

January 01, 2005

Stochastic and deterministic combinatorial optimization solutions to an electronic product disassembly flow shop

Surendra M. Gupta
Northeastern University

Seamus M. McGovern
Northeastern University

Recommended Citation

Gupta, Surendra M. and McGovern, Seamus M., "Stochastic and deterministic combinatorial optimization solutions to an electronic product disassembly flow shop" (2005). . Paper 107. <http://hdl.handle.net/2047/d10014056>



Laboratory for Responsible Manufacturing

Bibliographic Information

McGovern, S. M. and Gupta, S. M., "Stochastic and Deterministic Combinatorial Optimization Solutions to an Electronic Product Disassembly Flow Shop", ***Proceedings of the 2005 Northeast Decision Sciences Institute Conference***, Philadelphia, Pennsylvania, March 30-April 1, 2005 (CD-ROM).

Copyright Information

Copyright 2005, Surendra M. Gupta.

Contact Information

Dr. Surendra M. Gupta, P.E.
Professor of Mechanical and Industrial Engineering and
Director of Laboratory for Responsible Manufacturing
334 SN, Department of MIE
Northeastern University
360 Huntington Avenue
Boston, MA 02115, U.S.A.

(617)-373-4846 **Phone**
(617)-373-2921 **Fax**
gupta@neu.edu **e-mail address**

<http://www.coe.neu.edu/~smgupta/> **Home Page**

STOCHASTIC AND DETERMINISTIC COMBINATORIAL OPTIMIZATION SOLUTIONS TO AN ELECTRONIC PRODUCT DISASSEMBLY FLOW SHOP

Seamus M. McGovern, Northeastern University, Boston, MA 02115, (617)-494-2054, mcgovern.s@neu.edu

Surendra M. Gupta*, Northeastern University, Boston, MA 02115, (617)-373-4846, gupta@neu.edu

(*Corresponding author)

ABSTRACT

Disassembly takes place in remanufacturing, recycling and disposal, with a flow shop being the best choice for automation. The disassembly line balancing problem seeks a sequence which: is feasible, minimizes workstations, and ensures similar idle times, as well as other end-of-life specific concerns. Finding the optimal balance is computationally intensive due to exponential growth. Combinatorial optimization methods hold promise for providing solutions to the problem, which is proven NP-hard. Stochastic (genetic algorithm) and deterministic (greedy/hill-climbing hybrid heuristic) methods are presented and compared. Numerical results are obtained using a recent electronic product case study.

INTRODUCTION

More and more manufacturers are acting to recycle and remanufacture their post-consumer products due to environmental legislation, public awareness and extended manufacturer responsibility as well as the economic attractiveness of reusing products and parts. The first step of product recovery is disassembly. Disassembly is the methodical extraction of valuable parts and materials from discarded products. After disassembly, reusable components are cleaned, refurbished, tested and directed to inventory for remanufacturing operations. The recyclable materials can be sold to raw-material suppliers, while the residuals are sent to landfills.

This paper first mathematically defines the disassembly line balancing problem (DLBP) then proves for the first time that it is NP-hard, necessitating specialized solution techniques. While exhaustive search consistently provides the problem's optimal solution, its time complexity quickly limits its practicality. Combinatorial optimization is an emerging field that combines techniques from applied mathematics, operations research and computer science to solve optimization problems over discrete structures. These techniques include: greedy algorithms, integer and linear programming, branch-and-bound, divide and conquer, dynamic programming, local optimization, simulated annealing, genetic algorithms, and approximation algorithms. In this paper, the DLBP is solved using a stochastic metaheuristic (genetic algorithm or GA [19]) and a deterministic hybrid process consisting of a greedy sorting algorithm followed by a hill-climbing heuristic (Adjacent Element Hill Climbing; AEHC [16]).

