January 01, 2003

Greedy algorithm for disassembly line scheduling

Seamus M. McGovern
Northeastern University

Surendra M. Gupta
Northeastern University

Laboratory for Responsible Manufacturing

Bibliographic Information

McGovern, S. M. and Gupta, S. M. “Greedy Algorithm for Disassembly Line Scheduling”, ***Proceedings of the 2003 IEEE International Conference on Systems, Man, and Cybernetics***, Washington, DC, pp. 1737-1744, October 2003.

Copyright Information

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***

Greedy Algorithm for Disassembly Line Scheduling*

Seamus M. McGovern
Laboratory for Responsible Manufacturing
Department of Mechanical, Industrial and Manufacturing Engineering
Northeastern University
Boston, MA, U.S.A.
mcgovern.s@neu.edu

Surendra M. Gupta**
Laboratory for Responsible Manufacturing
Department of Mechanical, Industrial and Manufacturing Engineering
Northeastern University
Boston, MA, U.S.A.
gupta@neu.edu

Abstract - *Remanufacturing, recycling, and disposal recovery operations require the performance of disassembly activities. The disassembly line is the best choice for automated disassembly of returned products, however, finding the optimal balance is computationally intensive with exhaustive search quickly becoming prohibitively large. In this paper, a greedy algorithm is presented for obtaining optimal or near-optimal solutions to the disassembly line balancing problem. The greedy algorithm is a first-fit decreasing algorithm further enhanced to preserve precedence relationships. The algorithm seeks to minimize the number of workstations while accounting for hazardous and high demand components. A hill-climbing heuristic is then developed to balance the part removal sequence. Examples are considered to illustrate the methodology. The conclusions drawn from the study include the consistent generation of optimal or near-optimal solutions, the ability to preserve precedence, the speed of the algorithm and its practicality due to the ease of implementation.*

Keywords: Disassembly, disassembly line balancing, combinatorial optimization, greedy algorithm, hill-climbing, heuristics, product recovery.

1 Introduction

New, more rigid environmental legislation, increased public awareness, and extended manufacturer responsibility has caused a growing number of manufacturers to begin recycling and remanufacturing their post-consumed products after they have been disposed of 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.

With the goal of minimizing the amount of waste sent to landfills, product recovery obtains materials and parts from old or outdated products through 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.

Due to its role in product recovery, disassembly has gained a great deal of attention in the recent literature. 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

* **0-7803-7952-7/03/$17.00 Ó 2003 IEEE.**

** Corresponding author

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 a greedy algorithm and a subsequent hill-climbing heuristic. While exhaustive search consistently provides the problems' optimal solution, its exponential time complexity quickly reduces the practicality of this type of search. The combination of the greedy algorithm and the hill-climbing heuristic, however, is instrumental in rapidly obtaining near-optimal solutions to the DLBPs intractably large solution space. This technique is so rapid, in fact, that it should lend itself to real-time solution of the DLBP on a dynamic disassembly line as components arrive for disassembly on mixed-model and mixed-product lines. The greedy algorithm considered here is based on the First-Fit Decreasing (FFD) algorithm effectively used in computer processor scheduling and enhanced to preserve precedence relationships within the product being disassembled. The FFD is further modified to a multi-objective greedy algorithm that seeks to minimize the number of workstations while attempting to remove hazardous and high demand product components as early as possible. A hill-climbing heuristic is then developed to balance the part removal sequence (i.e., ensure that the idle times at each workstation are similar). Herein referred to as the Adjacent Element Hill Climbing (AEHC) heuristic, the AEHC only compares tasks assigned in adjacent workstations. This is done both to conserve search time (by not investigating all tasks in all workstations) and to only investigate swapping tasks that will most likely result in a feasible sequence (since the farther apart the positional changes, the less likely that precedence will be preserved for both of the tasks and for all of the tasks between them). Examples are 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 method, and its practicality due to the ease of implementation in solving disassembly line balancing problems.

2 Literature review

There are many steps involved in product recovery [4]. The first crucial step is disassembly. Disassembly is a methodical extraction of valuable parts/subassemblies and materials from post-used products through a series of operations [1] [9]. After disassembly, re-usable 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 and the residuals are disposed of.

