STOCHASTIC ALGORITHMS FOR THE LINE BALANCING PROBLEM IN AUTOMOTIVE INDUSTRY

Corinne Boutevin, Michel Gourgand, Sylvie Norre

Université Blaise Pascal – Clermont-Ferrand II
LIMOS CNRS FRE 2239
Campus Scientifique des Cézeaux
F - 63177 Aubière Cedex,
boutevin@iris.univ-bpclermont.fr, gourgand@isima.fr, norre@moniut.univ-bpclermont.fr

Abstract: This paper deals with the study of a line balancing problem. The considered problem is issued from the automotive industry. It consists in gradually assembling vehicles which go through a line. Assembly operations are made by workstations placed along the line. The goal is to assign operations to these workstations in order to minimize different criteria while satisfying several constraints. To solve this problem, stochastic algorithms are applied, namely stochastic descent and simulated annealing, for which two neighboring systems are proposed. They are applied on generated data and industrial data and provide interesting results. *Copyright © 2002 IFAC*

Keywords: automotive industry, decision-making, mathematical models, operations research, optimization problems.

1. INTRODUCTION

The line balancing problem is a very well-known problem in the automotive industry. This kind of industry uses assembly lines for manufacturing vehicles. The density (number of considered vehicles) and the diversity (number of types of vehicle) of the production require an optimization. The literature lists four classical models: the "single-model" (only one type of vehicle is considered), the "mixed-model" (several types of vehicle are considered), the "multi-model" (vehicles issued from the same type are manufactured in fixed batches) and the "batch-model" (a multi-model where the dimensions of batches must be computed). All these models have been studied, for instance, in (Bhattacharjee and Sahu, 1987; Van Zante-de Forkkert, 1997; Scholl, 1999; Rekiek, 2001).

The line balancing problem consists in searching a good assignment of operations to workstations. This optimization is difficult due to the combinatory (number of workstations, operations and types of vehicle) of this industrial problem and the number of constraints.

The problem is NP-complete (Scholl, 1999) even if there is one type of vehicle and no precedence constraints. It is solved, in the literature, with exact methods for small instances (Van Zante-de Forkkert, 1997) and with heuristics (Bhattacharjee and Sahu, 1987; Palekar, 1998; Scholl, 1999) for large instances (more than 50 operations).

To solve the line balancing problem, stochastic algorithms have been used: stochastic descent and simulated annealing. In this paper, two kinds of neighboring system are presented, used in the stochastic algorithms. Different objective functions have been tested, such as economical criteria.

In the first part, the industrial problem is presented. In the second part, a formalization of this problem is proposed. In the third part, two neighboring systems for stochastic algorithms are given. In the fourth part, computational results are presented, based on generated data and on two industrial cases.

2.PROBLEM PRESENTATION

The line is divided into several sections (figure 1), which contain several workstations (at most, 5 workstations); a workstation is assimilated to one operator. An operator realizes a set of operations on the vehicles; these operations depend on the type of the vehicle. Vehicles go through a line, with a constant speed, to be assembled. So the vehicle is available on each section during a same duration, all operations assigned to operators placed on the

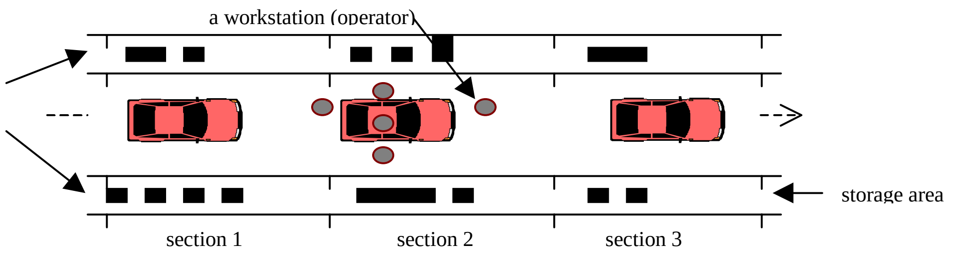

Fig. 1. Description of a line.

section must be realized during this period. An operation has its own execution time.

Different types of vehicle are considered; so the studied problem is a mixed-model. The set of operations required by a vehicle differs according to the type. So the load of an operator can considerably change according to the type of vehicle.

The realization of an operation requires different pieces and tools and containers for pieces. They must be placed along the line, on each side of the section containing the workstation (i.e. the operator) realizing the operation. So there is a storage area on each side of the section, for placing pieces and tools. As the storage area is limited and pieces and tools use place, the number of operations which can be assigned to a section (assigned to any workstation placed on the section) is limited.

