

January 01, 2003

## Solving disassembly sequence planning problems using combinatorial optimization

Surendra M. Gupta  
*Northeastern University*

Seamus M. McGovern  
*Northeastern University*

Sagar V. Kamarthi  
*Northeastern University*

---

### Recommended Citation

Gupta, Surendra M.; McGovern, Seamus M.; and Kamarthi, Sagar V., "Solving disassembly sequence planning problems using combinatorial optimization" (2003).. Paper 106. <http://hdl.handle.net/2047/d10013842>



Laboratory for Responsible Manufacturing

## **Bibliographic Information**

McGovern, S. M., Gupta S. M. and Kamarthi, S. V., "Solving Disassembly Sequence Planning Problems Using Combinatorial Optimization", ***Proceedings of the 2003 Northeast Decision Sciences Institute Conference***, Providence, Rhode Island, pp. 178-180, March 27-29, 2003.

## **Copyright Information**

*Copyright 2003, 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**

# SOLVING DISASSEMBLY SEQUENCE PLANNING PROBLEMS USING COMBINATORIAL OPTIMIZATION

**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

**Sagar V. Kamarthi**, Northeastern University, Boston, MA 02115, (617)-373-3070, sagar@coe.neu.edu

(\*Corresponding author)

## ABSTRACT

Disassembly activities take place in various recovery operations including remanufacturing, recycling, and disposal. The disassembly line is the best choice for automated disassembly of returned products, a feature that will become crucial in the future. It is, therefore, important that the disassembly line be designed and balanced so that it works as efficiently as possible. However, finding the optimal balance is computationally intensive with exhaustive search quickly becoming prohibitively large, even for relatively small products. In this paper, we solve the disassembly line balancing problem using combinatorial optimization techniques, which are instrumental in obtaining near-optimal solutions to problems with intractably large solution spaces. Specifically, a genetic algorithm is presented for obtaining optimal or near-optimal solutions for disassembly line balancing problems. An example is presented to illustrate the implementation of the methodology.

## INTRODUCTION

In recent years, the implementation of extended manufacturer responsibility together with new more rigid environmental legislation and increased public awareness has caused a growing number of manufacturers to begin recycling and remanufacturing their post consumed products after they are discarded by consumers. In addition, the economic attractiveness of reusing products, subassemblies or parts instead of disposing of them has further fueled this effort. Recycling is a process performed to retrieve the material content of used and non-functioning products. Remanufacturing, on the other hand, is an industrial process in which worn-out products are restored to like-new conditions. Thus, remanufacturing provides the quality standards of new products with used parts.

Product recovery aims to minimize the amount of waste sent to landfills by recovering materials and parts from old or outdated products by means of recycling and remanufacturing (including reuse of parts and products). There are many attributes of a product that enhance product recovery; examples include: ease of disassembly, modularity, type and compatibility of materials used, material identification markings and efficient cross-industrial reuse of common parts/materials.

The first crucial step of product recovery is disassembly. Disassembly is a methodical extraction of valuable parts/subassemblies and materials from discarded products through a series of operations. After disassembly, reusable parts/subassemblies are cleaned, refurbished, tested and directed to the part/subassembly inventory for remanufacturing operations. The recyclable materials can be sold to raw-material suppliers, while the residuals are sent to landfills.

Disassembly has recently gained a great deal of attention in the literature due to its role in product recovery. A disassembly system faces many unique challenges; for example, it has significant inventory problems because of the disparity between the demands for certain parts or subassemblies and their yield from disassembly. The flow process is also different. As opposed to the normal "convergent" flow in regular assembly environment, in disassembly the flow process is "divergent" (a single product is broken down into many subassemblies and parts). There is also a high degree of uncertainty in the structure and the quality of the returned products. The conditions of the products received are usually unknown and the reliability of the components is suspect. In addition, some parts of the product may cause pollution or may be hazardous. These parts tend to have a higher chance of being damaged and hence may require special handling, which can also influence the utilization of the disassembly workstations. Various demand sources may also lead to complications in disassembly line balancing. Disassembly line balancing is critical in minimizing the use of valuable resources (such as time and money) invested in disassembly and maximizing the level of automation of the disassembly process and the quality of the parts (or materials) recovered.

