:
                3,1.0,5:                         !processingtimes
                4,1.0,8:
                5,1.0,12:
                6,1.0,5:
                7,1.0,9:
                8,1.0,8:
                9,1.0,6.5:
                10,1.0,5.5:
                11,1.0,3:
                12,1.0,7.5;
STATIONS:       type1:
                type2:
                type3:
                type4:
                type5:
                type6:
                type7:
                type8:
                type9:
                type10:
                setup_proc;
SEQUENCES:      1, type1&
                   setup_proc:
                2, type2&
                   setup_proc:
                3, type3&
                   setup_proc:
                4, type4&
                   setup_proc:
                5, type5&
                   setup_proc:
                6, type6&
                   setup_proc:
                7, type7&
                   setup_proc:
                8, type8&
                   setup_proc:
                9, type9&
                   setup_proc:
                10, type10&
                    setup_proc;

TABLES:         1,1,, 0, 1, 3, 5, 7, 9,11,13,15,17:      !setuptimes
                2,1,, 2, 0, 4, 6, 8,10,12,14,16,18:
                3,1,,17,15, 0,13,11, 9, 7, 5, 3, 1:
                4,1,,18,16,14, 0,12,10, 8, 6, 4, 2:
                5,1,, 1, 3, 5, 7, 0,17,15,13,11, 9:
                6,1,, 2, 4, 6, 8,18, 0,16,14,12,10:
                7,1,, 9,11,13,15,17, 7, 0, 5, 3, 1:
                8,1,,10,12,14,16,18, 8, 6, 0, 4, 2:
                9,1,, 7, 5, 3, 1, 9,11,13,15, 0,17:
                10,1,,8, 6, 4, 2,10,12,14,16,18, 0:
                11,1,,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5:
```

```
                12,1,,1,  2, 10,10, 2, 2, 6, 6, 5, 5, 1:  !priorities
                13,1,,2,  1,  9, 9, 3, 3, 7, 7, 4, 4, 2:
                14,1,,3,  3,  1, 8, 4, 4, 8, 8, 3, 3, 3:
                15,1,,4,  4,  8, 1, 5, 5, 9, 9, 2, 2, 4:
                16,1,,5,  5,  7, 7, 1,10,10,10, 6, 6, 5:
                17,1,,6,  6,  6, 6,10, 1, 5, 5, 7, 7, 6:
                18,1,,7,  7,  5, 5, 9, 9, 1, 4, 8, 8, 7:
                19,1,,8,  8,  4, 4, 8, 8, 4, 1, 9, 9, 8:
                20,1,,9,  9,  3, 3, 7, 7, 3, 3, 1,10, 9:
                21,1,,10,10,  2, 2, 6, 6, 2, 2,10, 1,10;
TALLIES:        setup_time;
REPLICATE,      ,,10000;
DSTATS:         1,nq(1):                    !queuelength
                2,nq(2):
                3,nq(3):
                4,nq(4):
                5,nq(5):
                6,nq(6):
                7,nq(7):
                8,nq(8):
                9,nq(9):
                10,nq(10);
END;
```

→

Summary "Seize-Priority"

Replication ended at time : 10000.0

TALLY VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Observations
setup_time	.76134	2.1647	.00000	9.0000	1345

DISCRETE-CHANGE VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Final Value
nq(1)	3.9774	.90172	.00000	17.000	3.0000
nq(2)	2.3445	1.0648	.00000	11.000	1.0000
nq(3)	1.9119	1.0027	.00000	10.000	2.0000
nq(4)	1.1762	1.2619	.00000	8.0000	3.0000
nq(5)	29.084	.72834	.00000	73.000	73.000
nq(6)	1.7137	1.0894	.00000	10.000	.00000
nq(7)	1.7639	1.0080	.00000	7.0000	1.0000
nq(8)	1.0445	.99589	.00000	4.0000	.00000
nq(9)	.62620	1.3218	.00000	3.0000	1.0000
nq(10)	.40765	1.5423	.00000	3.0000	.00000

Run Time: 0 min(s) 26 sec(s)

SEARCHING ALGORITHM

```
BEGIN,          no,sear_alg;
;    This is model "sear_alg" with:
;       - the same experimental design as seize_prior;
;       - only one queue
;       - a branch/wait-combination replacing the seize-block
;       - a searching-algorithm selecting from the queue
;         the job with lowest setuptime