For a better comprehension of the problem, the figure 2 shows the UML diagram (UML, 1997) of a line, in which it is shown that:

- a line is divided in several sections, which contain several workstations (a section may be empty),
- a workstation is used by one and only one operator and realizes several operations,
- an operation realization requires tools and pieces; pieces are stored in racks or carriages.

So operations have to be assigned to workstations in order to satisfy the following constraints:

- ***C1****: the cycle time*
  On a section, a vehicle is available for a workstation, during a certain time called the cycle time. This duration represents the time between the exits of two consecutive vehicles from the line (Rekiek, 2001), i.e. the time between the arrivals of two consecutive vehicles on a section. Each workstation has to realize all its assigned operations during the cycle time, for any vehicle.
- ***C2****: the length of sections*
  The assignment of an operation to a workstation induces the storage of pieces and tools (required for the operation realization) on the section containing the workstation. However, pieces and tools have to use a distance inferior to twice the length of the section.
- ***C3****: the operator time*
  Each day, a list of vehicles must be assembled. So the workstation has to realize all his assigned operations on all the considered vehicles. All these operations must be realized during the operator time which represents the daily work time of the workstation.
- ***C4****: the incompatibility between two operations*
  It can happen that two operations cannot be realized by the same workstation. This is a dissociative constraint.

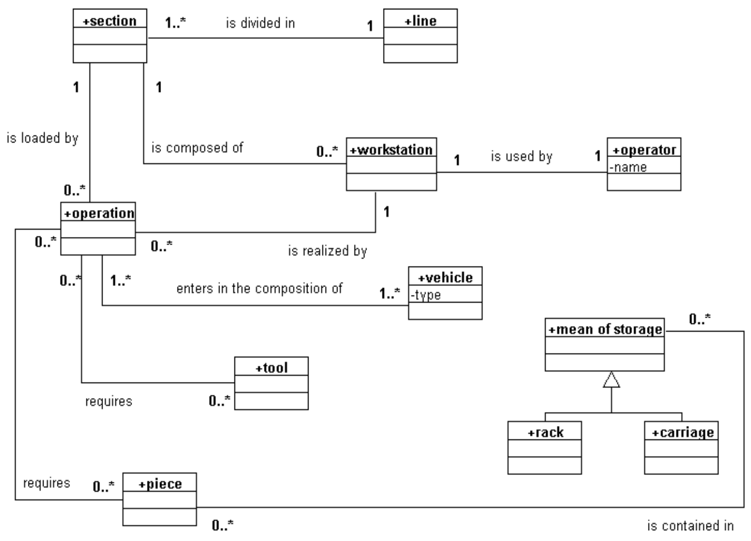

Fig. 2. Domain analysis UML classes diagram of an assembly line.

- ***C5**: the precedences between two operations*
  It can happen that an operation must be realized before another one (on a previous section).
- ***C6**: the impossible / obligatory assignments*
  It can happen that an operation must be (not be) assigned to a particular workstation.

Two cases are considered:

- non existing balancing: an initial solution (a balancing) is randomly generated. It necessarily satisfies constraints C4 to C6 but can be unfeasible for C1 to C3,
- existing balancing: a solution corresponding to a previous period. So the goal is to find a new line balancing corresponding to the next period. The previous balancing can be unfeasible for this next period (possible violation of C1, C2 or C3; but C4, C5 and C6 are not violated). So this balancing has to be modified to obtain a new one, which must be good and feasible. This modification is obtained by moving operations to other workstations.

So operations have to been assigned to workstations in order to satisfy all the constraints. The assignments are made in order to minimize a criterion such as, for instance:

- the number of used workstations (used means that the workstation realizes at least one operation),
- the number of used sections (used means that the section contains at least one used workstation),
- the number of moved operations in the case of an existing line (for an initial existing solution which is feasible or not),
- a linear combination of the three previous criteria (in order to make hierarchical optimization).

When a linear combination of the criteria is chosen, costs for each criterion are used, in the way of privilege one or more criteria (maintenance of workstations and/or sections, number of moved operations, …).

