Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Real Time Balancing of Complex Disassembly Lines

L. Duta*. F. G. Filip,**
I. Caciula***

**Computer Science Department, Valahia State University, Targoviste, 130065 ROMANIA (Tel: +40-723613305; e-mail: duta@valahia.ro).*
***The Romanian Academy and The National Institute forR&D in Informatics (ICI), Bucharest, ROMANIA (e-mail: ffilip@acad.ro ,filipf@ici.ro)*
**** Control Engineering Student, Valahia State University, ROMANIA (e-mail: viorel2004_ro@yahoo.com)*

Abstract: The objective of the Disassembly Line Balancing Problem (DLBP) is to use the resources of the disassembly line as efficiently as possible while meeting the demand. This issue is hard to attempt due to the inherent uncertainties that occur during the process. Starting from real industrial examples, this article presents a simple-to-apply method to accomplish the balancing of complex disassembly lines in real time. The basic idea of this method is to use mixed integer quadratic programming and branch and cut algorithm on the disassembly precedence graph. Results of simulations for disassembling two industrial products are presented

1. INTRODUCTION

Disassembly processes decompose products into parts or subassemblies in view of their reuse. They may be viewed as complex processes because are subject to various and unexpected perturbations and uncertainties caused by the used end-of-life state of the product.

The most difficult problem in a disassembly system is that a disassembly operation can fail any time because of the product or component degradation. In this case we have to choose between applying an alternative destructive disassembly operation (dismantling), and abandoning the disassembly procedure.

Once a perturbation occurs, a fast computation has to be made so as to assure the optimal assignment of the tasks to workstations. The decision must be taken in real time since in a used product the components states are not known from the beginning of the process and they could influence the flow of the process and imbalance the line. That is why the problem of *Disassembly Line Balancing in Real time* (DLBP-R) is a very complex and challenging one.

Balancing a disassembly line in real time means to equalize the station loads during the disassembly process by taking into account the tasks that have not been accomplished yet and the appropriate disassembly strategy (destructive or not) associated to these tasks so as complete the disassembly processing during the rest of the working time.

To accomplish the DLBP-R, the Mixed Integer Quadratic Programming (MIQP) method is proposed to be utilised and the results on two industrial applications are described. The remaining part of this paper is organised as it follows.

First the mathematical model of the DLBP is proposed. Next the main concepts and XPRESS-MP software product are reviewed. Two case studies taken from literature and the corresponding experimental results are presented in sections 4 and 5, respectively.

2. MATHEMATICAL MODEL FORMULATION

2.1. Preliminaries

What follows will be based on the following assumptions:

- the Disassembly Line is single-product and all the operations to be performed and their precedence relations may be represented by a single precedence graph, where nodes represent generic operations.
- two values of the operative time are associated to each operation: a disassembly time and a dismantling time. Operative times take the zero value in the case of a component missing in the used product.
- the architecture of the line is flow-shop
- a workstation can accomplish both destructive and non-destructive operations
- the number of workstations is fixed
- the flow on the line is continuous since the supply of the product is considered infinite

The following notations are used in this paper:

n	number of workstations
m	number of tasks
I	index of a workstation

j or k	index of a task
t_{cy}	cycle time
t_j	operational disassembly time of the task j
t'_j	operational dismantling time of the task j
f , F	objective function
W_i	workstation i
m_i	number of tasks accomplished on workstation i

2.2. Objective function

Considering the *balancing function* presented in (Duta *et al*, 2005) which gives the difference between the operational times and the cycle time the next formula gives one objective function to be minimised:

$$f = \sum_{i=1}^{n} \left(t_{cy} - \sum_{j \in W_i}^{m_i} t_j \right)^2 \qquad (1)$$

Minimizing the value of this function leads to a well-balanced disassembly line (Rekiek and Delchambre, 2006). In a static approach of the disassembly process the cycle time is defined as the operational time of the slowest workstation on the line (Nof, 1997; Gungor and Gupta, 1999).

$$t_{cy} = \max_{W_i} \sum_{j \in (tasks\ on\ W_i)} t_j \qquad (2)$$

A problem occurs in the case of the real time disassembly: the operation can be fulfilled or not or there are destructive operations to accomplish. The aim is to find the form of the objective function in real time. A form of this function was proposed in (Duta *et al*, 2007). In the mathematical model of the DBLP-R the form of the objective function is given by the equation (3). The aim is to obtain the minimal value of the cycle time:

$$F = \sum_{i=1}^{n} \left[t_{cy} - \sum_{j=1}^{m} \varphi_{ij} \psi_j \left[\theta_j t_j + (1-\theta_j) t'_j \right] \right]^2 \qquad (3)$$