          CREATE:     7;                        !interval between job-arrivals
          ASSIGN:
                      jobtype=DP(1,2):          !assigning jobtype and pro-
                      processtime=dp(jobtype+2  !cessingtime
                      );
          BRANCH,     1:                            !if machine is free and
                      if,nq(buffer)==0.and.busy==0, !queue is empty -> go to
                      ass_setup:                    !machine directly, avoi-
                      else,buffer;                  !ding the queue
buffer    QUEUE,      buffer;
          WAIT:       dummy;
ass_setup ASSIGN:                               !assigning setuptime
                      setuptime=tf(currjob,
                      jobtype):
                      busy=1;
          DELAY:      setuptime:                !setting up
                      MARK(arrsetup);
          TALLY:      setuptime,int(arrsetup    !measuring setuptime
                      );
          DELAY:      processtime;              !processing
          ASSIGN:                         !assigning value to "currjob",cor-
                      currjob=jobtype;    !responding to type of finished job
          BRANCH,     1:                  !start searching-algorithm only if
                      if,nq(1)>0,continue:!queue is not empty
                      else,exyt;
continue  ASSIGN:     x(4)=1:                   !searching for priority 1
                      X(2)=1;                   !starting search with type 1
sear      ASSIGN:     x(3)=tf(x(2)+11,currjob); !determining priority
          BRANCH,     1:
                      if,x(3)==x(4),foundtype:  !type x(2) has desired priority
                                                !->search for type x(2) in queue
                      else,newsear;             !type x(2) has not desired prior
newsear   ASSIGN:     x(2)=x(2)+1:              !->try with x(2)+1
                      NEXT(sear);
foundtype SEARCH,     buffer:
                      jobtype==x(2);
          BRANCH,     1:
                      if,j>0,nextjob:           !searched job found
                      else,2ndprior;            !searched jobtype not in queue
nextjob   REMOVE:     j,buffer,ass_setup:       !direct found job to machine
                      NEXT(exyt);
2ndprior  ASSIGN:
                      x(4)=x(4)+1:              !try with next priority
                      x(2)=1:
                      NEXT(sear);
exyt      ASSIGN:     busy=0:                   !releasing the machine
                      DISPOSE;
END;
```

```
BEGIN,          no;
;this is the exp-file for "sear_alg";
PROJECT,        such1,walter;
ATTRIBUTES:     arrsetup:
                jobtype:
                processtime:
                setuptime:
                dummy;
VARIABLES:      currjob,11:
                2:
                3:
                4:
                busy;
QUEUES:         buffer;

PARAMETERS:     (same as seize-priority)

TABLES: (same as seize-priority)
TALLIES:        setuptime;
DSTATS:          1,nq(1);                        !queuelength
REPLICATE,      ,,10000;
END;
→
```

Summary for "Searching Algorithm"

Replication ended at time : 10000.0

TALLY VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Observations
setuptime	.76134	2.1647	.00000	9.0000	1345

DISCRETE-CHANGE VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Final Value
nq(1)	44.050	.48071	.00000	84.000	84.000

Run Time: 1 min(s) 27 sec(s)

QUEUE-TANDEM

```
BEGIN,          no,qu_tand;
;this is the model "qu_tand". Its principal characteristics are:
;       - a combination of two queues (mainqueue and spring_board)
;       - use of the ranking-rule LVF within queue "spring_board"
;       - the necessity of dummydelays
;       - no need for a priority-matrix

          CREATE:     7;
signal1   SIGNAL:     1;
          ASSIGN:
                      dummy=2:
                      jobtype=DP(1,2):          !assigning jobtype and processt.
                      processtime=dp(jobtype+2
                      );
mainqu    QUEUE,      mainqueue;
          WAIT:       dummy;                    !waiting for signal2 (from
;                                               !finished job)
          ASSIGN:
                      setuptime=tf(currjob,     !assigning setuptimes
                      jobtype);
          QUEUE,      spring_board;             !queue with setuptime-ranking
          SEIZE:      forg_mach:                !job with lowest setuptime
                      NEXT(if_1);               !seizes machine
remove    REMOVE:     1,spring_board,mainqu;       !remaining jobs are sent
if_1      BRANCH,     1:                           !back to mainqueue
                      if,nq(spring_board)>0,
                      remove:
                      else,setup;
setup     DELAY:      setuptime:                   !setting up
                      MARK(arrsetup);
          TALLY:      setuptime,int(arrsetup);     !measuring setuptime
          DELAY:      processtime-0.0002;          !processing
          ASSIGN:
                      currjob=jobtype;          !assigning type of finished job
;                                               !to variable "currjob"
assmimmy1 ASSIGN:     mimmy=1;
          BRANCH,     1:
                      if,nq(mainqueue).eq.0,    !if there is no job in the main-
                      signwait1:                !queue -> wait for signal1 (from
                      else,dummydelay2;         !arriving job)
signwait1 QUEUE,      signwait1;
          WAIT:       mimmy;
dummydelay1 DELAY:    0.0001:
                      NEXT(signal2);
dummydelay2 DELAY:    0.0001;
signal2   SIGNAL:     2;
dummydelay3 DELAY:    0.0001;
          RELEASE:    forg_mach:                !releasing the machine
                      DISPOSE;
          CREATE:     ,1;                       !creating 1 job for first signal
signal2a  SIGNAL:     2:                        !to mainqueue
                      DISPOSE;
END;
```

