Description of the data file format ".ALB"

Armin Scholl, Christian Becker, FSU Jena

Nils Boysen, Malte Fliedner, University of Hamburg

The format ".ALB" is a flexible data format for many types of assembly line balancing problems. Descriptions of the problem types covered by the format can be found in Scholl (1999, ch. 1 and 2), Becker and Scholl (2006) as well as Boysen et al. (2007).

1 Problem characteristics and data blocks

Each file consists of data blocks, each of which characterizes specific information for a particular problem characteristic . Each block is introduced by a keyword / tag and ends when another tag follows. The tags are formatted as follows (`courier font` and blue color indicate that an expression is part of the format definition):

```
<keyword>
```

Inside the tag spaces are not allowed. After the closing ">" comments are allowed.

2 Structuring information within the blocks

The data values are specified line by line in a human as well as machine readable structure. Each data construct demands an own line and consists only of natural numbers and the following separation characters:

`,`	The comma separates pairs of nodes, different indexes to assign information to objects.
`:`	The colon separates index part and value part of an information.
`;`	The semicolon separates different values

When a value of a certain parameter x has to be given instead of its name, the notation val(x) is used. Additional spaces separating the elements are allowed.

3 Available data blocks

The following blocks, some of which are mandatory, are defined. Because some of them use information given by others it is recommended to use the given order of writing the blocks into the file.

3.1 Number of tasks

In a mandatory block, which must be the first one given, the number n of tasks is specified:

```
<number of tasks>
val(n)
```

3.2 Cycle time

In case that the cycle time c must be prespecified, this block is mandatory.

```
<cycle time>
val(c)
```

3.3 Number of stations

If the number m of stations is fixed (e.g. in case of an existing line), the following block is necessary.

```
<number of stations>
val(m)
```

3.4 Task times

This block is mandatory and specifies the task times t_i. The tasks must be numbered consecutively from 1 to n and be given in this sequence i=1,...,n. In order to screen the correctness of task-time-assignments the task number must be specified, too. In older versions of data files, the task IDs an the times might also be separated by a space.

```
<task times>
1:val(t_1)
2:val(t_2)
...
val(n):val(t_n)
```

If task processing alternatives exist (cf. section 3.15), the task times given in this block are those of the basic process.

3.5 Precedence relations

The precedence relations define restrictions for the sequence of performing tasks. Provided that the tasks i = 1,...,n constitute the nodes of a graph, an arrow (i,j) is added to the graph if task i must be completed before task j can be started. The precedence relation is direct if there is no other path in the graph which connects i and j.

In order to keep data examination simple, it is necessary to number the tasks according to a topological ordering, i.e., each precedence relation (i,j) fulfills $i < j$. Furthermore, only direct precedence relations should be contained, while transitive relations should be removed.

The precedence relations are given line by line using the task numbers separated by a colon:

```
<precedence relations>
val(i),val(j)
...
```

3.6 Sequence dependent task time increments

In some cases, performing a task j is complicated by another task i performed earlier. For example, a part mounted by task i partially conceals the mounting place for task j or sets an obstacle for performing task j. This results in a sequence dependent task time prolongation denoted by sd_{ij}, i.e. the time of task j is increased to $t_j' = t_j + sd_{ij}$ when i is performed before j.

```
<sequence dependent time increments>
i,j:val(sd[i,j])
```

3.7 Station cost

Concerning a particular planning horizon of length T, the station cost sc is the total cost caused by installing and operating a station at the assembly line. Instead of giving the station cost as a total cost it can be specified as cost per product unit scp. Given the cycle time c, the unit station cost is computed as $scp = \frac{sc \cdot c}{T}$.

`<total station cost>`	`<station cost per unit>`
`val(sc)`	`val(scp)`

In case that different stations differ in cost, the individual total cost sc_k or individual unit cost scp_k may be specified for the stations k = 1,...,m with m being the given number of stations or a sufficiently large upper bound on the number of stations. :

`<total individual station cost>`	`<individual station cost per unit>`
val(sc_1)	val(scp_1)
...	...
val(sc_m)	val(scp_m)

Among the different possibilities only one should be contained in the input file.

3.8 Task execution cost

Executing the tasks i=1,...,n may cause task related cost. Depending on the basis of computation, one may specify total task cost per planning horizon tc_i or the task cost per product unit tcp_i:

`<total task cost>`	`<task cost per unit>`
val(1):val(tc_1)	val(1):val(tcp_1)
...	...
val(n):val(tc_n)	val(n):val(tcp_n)

3.9 Parallel stations

Parallel stations are duplicates of some serial station such that the local cycle time is a multiple of the global cycle time.

The maximal number of times mp a station can be installed in parallel is specified as follows:

```
<maximum degree of parallelism>
val(mp)
```

3.10 Station equipment

In case that specialized equipment is necessary to perform certain tasks, the stations have to be equipped accordingly.