Gungor and Gupta presented the first introduction to the disassembly line balancing problem [5] [6] [8] and developed an algorithm for solving the DLBP in the presence of failures with the goal of assigning tasks to workstations in a way that probabilistically minimizes the cost of defective parts [7]. Tang et al. developed an algorithm to facilitate disassembly line design and optimization [12]. McGovern et al. applied combinatorial optimization techniques to the DLBP [11].

3 Notation

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

CT	cycle time
F	balance for a given solution sequence
F^*	current best balance
I	total idle time for a given solution sequence
I_j	total idle time of workstation j
I_{max}	maximum possible total idle time
I_{min}	minimum possible total idle time
ISS_k	binary value; 1 if k^{th} part is in solution sequence, else 0
j	workstation count (0,…, NWS)
k	part identification (1,…, n)
n	number of parts for removal
NWS	number of workstations for a given solution sequence
NWS_j	identification of workstations for a given solution
NWS_{max}	maximum possible number of workstations for PRT
NWS_{min}	minimum possible number of workstations for PRT
PRT	part removal time
PRT_k	time required to remove kth part
$\|\|PRT_U\|\|$	cardinality of the set of unique part removal times
PSG_k	k^{th} part in the solution sequence after application of the greedy algorithm
PSH_k	k^{th} part in the solution sequence after application of the AEHC heuristic
PSS_k	k^{th} part in sequence after sorting
PWS_k	workstation the k^{th} part is assigned to
WS_j	elapsed time in workstation j

4 The DLBP model description

The particular application investigated in this paper seeks to fulfill five 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.
3. Balance the disassembly line (i.e., ensure the idle times at each workstation are similar).
4. Remove hazardous components early in the disassembly sequence.
5. Remove high demand components before low demand components in the case of equal part removal times.

The result is an integer, multi-criteria decision making problem with an exponential search space. Testing a given solution against the precedence constraints fulfills objective 1. Minimizing the sum of the workstation idle times, which will also minimize the total number of workstations, attains objective 2. This objective is represented as:

$$\text{Min } Z = \sum_{j=1}^{NWS} (CT - WS_j) \quad (1)$$

Line balancing seeks to achieve Perfect Balance (all idle times equal to zero). When this is not achievable, either Line Efficiency (IE) or the Smoothness Index (SI) is used as a performance evaluation tool, Elsayed and Boucher [3]. We use a measure of balance that combines the two and is easier to calculate [11]. SI rewards similar idle times at each workstation, but at the expense of allowing for a large (suboptimal) number of workstations. This is because SI compares workstation elapsed times to the largest WS_j instead of the CT as this method does. IE rewards the minimum number of workstations, but allows unlimited variance in idle times between workstations because no comparison is made between WS_js. The balancing method used here simultaneously minimizes the number of workstations while aggressively ensuring that idle times at each workstation are similar. The method is computed 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 penalizes solutions where, even though the number of workstations may be minimized, one or more have an exorbitant amount of idle time when compared to the other workstations. It provides for leveling the workload between different workstations on the disassembly line. Therefore, a resulting minimum performance value is the more desirable solution indicating both a minimum number of workstations and similar idle times across all workstations. This objective is represented as:

$$\text{Min } Z = \sum_{j=1}^{NWS} (CT - WS_j)^2 \quad (2)$$

With Perfect Balance indicated by:

$$\text{Min } Z = 0 \quad (3)$$

Note that mathematically, this objective function effectively makes objective 2 redundant due to the fact that it concurrently minimizes the WS_js.

In addition, we find:

$$NWS_{max} = n \quad (4)$$

$$NWS_{min} = \left\lceil \frac{\sum_{k=1}^{n} PRT_k}{CT} \right\rceil \quad (5)$$

$$I = \sum_{j=1}^{NWS} (CT - WS_j) \quad (6)$$

Problem assumptions include:

1. Part removal times are deterministic
2. Part removal times are constant
3. Each product undergoes complete disassembly
4. All products contain all parts with no additions, deletions, or modifications
5. All the parts are assigned
6. Each part is assigned to one and only one workstation
7. The sum of the part removal times of all the parts assigned to a workstation must not exceed CT
8. The precedence relationships among the parts must not be violated

5 The greedy algorithm and AEHC heuristic

A two-phased approach is used to provide a very fast, near-optimal solution to the multi-objective DLBP. The first phase rapidly provides a feasible solution to the DLBP and minimum or near-minimum NWS using a greedy algorithm based on the FFD algorithm. The second phase is then implemented to compensate for the DLBP greedy algorithms' inability to balance the workstations. This local search quickly provides a near-optimal and feasible balance sequence using a hill climbing heuristic, AEHC.

