7th IFAC Conference on Manufacturing Modelling, Management, and Control
International Federation of Automatic Control
June 19-21, 2013. Saint Petersburg, Russia

Simulated Annealing Algorithm for Solving Sequence-Dependent Disassembly Line Balancing Problem

Can B. Kalayci*. Surendra M. Gupta.**

** Department of Industrial Engineering, Pamukkale University, Denizli, 20070 TURKEY (Tel: +90-544-6176177; e-mail: cbkalayci@pau.edu.tr).*
*** Mechanical and Industrial Engineering Department, 334 Snell Engineering Building, Northeastern University, 360 Huntington Avenue, Boston, MA 02115, USA (Tel: +1-617-373-4846; e-mail:gupta@neu.edu)*

Abstract: In this paper, we consider a sequence-dependent disassembly line balancing problem (SDDLBP) with multiple objectives that concerns with the assignment of disassembly tasks to a set of ordered disassembly workstations while satisfying the disassembly precedence constraints and optimizing the effectiveness of several measures considering sequence-dependent time increments among disassembly tasks. Due to the high complexity of the SDDLBP, there is currently no known way to optimally solve even moderately sized instances of the problem; therefore an efficient methodology based on the simulated annealing is proposed to solve the SDDLBP.

Keywords: Product recovery, disassembly, sequence-dependent disassembly line balancing, metaheuristics, simulated annealing

1. INTRODUCTION

Product recovery seeks to obtain materials and parts from old or outdated products through recycling, refurbishing and remanufacturing in order to minimize the amount of waste sent to landfills. See Gungor and Gupta (1999) and Ilgin and Gupta (2010) for an extensive review of product recovery. The first crucial and the most time consuming step of product recovery is disassembly. Disassembly operations can be performed at a single workstation, in a disassembly cell or on a disassembly line. Although a single workstation and disassembly cell are more flexible, the highest productivity rate is provided by a disassembly line and hence is the best choice for automated disassembly processes, a feature that will be essential in the future disassembly systems (Gungor and Gupta, 2002). Disassembly Line Balancing Problem (DLBP) is a multi-objective problem that is described by Gungor and Gupta (2002) and has mathematically been proven to be NP-complete by McGovern and Gupta (2007) making the goal to achieve the optimal balance computationally expensive. Exhaustive search works well enough in obtaining optimal solutions for small sized instances; however its exponential time complexity limits its application on the large sized instances. An efficient search method needs to be employed to attain a (near) optimal condition with respect to objective functions. Although some researchers have formulated the DLBP using mathematical programming techniques, it quickly becomes unsolvable for a practical sized problem due to its combinatorial nature. For this reason, there is an increasing need to use metaheuristic techniques such as genetic algorithms (GA) (Kalayci and Gupta, 2011a, McGovern and Gupta, 2007), ant colony optimization (ACO) (Kalayci and Gupta, 2012a, McGovern and Gupta, 2005), simulated annealing (SA) (Kalayci et al., 2012), tabu search (TS) (Kalayci and Gupta, 2011b), artificial bee colony (ABC) (Kalayci et al., 2011) and particle swarm optimization (PSO) (Kalayci and Gupta, 2012b). See McGovern and Gupta (2011) for more information on DLBP. Sequence-dependency concept is introduced by (Scholl et al., 2006) to assembly line balancing literature while it was adapted to disassembly by (Kalayci and Gupta, 2012a). Since the sequence-dependent disassembly line balancing problem (SDDLBP) is NP-complete, it cannot be solved in polynomial time in any known way. NP-complete or NP-hard expression represents a way of showing that certain classes of problems are not solvable in realistic time (Tovey, 2002). For this reason, efficient methodologies are necessary to reach near optimal solutions of SDDLBP. SA was selected as a solution approach for solving SDDLBP.