Where:

φ_{ij} is *the assignment coefficient* that defines assignment of tasks to stations for different products. It may take the following values:

$\varphi_{ij} = 1$, when the operation O_j can be assigned to workstation W_i

$\varphi_{ij} = 0$, otherwise

ψ_j is *the state coefficient* that defines which operation has already been performed and which one is still to be done. It may take the following values:

$\psi_j = 1$, if operation O_j has not still been performed

$\psi_j = 0$, otherwise.

θ_j *is the decision coefficient* that defines the modality of performance for the operations. It may take the following values:

$\theta_j = 1$ when the operation O_j is to be performed without damaging the product

$\theta_j = 0$ when the operation O_j has to be performed in a destructive way on product

2.3. Constraints

Before the beginning of the disassembly process when no operation have been done yet, $\psi_j = 1 \ \forall j \in \overline{1..m}$.

If the cycle time is considered a positive sum and its formula from equation (2) is taken into account, it means that this variable is *an upper bound* on the workload assigned to each workstation. Therefore, for every $i = \overline{1..n}$ the following constraint is valid:

$$\sum_{j=1}^{m} \varphi_{ij} \psi_j \left[\theta_j t_j + (1-\theta_j) t'_j \right] \le t_{cy} \qquad (4)$$

Every task is performed on a single workstation. A task cannot be divided between workstations. Equation (5) represents *the non divisibility constraint*.

$$\sum_{i=1}^{n} \varphi_{ij} = 1 \quad \forall j = \overline{1..m} \qquad (5)$$

If task k is to done before task j then it cannot be assigned to a station downstream from task j (Nof, 1997), the *precedence constraint* is obtained.

$$\sum_{i=1}^{n} i \cdot \varphi_{ik} - \sum_{i=1}^{n} i \cdot \varphi_{ij} \leq 0 \qquad (6)$$

All coefficients from the previous paragraph may take binary variables. A possible combination of the three binary coefficients is given in the Table 1.

Table 1.

No	φ_{ij}	ψ_j	θ_j	Meaning
1	1	0	0	Operation j made in a destructive way on the workstation i
2	1	0	1	Operation j made in a non-destructive way on the workstation i
3	1	1	0	Operation j has to be made in a destructive way on the workstation i
4	1	1	1	Operation j has to be made in a non-destructive way on the workstation i

Next constraint expresses the fact that a task can be done in two ways: destructive or not.

$$\theta_j \in \{0,1\} \qquad (7)$$

Equation (3) together with equations (4), (5), (6), and (7) form a linear mathematical model. In this model all variables (excepting the cycle time and the operational times) are binaries.

3. MIXED INTEGER QUADRATIC PROGRAMMING

3.1. The algorithm

The optimisation problem consists in calculating the minimal cycle time from the function F (equation (3)) taking into account the four linear constraints above. The input size of the problem is the length of a binary representation of the problem data.

In fact, we have to deal with a real time decision problem (Filip, 2005, 2007).

The three binary coefficients defined in the previous section can be considered of two types: fixed and variable. The assignment coefficient is known and given at the beginning of the process. The state coefficient and the decision coefficient are both known only during the disassembly process. They take real time values.

Therefore, the optimisation problem is reduced to a decision problem for which the validity of the solution can be checked in time that is polynomial in the size of the input.

For computing, a *branch-and-cut algorithm* is run. The branch and cut algorithm combines the branch and bound and the cutting plane methods. Branch-and-bound algorithm builds a search tree and maintains a list of sub-problems of the linear problem relaxation that still need to be considered. The idea is to develop better upper bounds on the integer program until an optimal solution is determined. *A cutting plane* is a linear constraint that reduces the space of solution search during the optimisation procedure. (Brucker and Knust, 2006).