LITERATURE REVIEW

Product recovery involves a number of steps [8]. The first crucial step is disassembly [2] [11]. Gungor and Gupta presented the first introduction to the DLBP [5] [6] [7]. McGovern and Gupta developed a greedy/2-Opt hybrid algorithm [13] and an uninformed general-purpose search heuristic [15] for the DLBP. Gupta and McGovern presented an ant colony optimization solution to the DLBP in [10].

THE DLBP MODEL DESCRIPTION

The following notation are used in the paper:

CT	cycle time; max time at each workstation			bound for the decision version of DLBP
d_k	demand; quantity of part k requested	I		total idle time for a given solution sequence
D	demand rating for a solution; also, demand bound for the decision version of DLBP	j		workstation count ($1, \dots, NWS$)
F	measure of balance for a given solution	k		part identification ($1, \dots, n$)
h_k	binary value; 1 if part k is hazardous, else 0	n		number of part removal tasks
H	hazard rating for a solution; also, hazard	N		the set of natural numbers ($0, 1, 2, \dots$)
		NWS		workstations required for a given solution

PRT_k	part removal time required for k th task	R_k	binary value; 0 if part k can be removed in the same direction as part $k + 1$, else 1
PS_k	k^{th} part in a solution sequence (i.e., for solution “3, 1, 2” $PS_2 = 1$)	ST_j	station time; total processing time requirement in workstation j
r_k	integer corresponding to removal direction	V	maximum range for a workstation’s idle time
R	direction rating for a solution; also, direction bound for the decision version of DLBP		

In DLBP, a solution consists of an ordered sequence of work elements (tasks, components or parts). For example, if a solution consisted of the n -tuple $\langle 5, 2, 8, 1, 4, 7, 6, 3 \rangle$, then part 5 would be removed first, followed by part 2, then part 8, and so on.

The application investigated in this paper seeks to fulfill five objectives:

1. Minimize the number of workstations and hence, minimize the total idle time.
2. Ensure workstation idle times are similar.
3. Remove hazardous parts early in the sequence.
4. Remove high-demand parts before low-demand parts.
5. Minimize the number of direction changes.

A major constraint is the requirement to provide a feasible disassembly sequence for the product being investigated. Testing a given solution against the precedence constraints fulfills the major constraint of precedence preservation. Minimizing the sum of the workstation idle times also minimizes the total number of workstations. This attains objective 1 and is described by:

$$I = \sum_{j=1}^{NWS} (CT - ST_j) \quad (1)$$

Line balancing seeks to achieve *perfect balance* (all idle times equal to zero). When this is not possible, either Line Efficiency (LE) or the Smoothness Index (SI) is often used as a performance evaluation tool [3]. SI rewards similar idle times at each workstation, but at the expense of allowing for a large number of workstations. This is because SI compares workstation elapsed times to the largest ST_j instead of to CT . LE rewards the minimum number of workstations, but allows unlimited variance in idle times between workstations because no comparison is made between ST_j s. This paper makes use of the balancing method developed by McGovern and Gupta [17]. This method simultaneously minimizes the number of workstations while aggressively ensuring that idle times at each workstation are similar, though at the expense of a nonlinear objective function. The method penalizes solutions where, even though the number of workstations may be minimized, one or more have an exorbitant amount of idle time. Therefore, a resulting minimum performance value is the more desirable solution, indicating both a minimum number of workstations and similar idle times. The McGovern-Gupta measure of balance is represented as:

$$F = \sum_{j=1}^{NWS} (CT - ST_j)^2 \quad (2)$$

with perfect balance indicated by $F = 0$. Note that formula (2) effectively makes formula (1) redundant due to the fact that it concurrently minimizes the number of workstations. Hazard and demand measures have been developed and are represented by:

$$H = \sum_{k=1}^n (k \cdot h_{PS_k}), \quad h_{PS_k} = \begin{cases} 1, & \text{hazardous} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

$$D = \sum_{k=1}^n (k \cdot d_{PS_k}), \quad d_{PS_k} \in \mathbb{N}, \forall PS_k \quad (4)$$