2. NOTATION

c	Cycle time (Maximum time available at each workstation)
d_i	Demand; quantity of part i requested
F_0	Fitness values vector of initial solution
F_{best}	Fitness values vector of best solution
F_c	Fitness values vector of current solution
F_g	Fitness values vector at iteration g
g	Current iteration (generation) number of the algorithm
h_i	Binary value; 1 if part i is hazardous, else 0.
IP	Set (i,j) of parts such that task i must precede task j
i	Part identification, task count $(1,\ldots,n)$
j	Part identification, task count $(1,\ldots,n)$
k	Workstation count $(1,\ldots,m)$
m	Number of workstations required for a given

	solution sequence
m^*	Minimum possible number of workstations
n	Number of parts for removal
N	The set of natural numbers
ps	Population size
PS_i	i^{th} part in a solution sequence
r	Uniformly distributed random number between 0 and 1.
S_0	Initial solution
S_c	Current solution
S_{best}	Best solution
S_g	Solution at iteration g
sd_{ij}	Sequence dependent time increment influence of i on j
ST_j	station time; total processing time requirement in workstation j
T_0	Initial temperature
T_g	Temperature at iteration g
t_i	Part removal time of part i
t_i'	Part removal time of part i considering sequence dependent time increment
$tlimit$	Time limit of the algorithm to be executed

3. PROBLEM DEFINITION AND FORMULATION

The sequence dependent disassembly line balancing problem (SDDLBP) investigated in this paper is concerned with a paced disassembly line for a single model of product that undergoes complete disassembly. As opposed to the DLBP, in SDDLBP whenever a task interacts with another task, their task times may be influenced. Disassembling a particular component before another component may prolong (or curtail) the task time, as opposed to disassembling them in reverse order, because one component could hinder the other because it requires additional movements and/or prevents it from using the most efficient disassembly process. If task j is performed before task i, its standard time t_j is incremented by sd_{ij}. This sequence dependent increment measures the prolongation of task j forced by the interference of already waiting task i.

Illustrative example: The precedence relationships (solid line arrows) and sequence dependent time increments (dashed line arrows) for an 8 part PC disassembly process are illustrated in Fig. 1 and their knowledge database is given in Table 1. This example is modified from (Gungor and Gupta, 2002).

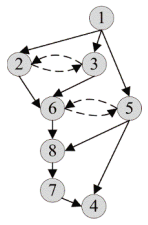

Fig 1. Precedence relationships (solid line arrows) and sequence dependent time increments (dashed line arrows) for the PC example

Table 1: Knowledge database for the PC example

Part	Task	Time	Hazardous	Demand
PC top cover	1	14	No	360
Floppy drive	2	10	No	500
Hard drive	3	12	No	620
Back plane	4	18	No	480
PCI cards	5	23	No	540
RAM modules	6	16	No	750
Power supply	7	20	No	295
Motherboard	8	36	No	720

Sequence dependencies for the PC example are given as follows: $sd_{23}=2, sd_{32}=4, sd_{56}=1, sd_{65}=3$. For a feasible sequence $\langle 1,2,3,6,5,8,7,4\rangle$; since part 2 is disassembled before part 3, sequence dependency $sd_{32}=4$ takes place because when part 2 is disassembled, the obstructing part 3 is still not taken out, i.e., the part removal time for part 2 is increased which results in $t_2'=t_2+sd_{32}=14$; similarly since part 6 is disassembled before part 5, sequence dependency $sd_{56}=1$ takes place because when part 6 is disassembled, the obstructing part 5 is still not taken out, i.e., the part removal time for part 6 is increased which results in $t_6'=t_6+sd_{56}=17$. In this paper, the precedence relationships considered are of AND type and are represented using the immediately preceding matrix $[y_{ij}]_{n\times n}$, where

$$y_{ij}=\begin{cases}1 & \text{if task } i \text{ is executed after task } j\\ 0 & \text{if task } i \text{ is executed before task } j\end{cases} \quad (1)$$

In order to state the partition of total tasks, we use the assignment matrix $[x_{jk}]_{n\times m}$, where

$$x_{jk}=\begin{cases}1 & \text{if part } j \text{ is assigned to station } k\\ 0 & \text{otherwise}\end{cases} \quad (2)$$

The matrix $\left[sd_{ij}\right]_{n\times n}$ holds the sequence-dependent time increments data:

$$sd_{ij}=\begin{cases}sd_{ij} & \text{if part } i \text{ prolongs removal time of part } j\\ 0 & \text{otherwise}\end{cases} \quad (3)$$