Applying this method, in each node of the search tree, separation routines may be called to improve the quality of the linear relaxation in that node. The cutting planes serve to keep the size of the search tree small, while branching may prove useful when cuts are difficult to find or are not effective in improving the current solution (Gueret *et al*, 2002). The steps of the *branch-and-cut* algorithm are the following:

Step 1
Initialize the list of the initial linear programming relaxation

Step 2
Take an instance from the previous list.
If the list is empty, the best known feasible solution is optimal. In the event that no feasible solution has been found, the problem is infeasible

Step 3
Complete pre-processing

Step 4
Use heuristics to try to find an integral solution. If a feasible solution with value less than the upper bound is found, then update the value of the upper bound

Step 5
Solve the linear problem

Step 6
If the problem is infeasible than fathom the node and return to *Step 2*
If the solution value is greater than or equal to the upper bound, fathom the node and return to *Step 2*
If the solution is integral, update the upper bound (the value of the cycle time), fathom the node and return to *Step 2*

Step 7
Attempt a better solution in respect of the input criteria. If successful, return to *Step 3*

Step 8
Generate cutting planes. If successful, add cutting planes to the linear problem and return to *Step 5*

Step 9
Branch. Append the resulting sub-problems to the list of the linear programming relaxation and return to *Step 2*.

3.2. Implementation

The previous algorithm is used in a particular case, as the objective function is not a linear one. The proposed solving method is to apply integer programming on a quadratic function with linear constraints.

Quadratic Programming is the name given to the problem of finding the minimum (or maximum) of a quadratic function of the decision variables subject to linear equality/inequality constraints.

Mixed Integer Quadratic Programming is a quadratic programming method in which the decision variables take discrete values. In other words, is an integer programming method applied on a quadratic objective function (Pangborn, 2002)

To run simulations, the XPRESS-MP software was utilised. This is a linear and integer programming optimiser which has been programmed to handle a broad range of optimisation problems. His quadratic module allows the optimisation of a quadratic function. The main advantage of this software is that the user works in the Console Mode and he can modify the code of the program to suit the data of the problem.

The XPRESS optimiser uses Branch and Bound technique to solve mixed integer programming problems. The relaxed problem is a linear programming problem and can be solved by exploring the tree of solutions using the *cut-off* value method. When a better value of the solution is found in a solution node, this can act as a *cut-off* for outstanding nodes (Dash, 2007). Simulations performed with XPRESS optimizer give the number of iterations and cuts made in the solution space.

4. TWO CASE STUDIES

The mathematical model obtained in the section 2 is used to simulate the disassembly line balancing for two types of products: a unit of voice recognition (Kizilkaya and Gupta, 2006), and a cell phone (Gupta *et a*l, 2004). The computing times and the results are compared.

For the first product, operational disassembly times are given in the Table 2 and the precedence graph in the figure 1. The disassembly line has five workstations.

The second product is a Samsung SCH-3500 cell phone which data is presented in (Gupta *et al*, 2004). Operational disassembly times are given in the Table 3 and the precedence graph in the figure 2. In this case the disassembly line is served by nine workstations.

The complete assembly of the voice recognition unit consists of 18 components including the pouch and the belt subassembly (Kizilkaya and Gupta, 2006).

Table 2 - Parts of the voice recognition unit

Part No	Part	t_j (s)	$t_j^{'}$ (s)
1	Pouch	15	13
2	Belt	15	14
3	Antenna	17	18
4	Lower Audio Cable	20	18
5	Battery	14	12
6	Lower Body Casing	5	6
7	Upper Body Casing	5	7
8	Battery Holder	17	16
9	Battery Release	21	19
10	Flex Rubber	18	16
11	Body Casing Screws	20	21
12	Battery Screws	14	13
13	Battery Springs	12	10
14	Power Button	17	16
15	Power Button Screws	14	15
16	Power Button Plate	13	14
17	PC Board Assembly	17	16
18	Cisco RF Card	15	17

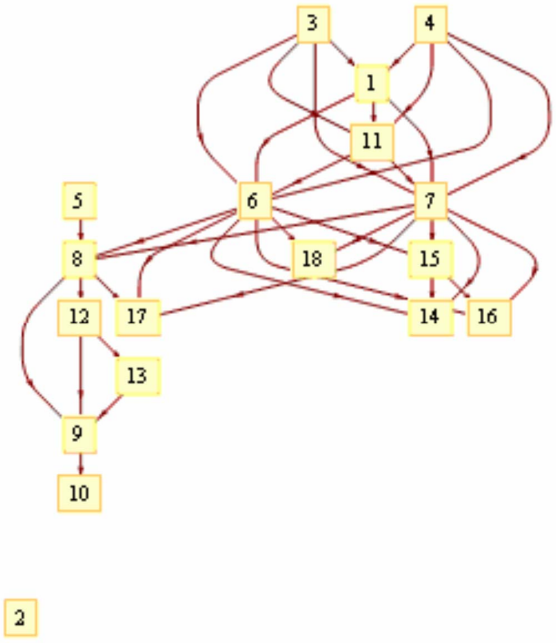

Fig.1. Precedence graph of the voice recognition unit

To perform simulation, operational dismantling times $t_j^{'}$ are also needed. The program decides in real time which operation is made on which workstation and in what manner so as to obtain the balance of the line.