The number ne of different equipments which are required by any task is defined by:

```
<number of equipments>
val(ne)
```

The equipments are labelled from 1 to ne. Installing an equipment q = 1,...,ne causes a cost ec_q .

```
<equipment cost>
1:val(ec1)
...
val(ne):val(ecne)
```

The equipments necessary for performing a task i=1,...,n are given as a list of their labels segregated by commas. Any task which does not require specialized equipment must not appear in the list. The order of the tasks is arbitrary. The following block definition is an example which assigns the equipments 2 and 4 to task 1 and the equipments 1 and 3 to task 2:

```
<equipments per task>
1:2,4
2:1,3
...
```

3.11 Wage cost

When operating (manual) assembly lines, wages have to be paid to workers. In case that each worker gets the same wage rate irrespective of his qualification and the tasks he has to perform, the wage cost is given as a total cost wc per planning horizon, per product unit (wcp), or per time unit (wct).

```
<total wage>          <wage per unit>          <wage per time>
val(wc)               val(wcp)                 val(wct)
```

Frequently, the wages paid depend on the qualification needed to perform the tasks. Then, the wage rates are specified as task dependent wage rates wc_i (or wcp_i or wct_i) for all tasks i=1,...,n:

<total task wage>	<task wage per unit>	<task wage per time>
1:val(wc_1)	1:val(wcp_1)	1:val(wct_1)
...	...	...
val(n):val(wc_n)	val(n):val(wcp_n)	val(n):val(wct_n)

3.12 Task-to-task assignment restrictions

In some cases, two tasks have to be assigned to the same station. This may, e.g., be caused by an expensive equipment which can only be installed once. All such pairs (i,j) of linked tasks are given in the following data block line by line:

```
<linked tasks>
val(i),val(j)
...
```

Whenever, two tasks must not be performed in the same station, e.q. drilling and measuring operations, the respective task pair (i,j) is incompatible. All incompatible task pairs are listed in the following data block:

```
<incompatible tasks>
val(i),val(j)
...
```

3.13 Task-to-station assignment restrictions

When assigning tasks to stations there may be some restrictions which avoid or fix certain combinations.

If a task i has to be performed in a certain area of the assembly line, given by a sector between its first possible station $\underline{sf}_i$ and its last possible station $\overline{sf}_i$, this task-station-assignment is fixed as follows:

<tasks fixed to sector>

val(i):val($\underline{sf}_i$),val($\overline{sf}_i$)

...

Note that $\underline{sf}_i = \overline{sf}_i$, if there is only one station where task i can be performed.

The other way round, it may be forbidden to perform a task i at some stations which are specified as a list of station numbers following the task number i. An example, which excludes the task 1 from the stations 3, 5, and 6 as well as the task 2 from the stations 4, 7, and 8, is given below:

```
<tasks excluded from station>
1:3,5,6
2:4,7,8
...
```

3.14 Station load restrictions

A station load is defined as a set of tasks assigned to that station. The most important aspect to be considered when forming station loads is the cycle time restriction, i.e., the sum of task times must not exceed the cycle time.

Additionally, there may be further restrictions on the feasibility of station loads. For example, each task may require certain parts to be mounted at a work piece. These parts must be stored aside the line. However, the space available may be restricted such that this load is not feasible due to violating the space restriction. Another example is present in case of job enrichment, where it may be necessary to assign at least a certain number of different tasks to a station.

In order to formulate such restrictions in a general manner, a number na of task attributes can be defined.

```
<number of task attributes>
val(na)
```

Whenever a task i contributes to an attribute a=1,...,na, the contribution value v_{ia} is given in the following data block. All values not equal to zero are listed line by line.

<task attribute values>

val(i),val(a):val(v_{ia})

...

Each attribute a=1,...,na is associated with a lower bound lv_a and/or an upper bound uv_a on the cumulated value which must be fulfilled by a station load to be feasible for the attribute a. If only an upper bound (for example maximal storage space) is required, the lower bound can be set to ’n.a.’ (not available) and vice versa as shown in the formatting example below.

<attribute bounds per station>

1:val(lv_1),val(uv_1)

...

5:n.a.,100

6:20,n.a.

...

na:val(lv_{na}),val(uv_{na})

3.15 Processing alternatives

In many cases, there a several technological possibilities to perform a single task, a subassembly or the complete assembly process.

In case that each task can be done in an individual manner, there may be alternatives to the basic processing documented by the basic task times (cf. section 3.4) and the basic task execution cost (cf. section 3.8). These alternative execution processes p = 1,..., np(i) are listed following the corresponding task i, seperated by semicolons. Each alternative is given as pair (t_{ip},tc_{ip}) of execution time t_{ip} and execution cost tc_{ip}, seperated by commas. Only the tasks having processing alternatives must be specified in arbitrary order.

```
<task process alternatives>
```

$i:t_{i1},tc_{i1};t_{i2},tc_{i2};...;t_{i,np(i)},tc_{1,np(i)}$

```
...
```

3.16 Mounting positon

In case that there are two or more workers in each station, they are not able to work simultaneously in the same area of the workpiece (e.g. a car). So different mounting positions are defined and in each station only one worker can work at each mounting position (e.g. on the left front side of the car). Every task i demands a number nmp(i) of mounting positions ($nmp(i) \geq 0$).

```
<mounting position>
```

$i:val(mp_{i1});val(mp_{i2});...;val(mp_{i,nmp(i)})$

```
...
```

3.17 Incompatible mounting positions

On big workpieces it is not meaningful to let one worker move from one end to an other, e.g. from the right rear side to the left front side of a car. If two tasks have different mounting positions i and j ($i \neq j$) that are incompatible among each other, they must not be assigned to the same workstation.

```
<incompatible mounting positions>
```

$val(mp_i),val(mp_j)$

```
...
```

3.18 Termination

In order to enable a file parser to recognize that all data blocks are finished the file should be terminated with the following tag:

```
<end>
```

4 References:

Scholl, A. (1999): Balancing and sequencing of assembly lines. 2nd ed., Physica, Heidelberg.

Becker, C.; Scholl, A. (2006): A survey on problems and methods in generalized assembly line balancing. "Invited Review" in: European Journal of Operational Research 168, S. 694-715.

Boysen, N.; Fliedner, M.; Scholl, A. (2007): A classification of assembly line balancing problems. European Journal of Operational Research 183, 674-693.