The mathematical formulation of SDDLBP is given as follows:

$$\min f_1 = m \quad (4)$$

$$\min f_2 = \sum_{j=1}^{m}\left(c-ST_j\right)^2 \quad (5)$$

$$\min f_3 = \sum_{i=1}^{n} i\times h_{PS_i}, \quad h_{PS_i}=\begin{cases}1 & \text{hazardous}\\ 0 & \text{otherwise}\end{cases} \quad (6)$$

$$\min f_4 = \sum_{i=1}^{n} i\times d_{PS_i}, \quad d_{PS_i}\in N, \forall PS_i \quad (7)$$

Subject to:

$$\sum_{k=1}^{m} x_{jk}=1, \quad j=1,\ldots,n \quad (8)$$

$$\left\lceil \frac{\sum_{i=1}^{n} t_i}{c} \right\rceil \le m \le n \quad (9)$$

$$\sum_{j=1}^{n} \left(t_j + \sum_{i=1}^{n} sd_{ij} \times y_{ij} \right) \times x_{jk} \le c \quad (10)$$

$$x_{ik} \le \sum_{k=1}^{m} x_{jk}, \quad \forall (i,j) \in IP \quad (11)$$

The first objective given in equation (4) is to minimize the number of workstations for a given cycle time (the maximum time available at each workstation). The second objective given in equation (5) is to aggressively ensure that idle times at each workstation are similar, though at the expense of the generation of a non-linear objective function (McGovern and Gupta, 2007). As the third objective (see equation (6)), a hazard measure developed to quantify each solution sequence's performance, with a lower calculated value being more desirable (McGovern and Gupta, 2007). This measure is based on binary variables that indicate whether a part is considered to contain hazardous material (the binary variable is equal to 1 if the part is hazardous, else 0) and its position in the sequence. A given solution sequence hazard measure is defined as the sum of hazard binary flags multiplied by their position number in the solution sequence, thereby rewarding the removal of hazardous parts early in the part removal sequence. As the fourth objective (equation (7)), a demand measure was developed to quantify each solution sequence's performance, with a lower calculated value being more desirable (McGovern and Gupta, 2007). This measure is based on positive integer values that indicate the quantity required of a given part after it is removed (or 0 if it is not desired) and its position in the sequence. A solution sequence demand measure is then defined as the sum of the demand value multiplied by the position of the part in the sequence, thereby rewarding the removal of high demand parts early in the part removal sequence. The constraints given in; equation (8) ensures that all tasks are assigned to at least and at most one workstation (the complete assignment of each task), equation (9) guarantees that the number of work stations with a workload does not exceed the permitted number, equation (10) ensures that the work content of a workstation cannot exceed the cycle time and equation (11) imposes the restriction that all the disassembly precedence relationships between tasks should be satisfied.

4. PROPOSED SIMULATED ANNEALING ALGORITHM

DLBP was proven to be NP-complete (McGovern and Gupta, 2007). Since SDDLBP is a generalization of DLBP (setting all sequence dependent time increments to zero, SDDLBP reduces to DLBP), SDDLBP is NP-complete, too. Since SDDLBP falls into the NP-Complete class of combinatorial optimization problems, when the problem size increases, the solution space is exponentially increased and an optimal solution in polynomial time cannot be found as it can be time consuming for optimum seeking methods to obtain an optimal solution within this vast search space. Therefore, it is necessary to use alternative methods in order to reach (near) optimal solutions faster. In this regard, nature has inspired many heuristic algorithms to obtain reasonable solutions to complex problems. For this reason, a fast and effective simulated annealing based solution approach is proposed to solve SDDLBP. Simulated annealing is an iterative random search technique that has been applied to many optimization problems in a wide variety of areas, including the assembly line balancing (ALBP). Simulated annealing is a stochastic approach used for solving many combinatorial optimization problems by inspiration from the physical annealing process of metals. Simulated Annealing gets its name from the physical annealing of solid that is heated to a very high temperature and then cooled at a slow rate, spending a relatively large amount of time near the freezing point of the solid.