5.1 Greedy model description and the algorithm

A greedy strategy always makes the choice that looks the 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. Greedy algorithms do not always yield optimal solutions, but for many problems they do [2]. The DLBP greedy algorithm was built around the FFD algorithm. The FFD algorithm looks at each element in a list, from largest to smallest (*PRT* in the DLBP), and puts that element into the first workstation in which it fits. When all of the work elements have been assigned to a workstation, the process is complete. The greedy FFD is further modified with priority rules to meet multiple objectives. The most hazardous parts are prioritized to the earliest workstations, greedy ranked large removal time to small. The remaining non-hazardous parts are greedy ranked next, large removal times to small. In addition, selecting the part with the larger demand ahead of those with lesser demands breaks any ties for parts with equal part removal times. This is done to prevent damage to these more desirable parts. The DLBP greedy algorithm provides an optimal or near optimal minimum number of workstations; the more constraints, the more likely the minimal number of workstations is found. The level of performance (minimal *NWS*) tends to increase with the number of precedence constraints.

The specific details for this implementation are as follows. The DLBP greedy algorithm first sorts the list of parts. The sorting is based on part removal times, whether or not the part contains hazardous materials, and the subsequent demand for the removed part. Hazardous parts are put at the front of the list for selection into the solution sequence. The hazardous parts are ranked from largest to smallest part removal times. The same is then done for the non-hazardous parts. Any ties (i.e., two parts with equal part removal times) are not randomly broken, but rather ordered based on the demand for the part, with the higher demand part being placed earlier on the list.

Once the parts are sorted in this multi-criteria manner, the parts are placed in workstations in FFD greedy order while preserving precedence. Each part in the sorted list is examined from first to last. If the part had not previously been put into the solution sequence (as described by ISS_k), the part is put into the current workstation if idle time remains to accommodate it and as long as putting it into the sequence at that position will not violate any of its 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 (i.e., its ISS_k value remains 0) 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 on the greedy solution list, a new workstation is created and the process is repeated. The following is the DLBP greedy procedure (post-sorting).

```
PROCEDURE_GREEDY{
    prt_ctr = 0;
    j = 0 ;
    FOR ∀ k {
        WHILE ((ISS PSSprt_ctr) ∨
                (PRT PSSprt_ctr > I j) ∨
                (PROCEDURE_PREC_FAIL)){
            IF (prt_ctr < n - 1){
                prt_ctr = prt_ctr + 1;
            }
            ELSE{
                j = j + 1;
                prt_ctr = 0;
            }
        }
        I j = I j – PRT PSSprt_ctr;
        PSG k = PSS prt_ctr;
        PWS k = j;
        ISS PSSprt_ctr = 1;
        prt_ctr = 0;
    }
    F = PROCEDURE_CALC_BALANCE;
    RETURN;
}
```

While being very fast and generally very efficient, the FFD-based greedy algorithm is not always able to optimally 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 results in extremely poor balance. This limitation led to the development of the AEHC to fulfill objective 3.

5.2 Hill climbing description and the heuristic

The second phase of the two-phase approach to the multi-objective DLBP is implemented to compensate for the DLBP greedy algorithms' inability to balance the workstation assignments. The second phase quickly provides a near-optimal and feasible balance sequence using a hill-climbing local search heuristic, AEHC. Hill-climbing is an iterated improvement algorithm, basically a gradient descent/ascent. It makes use of an iterative greedy strategy, which is to move in the direction of increasing value. A hill-climbing algorithm evaluates the successor states and keeps only the best one [10]. AEHC is designed to consider swapping each task in every workstation with each task in the next adjacent workstation in search of improved balance. It does this while preserving precedence and not exceeding *CT* in any WS_j.

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.