McGovern and Gupta [13] provide further explanation of formulae (1) through (4). Part removal directions and the subsequent direction measure are expressed as:

$$r_{PS_k} = \begin{cases} + 1, direction & : + x \\ - 1, direction & : - x \\ + 2, direction & : + y \\ - 2, direction & : - y \\ + 3, direction & : + z \\ - 3, direction & : - z \end{cases} \quad (5)$$

$$R = \sum_{k=1}^{n-1} R_k, R_k = \begin{cases} 1, r_{PS_k} \neq r_{PS_{k+1}} \\ 0, otherwise \end{cases} \quad (6)$$

The overall goal is to minimize each of these measures, with a priority of F , then H , then D , then R . Problem assumptions include the following:

- Part removal times are deterministic, constant, and integer (or able to be converted to integers).
- Each product undergoes complete disassembly.
- All products contain all parts with no additions, deletions or modifications.
- Each task is assigned to exactly one workstation.
- The sum of the part removal times of all parts assigned to a workstation must not exceed CT .
- Part precedence relationships must be enforced.

NP-HARDNESS PROOF

The decision version of DLBP has been shown to be NP-complete [14]. The problem can be proven to be NP-hard in two steps by showing that: 1.) the NP-complete decision problem is no harder than its optimization problem, and 2.) the decision problem is NP-complete [4].

Theorem 1: DLBP is NP-hard.

Proof: 1.) The decision version of DLBP is no harder than its optimization version. Both versions require preservation of the precedence constraints. The optimization version of DLBP asks for a sequence that has the minimum difference between largest and smallest idle times among all disjoint sets, the minimum sum of hazardous part binary values multiplied by their sequence position, the minimum sum of demanded part values multiplied by their sequence position, and the minimum number of part removal direction changes. The decision version includes numerical bounds V , H , D and R as additional parameters and asks whether there exists NWS disjoint sets with a difference between largest and smallest idle times no more than V , a sum of hazardous part binary values multiplied by their sequence position no more than H , a sum of demanded part values multiplied by their sequence position no more than D , and a number of part removal direction changes no more than R . So long as the difference between largest and smallest idle times, the sum of hazardous part binary values multiplied by their sequence position, the sum of demanded part values multiplied by their sequence position, and the number of part removal direction changes is relatively easy to evaluate, the decision problem can be no harder than the corresponding optimization problem. If we could find a minimum difference between the largest and smallest idle times, a minimum sum of hazardous part binary values multiplied by their sequence position, a minimum sum of demanded part values multiplied by their sequence position, and a minimum number of part removal direction changes for the DLBP optimization problem in polynomial time, then we could also solve the associated decision problem in polynomial time. All we need do is find the minimum idle time and the maximum idle time from all NWS subsets, compute their difference, and compare that difference between largest and smallest idle times to the given bound V ; sum all the hazardous part binary values multiplied by their sequence position and compare that sum to the given bound H ; sum all the demanded part values multiplied by their sequence position and compare that sum to the given bound D ; and sum all of the part removal direction changes and compare that sum to the given bound R . Therefore, the decision version of DLBP is no harder than its optimization version.

2.) From [14], the decision version of DLBP is NP-complete. Therefore, from 1.) and 2.), DLBP is NP-hard.

ø

THE GREEDY AND AEHC HEURISTICS

A greedy strategy always makes the choice that looks best at the moment, that is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution. The DLBP Greedy algorithm was built around First-Fit-Decreasing (FFD) rules. Originally developed for the bin-packing problem, FFD looks at each element in a list, from largest to smallest (*PRTs* in the DLBP) and puts that element into the first workstation in which it fits. In DLBP Greedy, each part in two sorted lists (hazardous parts, large *PRT* to small, and remaining parts, large *PRT* to small) is examined. If the part had not previously been put into the solution sequence (as shown by a Tabu-type list), the part is put into the current workstation if idle time remains to accommodate it and as long as that position will not violate any precedence constraints. If no workstation can accommodate it at the given time in the search due to precedence constraints, the part is maintained on the sorted list and the next part (not yet selected) on the sorted list is considered. If all parts have been examined for insertion into the current workstation, a new workstation is created and the process is repeated.