The proposed approach starts with an initial solution that is defined as the current solution. Then, a neighbour solution is obtained from the current solution. The cost of the neighbour solution is calculated and compared with the cost of the current solution. If the objective function value is inferior to that of the current solution, the neighbouring solution becomes the new current solution. If the neighbouring solution provides an objective function value superior to that of the current solution, the neighbouring solution may still become the current solution with a probability if a certain acceptance criterion is met. Otherwise the current solution remains unchanged. A distinctive feature of Simulated Annealing is that inferior solutions are sometimes accepted as the current solution to try and so, prevent getting trapped at local optima.

A configuration is a solution to a given problem. In the proposed SA algorithm, elements of the solution string are integers. Each element represents a task assignment to work station. The strategy of building a feasible balancing solution is the key issue to solve the SDDLBP. We use station-oriented procedure for a solution constructing strategy in which solutions are generated by filling workstations successively one after the other (Ding et al., 2010). The procedure is initiated by the opening of a first station. Then, tasks are successively assigned to this station until more tasks cannot be assigned and a new station is opened. In each iteration, a task is randomly chosen from the set of candidate tasks to assign to the current station. When no more tasks may be assigned to the open station, this is closed and the following station is opened. The procedure finalizes when there are no more tasks left to assign. A new solution obtained from a current solution by using a specific move is called a neighbourhood solution. In the proposed SA algorithm, interchanging two tasks (SWAP) or inserting a task to a different work station (INSERT) is implemented as a moving strategy such that the new neighbouring solutions are ensured to be feasible. By guaranteeing feasibility in each operation, the necessity of the repair function is prevented. In SWAP, two randomly selected tasks from two randomly selected workstations are exchanged and in INSERT, a randomly selected task from a randomly selected workstation is inserted into another randomly selected workstation while satisfying the precedence constraints. Examples for SWAP and INSERT operators are given in Fig. 2 and Fig. 3,

respectively. Flow diagram of the proposed approach is given at Fig. 4.

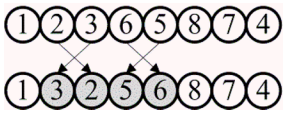

Fig 2. SWAP operation

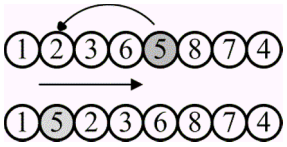

Fig 3. INSERT operation

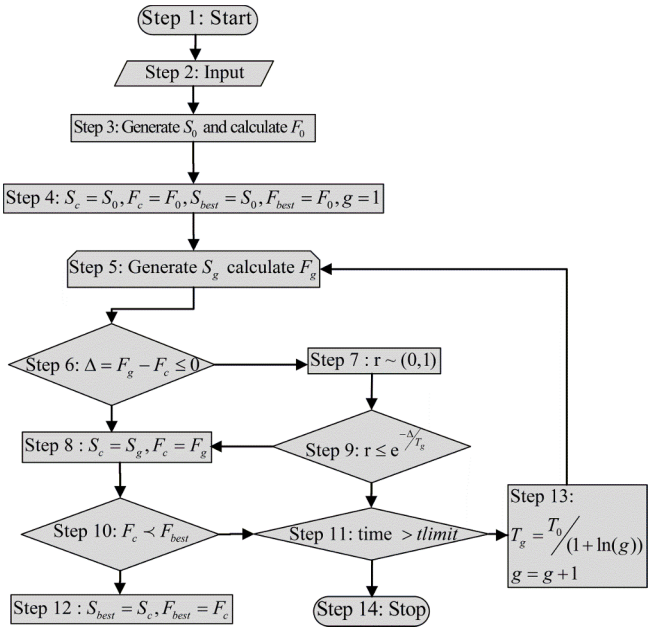

Fig 4. Flow diagram of the proposed SA approach

The steps of the proposed approach are given below:

Step 1: Start.

Step 2: Read disassembly data, initialize parameters, construct vectors and matrices and start iteration

Step 3: Generate initial solution (S_0) by selecting the best solution found by heuristics such as Greatest Ranked Positional Weight, Greatest Number of Successors, Longest Processing Time, Smallest Task Number and Smallest Upper Bound, repair generated solutions if necessary, calculate the fitness values of the initial solution and set to F_0 vector.