The neighborhood definition and search details of the AEHC are as follows. After the DLBP greedy algorithm generates a minimum *NWS*, feasible solution, the AEHC heuristic is applied to improve the balance. The AEHC does this by going through each task element (part) in each workstation and comparing it to each task element in the next adjacent workstation. If the two task elements can be exchanged while preserving precedence, without exceeding either workstations available idle time, and with a resulting improvement in the overall balance, the exchange is made and the resulting solution sequence is saved as the new solution sequence. This process is repeated until task elements of the last workstation have been examined. The heuristic is given below in pseudo-code format.

```
PROCEDURE_HILL_CLIMB{
  rt_ctr = 0;
  PROCEDURE_COPY (∀ PSG_k, I_j, F, NWS, → ∀ best);
  WHILE (PWS_rt_ctr = 0){
    rt_ctr = rt_ctr + 1;
  }
  DO{
    prt_rt_start = rt_ctr;
    F* = best.F;
    prt_lft = 0;
    FOR ∀ j ∈ NWS{
      WHILE (PWS_prt_lft = j){
        prt_rt = prt_rt_start;
        WHILE (PWS_prt_rt = j + 1){
          PROCEDURE_COPY (∀ best, → ∀ temp);
          PROCEDURE_SWAP (prt_rt, prt_lft, j, temp);
            IF ((temp.I_j ≥ 0) ∧
            (temp.I_j+1 ≥ 0) ∧
            (temp.F < best.F) ∧
            (PROCEDURE_PRECEDENCE_PASS (
                (∀ temp, prt_rt, prt_lft))){
              PROCEDURE_COPY (∀ temp → ∀ best);
            }
          prt_rt = prt_rt + 1;
        }
        prt_lft = prt_lft + 1;
      }
      prt_rt_start = prt_rt;
    }
  }
  WHILE (best.F < F*);
  PROCEDURE_COPY (∀ best→ PSH_k, I_j, F);
  RETURN;
}
```

As is the norm with greedy algorithms, the DLBP greedy process is run once to determine a solution. Hill-climbing, however – like many combinatorial optimization techniques – is typically continuously run on subsequent solutions for as long as is deemed appropriate or 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 [10]. Repeating the AEHC method in this way provides improved balance over time. The AEHC was tested both ways; run only once after the greedy solution was generated, as well as run until the local optimum was obtained. A single AEHC iteration has several benefits, especially since the problem sets investigated in this paper were relatively small and lent themselves to good solutions after only one iteration. Also, the AEHC was seen to provide its largest balance performance improvement in just one iteration. AEHC approaches or reaches the local optima the first time it is run. Additionally, a single iteration of AEHC is recommended since the process proposed in this paper is aimed at a real-time solution of the DLBP on a dynamic mixed-model and mixed-product disassembly line.

6 Numerical results

The developed algorithms were investigated on a variety of test cases to confirm their performance and to optimize parameters. Both the DLBP greedy algorithm and the AEHC were used sequentially to provide a solution to the disassembly line balancing problem presented by Gungor and Gupta [8] where the objective is to completely disassemble a given component consisting of n subassemblies on a disassembly line operating at a speed which allows CT seconds for each workstation to perform its required disassembly tasks. This provided 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) as shown in table 1. It consists of 8 subassemblies with part removal times of PRT = {14, 10, 12, 18, 23, 16, 20, 36}. The disassembly line is operating at a speed that allows 40 seconds (CT = 40) for each workstation.

Table 1. Knowledge base of the personal computer example

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

The DLBP greedy algorithm and the AEHC obtained an optimal solution for this problem and did so very quickly. The greedy algorithm alone was able to successfully find one of the four equivalent, optimal solutions (table 2). The AEHC found no better solution nearby and returned the original, optimal solution sequence. The speed for the C++ implemented program on this problem was less than 1/100th of a second for both the DLBP greedy algorithm and the AEHC on a 2.5GHz P4 x86 family computer.

Table 2. The optimal disassembly sequence solution

Part removal sequence	Workstation 1	2	3	4	Time to remove part (in seconds)
1	14				14
5	23				23
3		12			12
6		16			16
2		10			10
8			36		36
7				20	20
4				18	18
Total time	37	38	36	38	
Idle time	3	2	4	2	

The optimality of the *NWS* minimization and the balancing is demonstrated by the solution consisting of a total of four workstations with 3 $\pm$ 1 seconds per workstation of idle time. Therefore, the obtained solution was 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. This solution is consistent with the solution found by M^{c}Govern et al. [11].

The first three objectives (generate a feasible disassemble sequence, minimize the number of workstations, and balance the disassembly line) were achieved by the DLBP greedy algorithm and the AEHC. The last two objectives (remove hazardous components early in the disassembly sequence and remove high demand components before low demand components) were preempted by rigid enforcement of the precedence constraints. The DLBP greedy algorithm and the AEHC heuristic consistently and rapidly found an optimal solution in what approaches an exponentially large search space (potentially as large as 8! or 40,320).