While being very fast and efficient, the algorithm is not always able to minimize the number of workstations. In addition, there is no capability to balance the workstations; in fact, the FFD structure lends itself to filling the earlier workstations as much as possible, often to capacity, while later workstations have progressively greater and greater idle times. This can result in extremely poor balance. This limitation led to the development of AEHC to fulfill the second objective [16].

Hill climbing is an iterated improvement algorithm that makes use of an iterative greedy strategy (move in the direction of increasing value). AEHC is designed to consider swapping each task in every workstation with each task in its subsequent workstation in search of improved balance. It does this while observing precedence constraints and not exceeding *CT* in either workstation. Only adjacent workstations are compared to enable a rapid search and since it is deemed unlikely that parts several workstations apart can be swapped and still preserve the precedence of all of the tasks in-between.

As is the norm with greedy algorithms, the DLBP Greedy process is run once to determine a solution. Hill climbing, however, can be continuously run on subsequent solutions for as long as is deemed acceptable by the user or until it is no longer possible to improve, at which point it is assumed that the (local) optimum has been reached.

THE DLBP GENETIC ALGORITHM

A GA has a solution structure defined as a *chromosome*, which is made up of *genes* (parts in DLBP) and generated by two *parent* chromosomes (each with its own measure of *fitness*) from the *pool* (or *population*) N , of solutions. New solutions are made from old by the methods of *crossover* (sever parent genes and swap severed sections) and *mutation* (randomly vary genes within a chromosome). Initially, a feasible population is generated and the fitness of each chromosome in this *generation* is calculated. An even integer of $R_x N$ parents is randomly selected for crossover to produce $R_x N$ *children* (children make up $100 \times R_x$ percent of each generation's population). Mutation is randomly (based on the R_m value) performed. The $R_x N$ least fit parents are removed and the process is repeated.

DLBP GA was modified from a general-purpose GA in several ways. A major challenge with any GA implementation is determining a chromosome representation that remains valid after each generation; the precedence preservative crossover (PPX) [1] is used here. PPX creates a mask (one for each child, every generation) that consists of random 1s and 2s indicating which parent part information should be taken from to create the child. Also, instead of the worst portion of the population being selected for crossover, all of the population was randomly considered for crossover to enable more diversity. Mutation is randomly performed by selecting a single chromosome and exchanging two of its disassembly tasks (while ensuring precedence is preserved). In addition, mutation is performed only on the children in order to address the small population used (since there is no desire to carry over a poor performing parent unchanged for generations) and to counter PPXs tendency to duplicate parents. DLBP GA is modified to treat duplicate solutions as if they had the worst fitness performance, relegating them to replacement in the next generation. To avoid becoming trapped in local optima, the DLBP GA is run, not until a desired level of performance was reached but for as many generations as is acceptable by the user. Only feasible disassembly sequences are considered as members of the population. The fitness is computed using formula (2). The original technique of randomly generating the initial population [19] was ineffective with the electronic product case study studied here due to its relatively large size and numerous

precedence constraints, so four feasible and diverse solutions were manually generated to hot start DLBP GA. A small population was used (20 versus the more typical 10,000 to 100,000) to minimize data storage requirements and simplify analysis, while a large number of generations were used (10,000 versus 10 to 1,000) to compensate for this small population while not being so large as to take excessive processing time. Lower than the recommended 90% [12], a 60% crossover was selected based on test and analysis (60% crossover provided better solutions and did so with 1/3 less processing time). Also, previous assembly line balancing and disassembly GA literature indicate best results typically being found with crossover rates of 0.5 to 0.7. Mutation was performed 1.0% of the time. Although some texts recommend 0.01% mutation and applications in journals have used as much as 100% mutation, it was found that 1.0% gave excellent performance.