In this paper, we solve the disassembly line balancing problem (DLBP) using combinatorial optimization. Combinatorial optimization is instrumental in obtaining near-optimal solutions to problems with intractably large solution spaces. A genetic algorithm is presented for obtaining optimal or near-optimal solutions for the DLBP. The genetic algorithm considered here involves a randomly generated initial population with precedence preservative crossover (PPX), mutation and fitness calculation performed over many generations. An example is considered to illustrate the implementation of the methodology. The conclusions drawn from the study

include the consistent generation of optimal or near-optimal solutions, the ability to preserve precedence relationships, the superior speed of the algorithm and its practicality due to the ease of implementation in solving disassembly line balancing problems.

### LITERATURE REVIEW

Many authors have discussed the different aspects of product recovery. Brennan et al. [2], and Gupta and Taleb [6] investigated the problems associated with disassembly planning and scheduling.

Gungor and Gupta [4], [5] presented the first introduction to the disassembly line balancing problem by developing an algorithm for solving the DLBP with the goal of assigning tasks to workstations in a way that probabilistically minimizes the cycle time.

For a comprehensive review of environmentally conscious manufacturing and product recovery, see Gungor and Gupta [3].

### MODEL DESCRIPTION AND THE ALGORITHM

A version of a combinatorial optimization solution technique known as the genetic algorithm (GA); a parallel neighborhood search technique, is used to provide a near-optimal solution to the disassembly line balancing problem presented by Gungor and Gupta [5] where the objective is to completely disassemble a given product consisting of 8 components on a disassembly line operating at a speed which allows 40 seconds for each workstation to perform its required disassembly tasks.

The particular application investigated in this paper seeks to fulfill three objectives: 1) provide a feasible disassembly sequence for the product being investigated, 2) minimize the number of disassembly workstations and hence, minimize the total idle time, and 3) balance the disassembly line (i.e., ensure the idle times at each workstation are similar and minimized).

The following notation are used in the remainder of the paper:

$CF_i$	fitness of chromosome $i$
$CT$	cycle time
$G$	number of genes (parts for removal)
$N$	number of chromosomes (population)
$NWS_i$	total number of workstations for chromosome $i$
$PRT_{ik}$	time required to remove $k$ th part in chromosome $i$
$R_m$	crossover rate
$R_x$	mutation rate
$WS_j$	elapsed time in workstation $j$

A chromosome (solution) consisted of a sequence of genes (work elements). For example, if a chromosome in

the population consisted of “5 2 8 1 4 7 6 3,” then component 5 would be removed first, followed by component 2, then component 8, and so on. A population, or pool, of size  $N$  was used. Only feasible disassembly sequences were considered as members of the population or as offspring. The fitness was computed for each chromosome based on the minimum number of workstations required as well as the sum of the square of the idle times for all the workstations. This was done to penalize solutions where, even though the number of workstations is low, one or more have an exorbitant amount of idle time when compared to the other workstations. This tends to balance the task assignment among different workstations on the disassembly line. Therefore, a minimum fitness value is the more desirable solution indicating both a minimum number of workstations and similar idle times across all workstations.

The GA for this DLBP was constructed as follows. The initial, feasible population was randomly generated, preserving the precedence constraints of the product being studied. The fitness of each chromosome in this generation was then calculated. An even number of  $R_x N$  parents were randomly selected for crossover to produce  $R_x N$  offspring (offspring make up  $R_x N * 100$  percent of each generation's population). An elegant crossover, the precedence preservative crossover (PPX) developed by Bierwirth, et al. [1], was used to create the offspring. The PPX first creates a mask (one for each child, every generation). The mask consists of random 1s and 2s, indicating which parent information should be taken from. If, for example, the mask for child 1 reads 221211, the first two parts (i.e., from left to right) in parent 2 would make up the first two genes of child 1 (and these parts would be stricken from the parts available to take from both parent 1 and 2); the first available (i.e., not stricken) part in parent 1 would make up gene three of child 1; the next available part in parent 2 would make up gene four of child 1; the last two parts in parent 1 would make up genes five and six of child 1. This technique is repeated using a new mask for child 2. After crossover, mutation is randomly conducted. Mutation was occasionally performed by randomly selecting a child then exchanging two of its disassembly tasks while ensuring precedence is preserved. The  $R_x N$  least fit parents are removed by sorting the entire parent population from worst-to-best based on fitness. Since the GA saves the best parents from generation to generation and it is possible for duplicates of a solution to be generated using PPX, the solution set could consist of multiple copies of the same answer resulting in the algorithm potentially becoming trapped in a local optima. This becomes more likely in a GA with solution constraints (such as precedence requirements) and small populations, both of which are seen in our problem. To