3.PROBLEM FORMALIZATION

3.1 Notations

- OW: matrix linking operations to workstations $OW_{i,j}$ = 1 if the operation *i* is assigned to the workstation *j*, 2 if it can not be assigned to this workstation and 0 otherwise
- OWI: the initial OW matrix (associated to the initial solution)
- WS: matrix linking workstations to sections $WS_{j,k}$ = 1 if the workstation *j* is placed on the section *k*, and 0 otherwise
- OT: matrix linking operations to types of vehicle. $OT_{i,m}$ = 1 if the operation *i* is realized on vehicles of the type *m*, and 0 otherwise
- IN: matrix linking incompatible operations. $IN_{i_1,i_2} = 1$ if the operations i_1 and i_2 are incompatible, and 0 otherwise
- PR: matrix linking operations with a precedence relation. $PR_{i_1,i_2} = 1$ if the operation i_1 must precede the operation i_2, and 0 otherwise
- TD_i: timed duration of the operation *i*
- AD_i: allocated duration of the operation *i* ($AD_i = Ce\ .TD_i$, where Ce is the effort coefficient, equal for instance to 1.2)
- LS_k: length of the section k
- T_l: type of the vehicle *l*
- N_m: number of vehicles issued from the type *m*
- D_i: distance used by the operation *i* for placing pieces and tools required for the operation realization
- WOR = {workstations}
- SEC = {sections}
- OPE = {operations}
- TYP = {types of vehicle}
- VEC = {vehicles}
- $A_j = \{i \in OPE \ / \ OW_{i,j} = 1 \}$
- $A_{jm} = \{i \in OPE \ / \ OW_{i,j} \ . \ OT_{i,m} = 1\}$ $\left(A_j = \bigcup_{m \in TYP} A_{j,m} \right)$
- nmov: number of operations moved from a workstation to another one
- costmov: cost of an operation moving
- nwork: number of used workstations
- costwork: cost of a workstation maintenance
- nsection: number of used sections
- costsection: cost of a section maintenance
- cyctime: cycle time
- opertime: operator time

3.2 Constraints formalization

Then, using these notations, the six previous constraints can be formalized as follows:

- ***C1***: $\forall j \in WOR$

$$\underset{m \in TYP}{\mathrm{Max}} \sum_{i \in A_{jm}} TD_i \, OW_{i,j} (2 - OW_{i,j}) \le cyctime$$

- ***C2***: $\forall k \in SEC$

$$\sum_{j \in WOR} \sum_{i \in A_j} D_i \, OW_{i,j} (2 - OW_{i,j}) WS_{j,k} \le LS_k$$

- ***C3***: $\forall j \in WOR$

$$\sum_{m \in TYP} \sum_{i \in A_{jm}} AD_i \, OW_{i,j} (2 - OW_{i,j}) N_m \le opertime$$

- ***C4***: $\forall j \in WOR, \forall (i_1, i_2) \in OPE^2$

$$IN_{i_1,i_2} (OW_{i_1,j} + OW_{i_2,j}) \le 1$$

- ***C5***: $\forall (i_1, i_2) \in OPE^2 \ / \ PR_{i_1,i_2} = 1$

$$\sum_{k \in SEC} \sum_{j \in WOR} k \, OW_{i_1,j} (2 - OW_{i_1,j}) WS_{j,k} \le$$
$$\sum_{k \in SEC} \sum_{j \in WOR} k \, OW_{i_2,j} (2 - OW_{i_2,j}) WS_{j,k}$$

- ***C6***: this last constraint is included in all the other constraints with the factor $(2 - \mathrm{OW}_{\mathrm{i,j}})$.

All these constraints are used in the proposed stochastic algorithms to test the feasibility of solutions. The obtained model is on linear.

4.NEIGHBORING SYSTEMS PROPOSITION

In this part, are presented neighborhoods for stochastic algorithms for solving the line balancing problem. The implemented methods are stochastic descent and simulated annealing.

4.1 Proposition of two neighboring systems.

A solution is an assignment of operations to workstations. So a neighboring solution is built by moving an operation to another workstation. From a current solution, this solution is slightly modified to obtain a new one, which is neighbor from the first one.

A randomly neighboring system.

With this neighboring system, the operation to move and the final workstation are randomly chosen. The algorithm of this system is:

R
- an operation *i* is randomly chosen among all operations
- let be j_1 the workstation which presently realizes the operation *i*
- a workstation $j_2 \neq j_1$ is randomly chosen
- the operation *i* is assigned (moved) to the workstation j_2

A guided neighboring system.

This second neighboring system consists in not choosing randomly the operation to move, the starting and the recipient workstations: the chosen operation is the one with the highest execution time. And the workstations are chosen among the used workstations (workstations which realize one or more operations). So the associated algorithm is:

G
- a workstation j_1 is randomly chosen
- the operation *i* is the one with the highest execution time, presently realized by the workstation j_1
- a workstation $j_2 \neq j_1$ is randomly chosen
- the operation *i* is assigned (moved) to the workstation j_2

When a new solution is built (by using R or G), not all the six constraints are used. So the neighboring systems can be applied for any industrial balancing problem.