```
BEGIN,          no;
;this is the experimental file for qu_tand
PROJECT,        qu_tand,walter;
ATTRIBUTES:     dummy:
                mimmy:
                jobtype:
                processtime:
                setuptime:
                arrsetup;
VARIABLES:      currjob,11;
RESOURCES:
                forg_mach;
QUEUES:
                mainqueue:
                signwait2:
                signwait1:
                spring_board,lvf(setuptime);
PARAMETERS:     (same as seize-priority)
TABLES:         1,1,,0,1,3,5,7,9,11,13,15,17:     !setuptimes
                2,1,,2,0,4,6,8,10,12,14,16,18:
                3,1,,17,15,0,13,11,9,7,5,3,1:
                4,1,,18,16,14,0,12,10,8,6,4,2:
                5,1,,1,3,5,7,0,17,15,13,11,9:
                6,1,,2,4,6,8,18,0,16,14,12,10:
                7,1,,9,11,13,15,17,7,0,5,3,1:
                8,1,,10,12,14,16,18,8,6,0,4,2:
                9,1,,7,5,3,1,9,11,13,15,0,17:
                10,1,,8,6,4,2,10,12,14,16,18,0:
                11,1,,1,1.5,2,2.5,3,3.5,4,4.5,    !setuptime without "prede-
                5,5.5;                            !cessor"
; no priority matrix needed!
TALLIES:        setuptime;
DSTATS:
                nq(mainqueue),queuelength;
REPLICATE,      ,,10000;
END;
```

Summary for Queue-Tandem

Replication ended at time : 10000.0

TALLY VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Observations
setuptime	.76134	2.1694	.00000	9.0000	1345

DISCRETE-CHANGE VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Final Value
queuelength	44.075	.48066	.00000	84.000	84.000

Run Time: 6 min(s) 15 sec(s)

SHORTEST RELATIVE SETUPTIME

```
BEGIN,          no,srs_first;

          CREATE:     7;
signal1   SIGNAL:     1;
          ASSIGN:
                      dummy=2:
                      jt=DP(1,2):              !assigning jobtype and processt.
                      processtime=dp(jt+2
                      );
          COUNT:      jt;                      !counting arriving jobs
mainqu    QUEUE,      mainqueue;
          WAIT:       dummy;                   !waiting for signal2 (from
;                                              !finished job)
          ASSIGN:
                      st=tf(currjob,jt):       !assigning setuptimes
                      rank_crit=               !assigning relative setupt.
                      st/(nc(jt)-nc(jt+10));
          QUEUE,      spring_board;            !queue with setuptime-ranking
          SEIZE:      forg_mach:               !job with lowest setuptime
                      NEXT(if_1);              !seizes machine
remove    REMOVE:     1,spring_board,mainqu;      !remaining jobs are sent
if_1      BRANCH,     1:                          !back to mainqueue
                      if,nq(spring_board)>0,
                      remove:
                      else,setup;
setup     DELAY:      st:                          !setting up
                      MARK(arrsetup);
          TALLY:      setup_time,int(arrsetup);    !measuring setuptime
          DELAY:      processtime-0.0002;          !processing
          COUNT:      jt+10;                       !counting leaving jobs
          ASSIGN:
                      currjob=jt;              !assigning type of finished job
;                                              !to variable "currjob"
assmimmy1 ASSIGN:     mimmy=1;
          BRANCH,     1:
                      if,nq(mainqueue).eq.0,   !if there is no job in the main-
                      signwait1:               !queue -> wait for signal1 (from
                      else,dummydelay2;        !arriving job)
signwait1 QUEUE,      signwait1;
          WAIT:       mimmy;
dummydelay1 DELAY:    0.0001:
                      NEXT(signal2);
dummydelay2 DELAY:    0.0001;
signal2   SIGNAL:     2;
dummydelay3 DELAY:    0.0001;
          RELEASE:    forg_mach:               !releasing the machine
                      DISPOSE;
          CREATE:     ,1;                      !creating 1 job for first signal
signal2a  SIGNAL:     2:                       !to mainqueue
                      DISPOSE;
END;
```