Step 4: Set initial solution (S_0) to current S_c and best solution S_{best}, initial fitness values vector F_0 to current F_c and best fitness value vectors F_{best} and set iteration g to 1.

Step 5: Generate a neighbour solution S_g by applying INSERT or SWAP with the probability of .5, repair the solution found if necessary and calculate its fitness values and set to F_g vector.

Step 6: Checking fitness values f_1, f_2, f_3, f_4 according to the priorities defined, if the fitness values in F_g vector of the solution S_g is less than or equal to the fitness values in F_c vector of the current solution S_c, go to step 8.

Step 7: Generate a uniform random number *r* and go to Step 9.

Step 8: Accept the neighbour solution found as current solution. Set solution S_g to current solution S_c and F_g to current fitness values vector F_c. Go to Step 10.

Step 9: If $r \le e^{(-\Delta/T_i)}$ where $\Delta = F_g - F_c$ go to Step 8, otherwise go to Step 11. Please note that Δ value is calculated considering the second objective (f_2) only, since this objective not only dominates the first objective (f_1), but also there is a priority of objectives in the following order: f_1, f_2, f_3, f_4. This means that second objective controls the algorithm.

Step 10: Checking fitness values f_1, f_2, f_3, f_4 according to the priorities defined, if the fitness values in F_c vector of the solution S_c is less than the fitness values in vector of the current solution S_{best}, go to step 12.

Step 11: If time limit is exceeded, go to Step 14, otherwise go to Step 13.

Step 12: Accept the neighbour solution found as the best solution. Set current solution S_c to best solution S_{best} found so far and F_c to best fitness values vector F_{best}.

Step 13: Adjust the cooling schedule as given in the following:

$$T_g = T_0/(1+\ln(g)) \tag{12}$$

And increase iteration number, go to Step 5.

Step 14: Stop.

Thus, SA algorithm decides whether a solution is better than another or not by comparing the objective values of each solution one by one. Since the objective values has infinite priorities with a goal programming perspective, first, second, third and fourth objectives are compared respectively from one to another. Once a solution with a lower objective value for the objective function that has higher priority is obtained, the new solution is accepted as a better solution since the new solution is superior to the current solution. Please note that the second objective is significant in terms of algorithm performance.

5. NUMERICAL RESULTS

The proposed algorithm was coded in MATLAB and tested on Intel Core2 1.79 GHz processor with 3GB RAM. The

software is investigated on two different scenarios.. A full factorial set of experiments was conducted to find the best T_0 which was found to be 2000 ($T_0 = 2000$). The first scenario is for a product consisting of n=10 components. The knowledge database and precedence relationships for the components are given in Table 2 and Fig. 5, respectively. The problem and its data were modified from (McGovern and Gupta, 2006) with a paced disassembly line operating at a speed which allows c=40 s for each workstation to perform its required disassembly tasks. The sequence dependencies for the 10 part product are given as: $sd_{14} = 1,\ sd_{23} = 2,\ sd_{32} = 3,\ sd_{41} = 4,\ sd_{45} = 4,\ sd_{54} = 2,\ sd_{56} = 2,\ sd_{65} = 4,\ sd_{69} = 3,\ sd_{96} = 1$. While the exhaustive search method was able to find optimal solution in 215t time on average, the proposed approach was able to successfully find the optimal solution ($f_1 = 5, f_2 = 67, f_3 = 5, f_4 = 9605$) in less than t time on average under the restriction of the system specifications given above. Table 3 depicts an optimal solution sequence.