4.2 The algorithm.

The presented algorithm concerns the stochastic descent and the simulated annealing. The normal text is for the stochastic descent and the simulated annealing requires also the bold text. The used neighboring system must be chosen between R and G previously presented. It is fixed during the algorithm execution.

The general algorithm is as follows:

```
- let be OWI an initial solution which is feasible or
  not. OWI is read or randomly created,
- let be OWB the matrix giving the best found
  solution
- let be HX the value of criterion (to minimize) for
  the present solution
      HX = nmov . costmov + nwork . costwork +
           nsection . costsection,
- let be HY the value of the criterion for the
  neighboring solution OW,
- let be HBX the value of the criterion for the best
  solution let be nbitermax the maximal number of
  iterations,
- let be f a non increasing function                      [bold]
- at the beginning: OWB = OWI and HBX = HX
  (HX = value of the criterion associated to OWI),
- initialization of the temperature $T_0$                  [bold]
- for nbiter = 1, …, nbitermax do
  ▪ $T_{nbiter} = f(T_{nbiter-1})$                         [bold]
  ▪ generation of a neighboring solution OW using
    R or G, and computation of HY
  ▪ if all constraints (C1 to C6) are satisfied then
    ▫ if HY ≤ HX then
      ⅄ OWB = OW
      ⅄ if HY < HX then
        ∂ HX = HY
        ∂ if HY < HBX then.
          € HBX = HY
        ∂ end if
      ⅄ end if
    ▫ else                                                 [bold]
      ⅄ choose randomly q between 0 and 1                  [bold]
      ⅄ let be p = exp ($-\frac{HY-HX}{T}$)                [bold]
      ⅄ if q < p then                                      [bold]
        ∂ OWB = OW                                         [bold]
        ∂ HX = HY                                          [bold]
      ⅄ else                                               [bold]
        ∂ OW = OWB                                         [bold]
      ⅄ end if                                             [bold]
    ▫ end if
  ▪ end if
- end for
```

5.COMPUTATIONAL RESULTS

In this part, are presented computational results based on generated data and industrial examples. The table 1 describes the three studied generated examples.

Table 1 Description of the data

Examples	**1**	**2**	**3**
Number of operations	40	100	500
Number of workstations	10	41	100
Number of sections	6	9	24
Number of types of vehicle	10	10	25
Number of vehicles	100	100	250

In the stochastic algorithms, three types of initial solution are used:

- feasible ***(I)***
- violation of the constraint C1 ***(II)***
- violation of the constraints C1, C2 and C3 ***(III)***

In the following tables, the used notations are:

- ***SD_R***: stochastic descent with the randomly neighboring system
- ***SD_G***: stochastic descent with the guided neighboring system
- ***SA_R***: simulated annealing with the randomly neighboring system
- ***SA_G***: simulated annealing with the guided neighboring system

The objective function is chosen among those:

- the number of used workstations and the number of moved operations (***HX1***: costsection=0, costmov=1, costwork=1000)
- the number of used sections (***HX2***: costsection=1, costmov=0, costwork=0)

The tables 2 to 4 show results obtained with a stochastic descent and a simulated annealing implemented with the randomly neighboring system R, for the three generated examples. The initial solutions are not necessarily feasible but the final solutions, given by the methods, are always feasible. Best results are bold.

The guided neighboring system G has been also tested. The table 5 shows results obtained with a feasible initial solution, by using the two neighboring systems. It can be noticed that the guided neighboring system provides better results: the number of workstations is divided by around three. And the quality of results increases as the combinatory of the problem grows.

Table 2 Results for the example 1

initial	**goal**		**SD_R**			**SA_R**	
soltion	**function**	***nwork***	***nsection***	***nmov***	***nwork***	***nsection***	***nmov***
I	***HX1***	5	4	28	**5**	**3**	**16**
	HX2	**10**	**4**	**1**	**10**	**4**	**1**
II	***HX1***	5	4	33	**5**	**4**	**15**
	HX2	**10**	**4**	**1**	**10**	**4**	**1**
III	***HX1***	5	4	26	**5**	**4**	**16**
	HX2	**10**	**4**	**1**	**10**	**4**	**1**

Table 3 Results for the example 2

initial	**goal**		**SD_R**			**SA_R**	
soltion	**function**	***nwork***	***nsection***	***nmov***	***nwork***	***nsection***	***nmov***
I	***HX1***	11	6	90	**11**	**7**	**71**
	HX2	**14**	**7**	**98**	**14**	**7**	**98**
II	***HX1***	**13**	**8**	**90**	41	9	0
	HX2	**25**	**8**	**99**	41	9	0
III	***HX1***	14	7	91	**14**	**9**	**58**
	HX2	**25**	**8**	**99**	**25**	**8**	**99**