The developed DLBP greedy algorithm and the AEHC heuristic were then used on a test case to further demonstrate their performances as well as their limitations. This was done by using part times consisting exclusively of prime numbers. They were further selected to ensure that no combinations of these part removal times allowed for any equal summations in order to reduce the number of possible optimal solutions. For example, part removal times 1, 3, 5 and 7 and $CT = 16$ would have minimum idle time solutions of not only one 1, one 3, one 5 and one 7 at each workstation, but various additional combinations of these as well since $1 + 7 = 3 + 5 = \frac{1}{2}\ CT$. Subsequently, the chosen instances were made up of parts with removal times of 3, 5, 7 and 11 and $CT = 26$. As a result, the optimal balance for all subsequent instances would consist of a perfect balance of precedence preserving combinations of 3, 5, 7 and 11 at each workstation with idle times of 0. To further complicate the data (i.e., provide a large, feasible search space), only one part was listed as hazardous and this was one of the parts with the largest part removal time. This was done so that only the hazardous sequencing would be demonstrated, while providing no solution sequence advantage to the DLBP greedy algorithm/AEHC heuristic. In addition, the demand was made equal for all the parts except for one to demonstrate demand sequencing, again while providing no advantage for the DLBP greedy algorithm/AEHC heuristic. Also, there were no precedence constraints placed on the sequence, a deletion that further challenges the methods' ability to attain an optimal solution. Finally, a small n was selected which decreases the *NWS* and tends to exaggerate less than optimal performance. The final test data consisted of 12 parts; 4 subassemblies with 4 unique part removal times of 3, 5, 7 and 11. The disassembly line is operated at a speed that allows 26 seconds ($CT = 26$) for each workstation (Table 3a). For any n parts, the following can be calculated:

$$NWS_{\max} = n \tag{7}$$

$$NWS_{\min} = \frac{n}{\left\| PRT_U \right\|} \tag{8}$$

$$I_{\max} = \frac{nCT - (\left\| PRT_U \right\| - 1)}{\left\| PRT_U \right\|} \tag{9}$$

$$I_{\min} = 0 \tag{10}$$

Since $\left\| PRT_U \right\| = 4$ in this paper, each *PRT* is generated by:

$$PRT_k = \begin{cases} 3, & 0 < k \le \frac{n}{4} \\ 5, & \frac{n}{4} < k \le \frac{n}{2} \\ 7, & \frac{n}{2} < k \le \frac{3n}{4} \\ 11, & \frac{3n}{4} < k \le n \end{cases} \tag{11}$$

This data set forced the DLBP greedy algorithm to obtain the near-optimal solution of 4 workstations, versus the optimal 3. Unlike the previous case study example however, this data did allow the AEHC to obtain a better-balanced solution nearby and demonstrate hazardous material and high demand item sequencing (Tables 3b, 3c, and 3d).

Table 3. Solution generation to local optima using manufactured data

a)

Original Data												
Part ID	1	2	3	4	5	6	7	8	9	10	11	12
PRT	3	3	3	5	5	5	7	7	7	11	11	11
Hazard	N	N	N	N	N	N	N	N	N	N	N	Y
Demand	0	0	0	0	1	0	0	0	0	0	0	0

b)

Sorted (greedy)												
Part ID	12	11	10	7	8	9	5	4	6	1	2	3
PRT	11	11	11	7	7	7	5	5	5	3	3	3
Hazard	Y	N	N	N	N	N	N	N	N	N	N	N
Demand	0	0	0	0	0	0	1	0	0	0	0	0

c)

DLBP Greedy Solution												
Part ID	12	11	1	10	7	8	9	5	4	6	2	3
PRT	11	11	3	11	7	7	7	5	5	5	3	3
WS	1	1	1	2	2	2	3	3	3	3	3	4
Hazard	Y	N	N	N	N	N	N	N	N	N	N	N
Demand	0	0	0	0	0	0	0	1	0	0	0	0

d)