NUMERICAL RESULTS

DLBP GA and the DLBP Greedy/AEHC hybrid were both written in C++ and run on a 1.6GHz PM x86-family workstation. The developed algorithms were investigated on a variety of test cases for verification and validation purposes. They were then compared using a case study from the literature based on the electronic product presented by Gupta *et al.* [9]. Here, the objective is to completely disassemble a cell phone consisting of $n = 25$ components with various precedence relationships in an electronic product disassembly flow shop with a line operating at a speed which allows $CT = 18$ seconds at each workstation.

For the results in table 1, the DLBP Greedy/AEHC heuristic hybrid was run three times to obtain an average computation time, while DLBP GA was run five times (unlike Greedy/AEHC, GA is highly probabilistic, so additional runs were conducted to achieve an average measure of performance along with best and worst case). DLBP GA typically found feasible solutions to the electronic product data set in less than one second. DLBP Greedy/AEHC found its solution extremely quickly, regularly taking less than 1% of the time of DLBP GA. For comparison, a much smaller problem ($n = 10$) solved by exhaustive search on the same workstation took almost 10 seconds. At this rate, solving the electronic product problem to optimality using exhaustive search would take over 250-million years. Neither technique was able to generate the optimal number of workstations or the optimal balance (with the assumption that $NWS = 9$ and $F = 9$ is optimal [9]). Using the four manually generated solutions for a hot start, DLBP GA gave five different answers over five runs, seeming to indicate that the hot start solutions were adequate in number and diversity. DLBP GA, on average, provided a better measure of balance and slightly better H and D placement, while DLBP Greedy/AEHC gave slightly better R placement. For a more in-depth study of these methods and analysis on more diverse problem sets, see [18] and [20]. Although these results are not optimal, this is more a reflection of the challenges posed even by seemingly simple disassembly problems more than an indication of any limitations of these techniques. Note that the inclusion of additional precedence constraints will increasingly move both methods towards optimal.

Solution	Time	NWS	F	H	D	R
hybrid	0.007	10	199	89	973	10
GA	0.863	10.0	81.0	78.2	915.2	10.2

Table 1: Comparison of averaged Greedy/AEHC and GA solutions for the electronic product example.

CONCLUSIONS

Two very fast combinatorial optimization approaches to the multi-objective DLBP were developed and compared. Though distinct in their approaches, both methods provide feasible solutions to disassembly flow shop problems, with one using a stochastic distributed search and the other, a hybrid pairing of two deterministic searches. These diverse solution-generating techniques appear well suited to the multiple-criteria decision-making problem format as well as to the solution of problems with nonlinear objectives. In addition, they are ideally suited to integer problems that generally do not lend themselves to application of traditional mathematical programming techniques.

REFERENCES

- [1] Bierwirth, C., Mattfeld, D. C., Kopfer, H. "On permutation representations for scheduling problems, parallel problem solving from nature." Voigt, H. M., Ebeling, W., Rechenberg, I., Schwefel, H. P. eds., *Lecture notes in computer science*. Berlin, Germany: Springer-Verlag, 1996, 1141, 3.10-3.18.