Table 4 Results for the example 3

initial	**goal**		**SD_R**			**SA_R**	
solution	**function**	***nwork***	***nsection***	***nmov***	***nwork***	***nsection***	***nmov***
I	***HX1***	60	22	486	**59**	**24**	**200**
	HX2	74	22	491	**56**	**21**	**496**
II	***HX1***	56	23	485	**53**	**23**	**486**
	HX2	100	24	1	**63**	**21**	**496**
III	***HX1***	**100**	**24**	**0**	**100**	24	**0**
	HX2	**100**	**24**	**0**	**100**	24	**0**

Table 5 Results with the guided neighboring system

Examples	**goal function**	**SD_R**			**SA_R**			**SD_G**			**SA_G**		
		nwork	*nsection*	*nmov*	*nwork*	*nsection*	*nmov*	*nwork*	*nsection*	*nmov*	*nwork*	*nsection*	*nmov*
1	***HX1***	6	4	28	5	3	16	4	4	37	**4**	**4**	**33**
	HX2	10	4	1	10	4	1	**6**	**3**	**29**	**6**	**3**	**29**
2	***HX1***	11	6	90	11	7	71	11	6	88	**10**	**7**	**80**
	HX2	14	7	98	14	7	98	16	6	68	**13**	**5**	**92**
3	***HX1***	60	22	486	56	23	230	40	18	439	**39**	**20**	**598**
	HX2	74	22	491	74	22	491	**41**	**18**	**438**	**41**	**18**	**438**

The table 6 concerns industrial data and the table 7 shows results. The conclusion is that the guided neighboring system gives best results.

Table 6 Description of industrial examples

Examples	**Line 1**	**Line 2**
Number of operations	310	861
Number of workstations	26	76
Number of sections	16	22
Number of types of vehicle	935	2012
Number of vehicles	17021	14280

6. CONCLUSION

In this paper, the line balancing problem has been studied and two neighboring systems (a randomly one and a guided one) have been implemented in stochastic algorithms. Best results have been obtained by using a guided neighboring system. But It can be noticed that these methods take a long time (approximately 6 hours with a PIII 800 Mhz) to solve a problem with a consequent combinatory (around 500 operations to assign to 50 workstations). So they have to be improved in order to treat real data.

The same problem has been also solved with heuristics (Boutevin and Norre, 2002). In the future, it will be motivating to study couplings between these heuristics and stochastic algorithms.

Now it is interesting to study the smoothing of the load of the line, which consists in distributing operations to workstations in order that there is an equivalent workload between each workstation and between each section. A first approach is actually studied using an heuristic.

It is also interesting to study the planning problem which is the distribution of operations among several lines and the operations sequencing on each line.

REFERENCES

Bhattacharjee, T.K. and S. Sahu (1987). A Critique of Some Current Assembly Line Balancing Techniques. *Indian Institute of Technology,* Kharagpur, Inde.

Boutevin,C. and S. Norre (2002). *"Equilibrage de la charge sur une ligne de montage de véhicules"*. Journal Européen des Systèmes automatisés (to appear).

Palekar, U.S. (1998). *"Assembly Line Balancing"*. Courses Notes.

Rekiek, B. (2001). Assembly Line Design, Multiple Objective Grouping Genetic Algorithm and the Balancing of Mixed-model Hybrid Assembly Line. Thesis, Université Libre de Bruxelles (Belgique).

Scholl, A. (1999). "Balancing and Sequencing of Assembly Lines". Edition Physica-Verlag Heidelberg.

UML, (1997). Specification of UML, Version 1.0, 13 January 1997 (Rational Software Corporation).

Van Zante-de Fokkert, J.I. and T.G. De Kok (1997). *"The Mixed and Multi Model Line Balancing Problem: a Comparison"*. European Journal of Operational Research (100), pp.399-412.

Table 7 Results for industrial examples

Examples	**goal function**	**SD_R**			**SA_R**			**SD_G**			**SA_G**		
		nwork	*nsection*	*nmov*	*nwork*	*nsection*	*nmov*	*nwork*	*nsection*	*nmov*	*nwork*	*nsection*	*nmov*
1	***HX1***	26	16	0	25	15	122	21	16	174	**10**	**9**	**241**
	HX2	26	15	57	26	11	296	22	13	47	**9**	**6**	**232**
2	***HX1***	76	22	0	76	22	0	68	22	52	**58**	**21**	**390**
	HX2	**76**	**22**	**0**	**76**	**22**	**0**	**76**	**22**	**0**	**76**	**22**	**0**