The precedence graphs were obtained from text files which contain the precedence relations between tasks and drawn with the help of Mathematica Software.

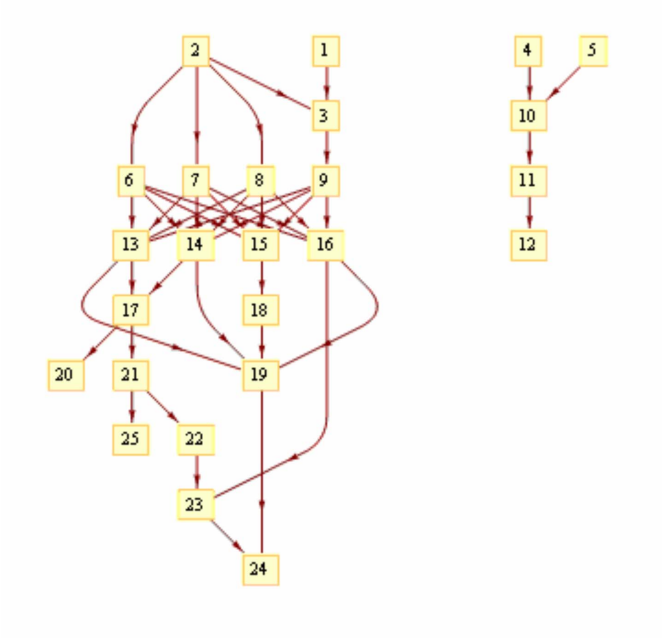

Fig.2. Precedence graph of the cell phone

In the case of the cell phone there are 25 subassemblies. The disassembly is supposed to be complete. The operational times are experimentally determined.

Table 3 - Parts of the cell phone

Part No	Part	t_j (s)	$t_j^{'}$ (s)
1	Antenna	3	2
2	Battery	2	1
3	Antenna Guide path	3	4
4	Bolt A	10	11
5	Bolt B	10	12
6	Bolt 1	15	14
7	Bolt 2	15	16
8	Bolt 3	15	12
9	Bolt 4	15	16
10	Clip	2	3
11	Rubber Seal	2	4
12	Speaker	2	2
13	White Cable	2	2
14	Red/Blue Cable	2	1
15	Orange Cable	2	3
16	Metal Top	2	3
17	Front Cover	2	1
18	Back Cover	3	2
19	Circuit Board	18	15
20	Plastic Screen	5	7
21	Keyboard	1	3
22	LCD	5	2
23	Sub-keyboard	15	14
24	Internal IC Board	2	1
25	Microphone	2	1

5. EXPERIMENTAL RESULTS

The program was run on an AMD Athlon 64 computer, at 2 GHz and 512 RAM.

Two cases are considered when performing the simulation: the static case which is before the beginning of the disassembly process and the dynamic case - during the development of the disassembly tasks. The program also gives the type of disassembly operation: destructive (d) or not (nd).

For the first product with 18 subassemblies the results are presented as it follows. Assignment of the tasks before starting the disassembly process is:

```
Workstation 1: 2d 3nd 4d (duration: 49)
Workstation 2: 1d 5d 11nd 6nd (duration: 50)
Workstation 3: 7nd 8d 15nd 14d (duration: 51)
Workstation 4: 12d 13d 16nd 18nd (duration: 51)
Workstation 5: 9d 10d 17d (duration: 51)
Minimum cycle time: 51
```

Assuming that the tasks 1, 2, 3, 4, 5, 11, 12 are already done, the rebalancing of the line at the moment that is made, looks like it follows:

```
Workstation 1: (duration: 0)
Workstation 2: 6nd 7nd 13d 15nd (duration: 34)
Workstation 3: 14d 16nd (duration: 29)
Workstation 4: 9d 18nd (duration: 34)
Workstation 5: 10d 17d (duration: 32)
Minimum cycle time: 34
```

The computing time was 0.1s in both cases. The branch and cut algorithm generated 27 iterations with 1436 cuts in the plan of solutions.