```
BEGIN,          no;
;this is the experimental file for srs_first
PROJECT,        cn1_10,walter;
ATTRIBUTES:     dummy:
                mimmy:
                jt:
                processtime:
                st:
                rank_crit:
                arrsetup;
VARIABLES:      currjob,11;
COUNTERS:       20;
RESOURCES:
                forg_mach;
QUEUES:
                mainqueue:
                signwait2:
                signwait1:
                spring_board,lvf(rank_crit);      !shortest relative setup t.
PARAMETERS:      (same as seize-priority)
TABLES:          (same as queue-tandem)
TALLIES:        setup_time;
DSTATS:
                nq(mainqueue),queuelength;
REPLICATE,      ,,10000;
END;
```

Summary for "Shortest Relative Setuptime First"

Replication ended at time : 10000.0

TALLY VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Observations
setup_time	.48258	3.3682	.00000	12.000	1378

DISCRETE-CHANGE VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Final Value
queuelength	34.525	.37802	.00000	56.000	51.000

COUNTERS

Identifier	Count	Limit
Counter 1	579	Infinite
Counter 2	271	Infinite
Counter 3	168	Infinite
Counter 4	80	Infinite
Counter 5	76	Infinite
Counter 6	52	Infinite
Counter 7	62	Infinite
Counter 8	48	Infinite
Counter 9	52	Infinite
Counter 10	41	Infinite
Counter 11	577	Infinite
Counter 12	270	Infinite
Counter 13	161	Infinite
Counter 14	75	Infinite
Counter 15	63	Infinite
Counter 16	49	Infinite
Counter 17	51	Infinite
Counter 18	41	Infinite
Counter 19	49	Infinite
Counter 20	41	Infinite

Run Time: 7 min(s) 12 sec(s)

SAME TYPE FIRST

```
BEGIN,          no,same_first;

          CREATE:     7;
          ASSIGN:
                      jobtype=DP(1,2):
                      processtime=dp(jobtype+2
                      );
          BRANCH,     1:
                      if,nq(buffer)==0.and.
                      busy==0,ass_setup:
                      else,buffer;
buffer    QUEUE,      buffer:
                      DETACH;
ass_setup ASSIGN:
                      setuptime=tf(currjob,
                      jobtype):
                      busy=1;
          DELAY:      setuptime:
                      MARK(arrsetup);
          TALLY:      setup_time,int(arrsetup
                      );
          DELAY:      processtime;
          ASSIGN:                             !determinating type of comple-
                      currjob=jobtype;        !ted job
          BRANCH,     1:
                      if,nq(1)>0,searchsame:  !checking if queue is empty
                      else,exyt;
searchsame SEARCH,    buffer:                 !searching f.job of same type
                      jobtype==currjob;
          BRANCH,     1:
                      if,j>0,remsame:         !search was successful
                      else,takefirst;         !no job of same type in qu.

remsame   REMOVE:     j,buffer,ass_setup:     !remove found job from qu.
                      NEXT(exyt);
takefirst REMOVE:     1,buffer,ass_setup:     !remove first job from qu.
                      NEXT(exyt);
exyt      ASSIGN:     busy=0:
                      DISPOSE;
END;

BEGIN,          no;
PROJECT,        same_first,walter;
ATTRIBUTES:     arrsetup:
                jobtype:
                processtime:
                setuptime;
VARIABLES:      currjob,11:
                2:
                3:
                4:
                busy;
QUEUES:         buffer;
PARAMETERS:     (same as seize-priority)
```

```
TABLES:          (same as queue-tandem)

TALLIES:         setup_time;
DSTATS:           1,nq(1);
REPLICATE,       ,,10000;
END;
```

Summary for "Same Type First"

Replication ended at time : 10000.0

TALLY VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Observations
setup_time	.71321	3.7977	.00000	18.000	1332

DISCRETE-CHANGE VARIABLES

Identifier	Average	Variation	Minimum	Maximum	Final Value
nq(1)	60.647	.39045	.00000	102.00	97.000

Run Time: 0 min(s) 32 sec(s)

References

HAYNES, H.D.; KOMAR, C.A.; BYRD, J.(JR.): The Effectiveness of Three Heuristic Rules for Job Sequencing in a Single Production Facility; in: Management Science, Vol.19, No.5, January, 1973; p. 575-80

KUSIAK, A.: Intelligent Manufacturing Systems, Englewood Cliffs, 1990

PANWALKAR, S.S.; ISKANDER, W.: A Survey of Scheduling Rules; in: Operations Research, Vol.25, No.1, Jan/Feb 1977

PEGDEN, C.D.; SHANNON, R.E.; SADOWSKI, R.P.: Introduction to Simulation Using SIMAN, Mc Graw Hill 1990