- [2] Brennan, L., Gupta, S. M., Taleb, K. N. "Operations planning issues in an assembly/disassembly environment." *International Journal of Operations and Production Planning*, 1994, 14(9), 57-67.
- [3] Elsayed, E. A., Boucher, T. O. *Analysis and control of production systems*. Upper Saddle River, NJ: Prentice Hall, 1994.
- [4] Garey, M. R., Johnson, D. S. *Computers and intractability: A guide to the theory of NP-completeness*. New York, NY: W. H. Freeman and Company, 1979.
- [5] Gungor, A., Gupta, S. M. "A systematic solution approach to the disassembly line balancing problem." *Proceedings of the 25th International Conference on Computers and Industrial Engineering*, New Orleans, Louisiana, March 1999, 70-73.
- [6] Gungor, A., Gupta, S. M. "Disassembly line balancing." *Proceedings of the 1999 Annual Meeting of the Northeast Decision Sciences Institute*, Newport, Rhode Island, March 1999, 193-195.
- [7] Gungor, A., Gupta, S. M. "Disassembly line in product recovery." *International Journal of Production Research*, 2002, 40(11), 2569-2589.
- [8] Gungor, A., Gupta, S. M. "Issues in environmentally conscious manufacturing and product recovery: a survey." *Computers and Industrial Engineering*, 1999, 36(4), 811-853.
- [9] Gupta, S. M., Evren, E., McGovern, S. M. "Disassembly sequencing problem: A case study of a cell phone." *Proceedings of the 2004 SPIE International Conference on Environmentally Conscious Manufacturing IV*, Philadelphia, Pennsylvania, October 2004, 43-52.
- [10] Gupta, S. M., McGovern, S. M. "Multi-objective optimization in disassembly sequencing problems (Invited Paper)." *Proceedings of the 2nd World Conference on Production & Operations Management and the 15th Annual Production & Operations Management Conference*, Cancun, Mexico, April 2004, CD-ROM.
- [11] Gupta, S. M., Taleb, K. N. "Scheduling disassembly." *International Journal of Production Research*, 1994, 32, 1857-1866.
- [12] Koza, J. R. *Genetic programming: On the programming of computers by the means of natural selection*. Cambridge, MA: MIT Press, 1992.
- [13] McGovern, S. M., Gupta, S. M. "2-opt heuristic for the disassembly line balancing problem." *Proceedings of the 2003 SPIE International Conference on Environmentally Conscious Manufacturing III*, Providence, Rhode Island, October 2003, 71-84.
- [14] McGovern, S. M., Gupta, S. M. "Combinatorial optimization methods for disassembly line balancing." *Proceedings of the 2004 SPIE International Conference on Environmentally Conscious Manufacturing IV*, Philadelphia, Pennsylvania, October 2004, 53-66.
- [15] McGovern, S. M., Gupta, S. M. "Demufacturing strategy based upon metaheuristics (Invited Paper)." *Proceedings of the 2004 IIE Industrial Engineering Research Conference*, Houston, Texas, May 2004, CD-ROM.
- [16] McGovern, S. M., Gupta, S. M. "Greedy algorithm for disassembly line scheduling (Invited Paper)." *Proceedings of the 2003 IEEE International Conference on Systems, Man, and Cybernetics*, Washington, DC, October 2003, 1737-1744.
- [17] McGovern, S. M., Gupta, S. M. "Metaheuristic technique for the disassembly line balancing problem." *Proceedings of the 2004 Northeast Decision Sciences Institute Conference*, Atlantic City, New Jersey, March 2004, 223-225.
- [18] McGovern, S. M., Gupta, S. M. "Multi-criteria ant system and genetic algorithm for end-of-life decision making." *Proceedings of the 35th Annual Meeting of the Decision Sciences Institute*, Boston, Massachusetts, November 2004, 6371-6376.
- [19] McGovern, S. M., Gupta, S. M., Kamarthi, S. V. "Solving disassembly sequence planning problems using combinatorial optimization." *Proceedings of the 2003 Annual Meeting of the Northeast Decision Sciences Institute*, Providence, Rhode Island, March 2003, 178-180.
- [20] McGovern, S. M., Gupta, S. M., Nakashima, K. "Multi-criteria optimization for non-linear end of lifecycle models." *Proceedings of the Sixth Conference on EcoBalance*, Tsukuba, Japan, October 2004.