For the second product with 25 components the results are the following (static and dynamic case respectively):

```
Workstation 1: 1d 2d 5nd (duration: 13)
Workstation 2: 6d (duration: 14)
Workstation 3: 3nd 8d (duration: 15)
Workstation 4: 7nd (duration: 15)
Workstation 5: 9nd 14d (duration: 16)
Workstation 6: 13d 15nd 16nd 17d 18d 20nd 21nd 22d
(duration: 16)
Workstation 7: 19d (duration: 15)
Workstation 8: 4nd 10nd 11nd 12d (duration: 16)
Workstation 9: 23d 24d 25d (duration: 16)
Minimum cycle time: 16
```

Other results are obtained by a simulation made when the tasks 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 are already accomplished:

```
Workstation 1: (duration: 0)
Workstation 2: (duration: 0)
Workstation 3: (duration: 0)
Workstation 4: (duration: 0)
Workstation 5: (duration: 0)
Workstation 6: 15nd 16nd 13d 14d 18d (duration: 8)
Workstation 7: 19d (duration: 15)
Workstation 8: 17d 20nd 21nd 22d 25d (duration: 10)
Workstation 9: 23d 24d (duration: 15)
Minimum cycle time: 15
```

For the second example the computing time was of 4.3s after 54 iterations and 12684 cuts.

6. CONCLUSIONS

A new model to accomplish the real time balancing of a disassembly line is presented in this article. The method is based on the branch-and-cut algorithm and Mixed Integer Quadratic Programming. The results of simulations on two end-of-life manufactured products show that the computational time is very good for products with less then 50 subassemblies. Moreover, the method takes into consideration the disassembly manner: destructive or not.

The obtained values of the cycle time and the balancing of the line using Mixed Integer Quadratic Programming are better than in the case of applying the Greedy algorithm, the heuristic research or the evolutionary algorithms. (Kizilkaya and Gupta, 2006), (Gupta *et al,* 2004), (McGovern and Gupta, 2007).

Further work will be concentrated on the revenues maximization of the end-of-life components in accordance with the components demand on a balanced disassembly line.

REFERENCES

Brucker P., Knust S. (2006) Complex Scheduling, Springer-Verlag Berlin

Dash (2007), and Associates Inc, Xpress Optimizer Reference Manual, Dash Optimization Ltd., Canada

Duta, L., Filip, G. F., Henrioud J. M. (2005) Applying equal piles approach to disassembly line balancing problem, Proceedings of the 16th IFAC World Congress, Prague, July 2005, on CD

Duta L., Henrioud J.M., Caciula I, (2007) A Real Time Solution to Control Disassembly Processes, Proceedings of the 4th IFAC Conference on Management and Control of Production and Logistics, MCPL ’07 , Sibiu, September 2007

Filip, F.G. (2005). Decizie asistata de calculator (Computer Aided Decision), Editura Tehnica, Bucuresti,

Filip F. G. (2007). Sisteme support pentru decizii (Decision Support Systems), Editura Tehnica, Bucuresti,

Gueret C, Prins C., Sevaux M. (2000) Programmation Linéaire, Ed. Eyrolles, Paris.

Gungor A, Gupta S (1999) A Systematic Solution Approach to the Disassembly Line Balancing Problem, Proceedings of the 25th International Conference on Computers and Industrial Engineering, pp. 70-73, 1999

Gupta S. M., Erbis E., McGovern S.M. (2004) Disassembly Sequencing Problem: A case study of a cell phone, Proceedings of SPIE Vol. 5583, pp 43-52

Kizilkaya E.A., Gupta S. M, (2006) Dynamic Kanban System for Disassembly Line Applied to an Industrial Voice Recognition Client Unit, Proceedings of SPIE, Vol 6385, pp 638503-1, 638503-8

McGovern S.M., Gupta S.M., (2007) A balancing method and a genetic algorithm for disassembly line balancing, European Journal of Operational Research, Vol. 179, issue 3, pp 692-708, June 2007

Pangborn G. (2002) A Branch-and-Cut-and-Price implementation for airline crew scheduling, Thesis, Cornell University

Nof S. (1997) Industrial Assembly, Chapman&Hall, 1997.

Rekiek B, Delchmbre (2006) Assembly Line Design, Springer-Verlag London, 2006

XPRESS-MP Release Version 2007, license available at http://www.dashoptimization.com/