avoid this, the GA was modified to treat duplicate solutions as if they had the worst fitness performance (highest numerical value), relegating them to replacement in the next generation. With this new ordering, the best unique  $(1 - R_x)N$  parents were kept along with all of the  $R_xN$  offspring to make up the next generation then the process was repeated. To again avoid becoming trapped in local optima, the GA for this particular problem, as with many combinatorial optimization techniques, was run, not until a desired level of performance was reached, but rather for as long as was acceptable by the user. Since this GA always keeps the best solutions from generation to generation, there is no risk of solution drift or bias, and the possibility of mutation allows for a diverse range of possible solution space visits over time.

### NUMERICAL RESULTS

The developed GA was investigated on a variety of test cases to confirm its performance and to optimize parameters. An example from the literature, Gungor and Gupta [5], was then selected as an application to an actual disassembly line balancing problem. This practical and relevant example consists of the data for the disassembly of a personal computer (PC). It consists of 8 parts (the number of genes is therefore 8) with part removal times of 14, 10, 12, 18, 23, 16, 20, and 36 respectively. It was assumed that the disassembly line operates at a speed that allowed 40 seconds ( $CT = 40$ ) for each workstation. The number of chromosomes (possible solutions) was selected to be 20, giving a total of 8 best answers carried over from generation to generation. The crossover rate was 60% (literature indicates best results have typically been found with crossover rates of from 0.5 to 0.7), which used  $R_xN$  (i.e.,  $0.6*20$  or 12) parents (and therefore produced 12 offspring for inclusion in the population each generation). A mutation was performed about 1 percent of the time.

The GA converged to moderately good solutions very quickly. It was able to find at least one of the four optimal solutions after no more than 10 generations. 100 generations consistently provided three to four of the optimal solutions, while 1000 generations almost always resulted in the generation of all four optimal solutions. Similarly, the speed for the C++ implemented program searching 1000 generations of 20 chromosomes each was less than one second on a 400MHz PC.

The balancing aspect of the algorithm worked extremely well with four equivalent (and, in fact, optimal) solutions being found. All four best solutions consisted of a total of four workstations with  $3 \pm 1$  seconds per workstation of idle time. Therefore, the obtained solutions were optimal in number of workstations and also provided idle times at each workstation of at least 5% but not more than 10% of the total disassembly time allocated to each workstation of 40 seconds.

All three objectives (i.e., generate a feasible disassemble sequence, minimize the number of workstations, and balance the disassembly line) were achieved in the DLBP implementation of this particular GA. Although a near-optimum combinatorial optimization technique, the DLBP GA consistently and rapidly found all of the optimal solutions in what approaches an exponentially large search space (potentially as large as  $8!$  or 40,320). In addition, the GA seems ideally suited to integer problems, a requirement of many disassembly problems, which generally do not lend themselves to rapid or easy solution by traditional optimum solution generating mathematical programming techniques.

### REFERENCES

- [1] Bierwirth, C., Mattfeld, D.C., and Kopfer, H., 1996, On permutation representations for scheduling problems, *Parallel Problem Solving from Nature*, Voigt, H.M., Ebeling, W., Rechenberg, I. and Schwefel, H.P. eds., Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, 1141, 3.10-3.18.
- [2] Brennan, L., Gupta, S.M. and Taleb, K.N., 1994, Operations planning issues in an assembly/disassembly environment. *International Journal of Operations and Production Planning*, 14(9), 57-67.
- [3] Gungor, A. and Gupta, S.M., 1999, Issues in environmentally conscious manufacturing and product recovery: a survey. *Computers and Industrial Engineering*, 36(4), 811-853.
- [4] Gungor, A. and Gupta, S.M., 2001, A solution approach to the disassembly line problem in the presence of task failures, *International Journal of Production Research*, 39(7), 1427-1467.
- [5] Gungor, A. and Gupta S.M., 2002, Disassembly line in product recovery, *International Journal of Production Research*, 40(11), 2569-2589.
- [6] Gupta, S.M. and Taleb, K.N., 1994, Scheduling disassembly. *International Journal of Production Research*, 32, 1857-1866.