Table 2: Knowledge database for the 10 part product

Task	Time	Hazardous	Demand
1	14	No	0
2	10	No	500
3	12	No	0
4	17	No	0
5	23	No	0
6	14	No	750
7	19	Yes	295
8	36	No	0
9	14	No	360
10	10	No	0

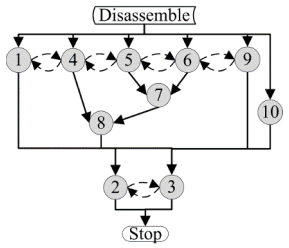

Fig. 5: Precedence relationships (solid line arrows) and sequence dependent time increments (dashed line arrows) for the 10 part product

Table 3: An optimal solution sequence for 10 Part product disassembly

Part removal sequence ↓	Workstations I	II	III	IV	V	Time to remove parts (in seconds)
6	17					
1	18					
10		10				
5		27				
7			19			
4			17			
8				36		
9					14	
2					13	
3					12	

The second scenario consists of a cellular telephone instance with n=25 components. The knowledge database and precedence relationships for the components are given in Table 4 and Fig. 6, respectively. The problem and its data were modified from Gupta et al. (2004) with a disassembly line operating at a speed which allows c=18 for each workstation to perform its required disassembly tasks. The sequence dependencies for the 25 part product are given as the follows: $sd_{45} = 2,\ sd_{54} = 1,\ sd_{67} = 1,\ sd_{69} = 2,\ sd_{76} = 2,\ sd_{78} = 1,\ sd_{87} = 2,\ sd_{96} = 1,\ sd_{13,14} = 1,\ sd_{14,13} = 2,\ sd_{14,15} = 2,\ sd_{15,14} = 1,\ sd_{20,21} = 1,\ sd_{21,20} = 2,\ sd_{22,25} = 1,\ sd_{25,22} = 2$

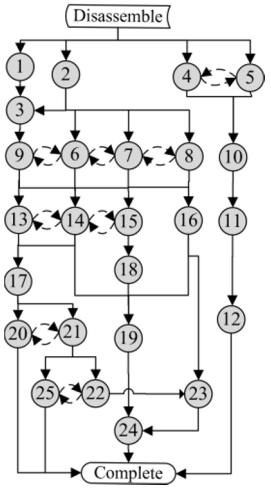

Fig. 6: Precedence relationships (solid line arrows) and sequence dependent time increments (dashed line arrows) for the 25 part product

Since within the vast search space (25!), the exhaustive search is prohibitive due to the exponential growth of the time complexity, i.e., the optimal solution is unknown. The proposed SA algorithm was able to find the best solution given in Fig. 7. It took less than 500t (*tlimit*) time to search for this solution under the restriction of the system specifications given above.

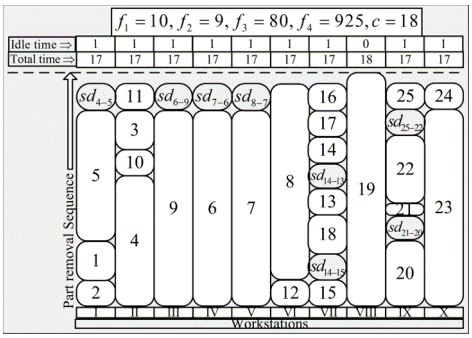

Fig 7. A typical solution found using the cellular telephone instance

Table 4: Knowledge database for 25 part product disassembly

Part	Task	Part Removal Time	Hazardous	Demand
Antenna	1	3	Yes	4
Battery	2	2	Yes	7
Antenna guide	3	3	No	1
Bolt (type 1) A	4	10	No	1
Bolt (Type1) B	5	10	No	1
Bolt (Type2) 1	6	15	No	1
Bolt (Type2) 2	7	15	No	1
Bolt (Type2) 3	8	15	No	1
Bolt (Type2) 4	9	15	No	1
Clip	10	2	No	2
Rubber Seal	11	2	No	1
Speaker	12	2	Yes	4
White Cable	13	2	No	1
Red/Blue Cable	14	2	No	1
Orange Cable	15	2	No	1
Metal Top	16	2	No	1
Front Cover	17	2	No	2
Back Cover	18	3	No	2
Circuit Board	19	18	Yes	8
Plastic Screen	20	5	No	1
Keyboard	21	1	No	4
LCD	22	5	No	6
Sub-keyboard	23	15	Yes	7
Internal IC Board	24	2	No	1
Microphone	25	2	Yes	4

6. CONCLUSIONS

SDDLBP is a recently reported multi-objective NP-complete optimization problem. The main objective of this chapter was to solve sequence-dependent disassembly line balancing problem (SDDLBP) which aimed to minimize the number of disassembly workstations, minimize the total idle time of all workstations by ensuring similar idle time at each workstation considering sequence dependent time increments, maximize the removal of hazardous components as early as possible in the disassembly sequence and maximize the removal of high demand components before low demand components. A fast, near-optimal, simulated annealing approach was modified, developed and presented in this chapter to solve multi-objective SDDLBP.

REFERENCES

Ding, L.-P., Feng, Y.-X., Tan, J.-R. & Gao, Y.-C. (2010) A new multi-objective ant colony algorithm for solving the disassembly line balancing problem. *The International Journal of Advanced Manufacturing Technology,* 48**,** 761-771.

Gungor, A. & Gupta, S. M. (1999) Issues in environmentally conscious manufacturing and product recovery: a survey. *Computers & Industrial Engineering,* 36**,** 811-853.

Gungor, A. & Gupta, S. M. (2002) Disassembly line in product recovery. *International Journal of Production Research,* 40**,** 2569-2589.

Gupta, S. M., Erbis, E. & McGovern, S. M. (2004) Disassembly sequencing problem: A case study of a cell phone. IN GUPTA, S. M. (Ed.) *Environmentally Conscious Manufacturing IV.* Bellingham, SPIE-International Society for Optical Engineering.

Ilgin, M. A. & Gupta, S. M. (2010) Environmentally conscious manufacturing and product recovery (ECMPRO): A review of the state of the art. *Journal of Environmental Management,* 91**,** 563-91.

Kalayci, C. B. & Gupta, S. M. (2011a) A hybrid genetic algorithm approach for disassembly line balancing. *Proceedings of the 42nd Annual Meeting of Decision Science Institute (DSI 2011).* Boston, MA, USA.

Kalayci, C. B. & Gupta, S. M. (2011b) Tabu search for disassembly line balancing with multiple objectives. *41st International Conference on Computers and Industrial Engineering (CIE41).* University of Southern California, Los Angeles, USA.

Kalayci, C. B. & Gupta, S. M. (2012a) Ant colony optimization for sequence-dependent disassembly line balancing problem. *Journal of Manufacturing Technology Management,* In press.

Kalayci, C. B. & Gupta, S. M. (2012b) A particle swarm optimization algorithm for solving disassembly line balancing problem. *Proceedings of Northeast Decision Sciences Institute 2012 Annual Conference.* Newport, Rhode Island, USA.

Kalayci, C. B., Gupta, S. M. & Nakashima, K. (2011) Bees Colony Intelligence in Solving Disassembly Line Balancing Problem. *Proceedings of the 2011 Asian Conference of Management Science and Applications (ACMSA2011).* Sanya, Hainan, China.

Kalayci, C. B., Gupta, S. M. & Nakashima, K. (2012) A Simulated Annealing Algorithm for Balancing a Disassembly Line. IN MATSUMOTO, M., UMEDA, Y., MASUI, K. & FUKUSHIGE, S. (Eds.) *Design for Innovative Value Towards a Sustainable Society.* Springer Netherlands.

McGovern, S. M. & Gupta, S. M. (2005) Ant colony optimization for disassembly sequencing with multiple objectives. *The International Journal of Advanced Manufacturing Technology,* 30**,** 481-496.

McGovern, S. M. & Gupta, S. M. (2006) Ant colony optimization for disassembly sequencing with multiple objectives. *The International Journal of Advanced Manufacturing Technology,* 30**,** 481-496.

McGovern, S. M. & Gupta, S. M. (2007) A balancing method and genetic algorithm for disassembly line balancing. *European Journal of Operational Research,* 179**,** 692-708.

McGovern, S. M. & Gupta, S. M. (2011) *The Disassembly Line: Balancing and Modeling,* New York, McGraw Hill.

Scholl, A., Boysen, N. & Fliedner, M. (2006) The sequence-dependent assembly line balancing problem. *OR Spectrum,* 30**,** 579-609.

Tovey, C. A. (2002) Tutorial on Computational Complexity. *Interfaces,* 32**,** 30-61.