AEHC Solution (to local optima)												
Part ID	12	11	1	10	5	8	3	7	4	6	2	9
PRT	11	11	3	11	5	7	3	7	5	5	3	7
WS	1	1	1	2	2	2	3	3	3	3	3	4
Hazard	Y	N	N	N	N	N	N	N	N	N	N	N
Demand	0	0	0	0	1	0	0	0	0	0	0	0

The speed was again less than 1/100th of a second for both the DLBP greedy algorithm and the AEHC. Although not an optimal solution, this contrived problem was still successfully solved by the greedy algorithm to within 11% of the optimal number of workstations (when compared to the worst case) and by the single iteration AEHC to within 8.3% of the optimal balance (when compared to the worst case). The problem was solved to within 8.1% of the optimal balance by the DLBP greedy algorithm/AEHC heuristic method (Figure 1) with the AEHC run until no better solutions were found (local optima) which only required one additional iteration.

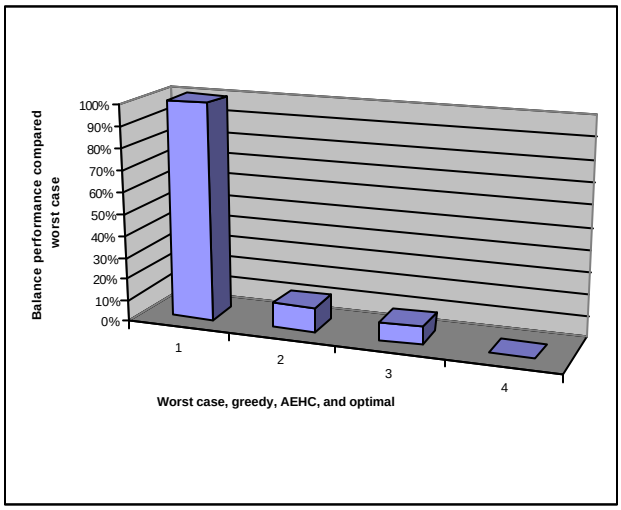

Figure 1. DLBP greedy algorithm/AEHC heuristic performance

Larger *n* or the inclusion of precedence constraints will increasingly move the DLBP greedy algorithm/AEHC heuristic towards the optimal solution. Note that a single search by the AEHC yields a solution effectively as well as the local optima search, but in potentially significantly less time. Also note that the first task listed in the greedy solution is part number 12 (*PRT* of 11 seconds), which is the only part labeled as hazardous (objective 4), and that the first part listed in the greedy solution with a *PRT* of 5 seconds is part number 5, which has the highest demand of the three parts having equal part removal times (objective 5). Though in this example hazardous and high demand item positions are maintained by the AEHC, this may not always be the case. The AEHC is designed to search without regard for hazardous materials since these items would not be expected to drift excessively (due to the adjacent nature of this search) and since the AEHC exclusively seeks the best balance. Although the AEHC also searches without regard for high demand items for the same reasons, their sequence position tends to stay the same or even improve (i.e., move to an earlier removal time in the disassembly sequence) since the higher demand items are initially ahead of the lower demand items (with equal part removal times) from the greedy placement and hence will be the first looked at to be moved forward in the disassembly sequence.

7 Conclusions

A very fast, near-optimal, two-phase approach to the multi-objective DLBP was developed and presented in this

paper. The first phase rapidly provides a feasible solution to the DLBP using a greedy algorithm based on the FFD algorithm and modified to meet multiple objectives. The DLBP greedy algorithm provides a near-optimal minimum number of workstations, with the level of optimality increasing with the number of constraints. The second phase quickly provides a near-optimal and feasible balance sequence using a hill-climbing heuristic referred to as AEHC. AEHC only completes a pair-wise workstation comparison to allow for rapid search and since it is deemed unlikely that tasks several workstations out can be swapped and still preserve precedence of all the tasks in-between. Although a near-optimum technique, the DLBP greedy algorithm/AEHC heuristic quickly found optimal solutions, or solutions within about ten percent of optimal in an exponentially large search space. The AEHC balancing method worked well, generating perfect balance solutions when able, while keeping within ten percent of the optimal balance otherwise. The DLBP greedy algorithm/AEHC heuristic appears well suited to the multi-criteria decision making problem format. Also, the multi-criteria DLBP greedy algorithm and the single iteration AEHC algorithm lend themselves to real-time solution of the DLBP on a dynamic mixed-model and mixed-product disassembly line. In addition, the DLBP greedy algorithm/AEHC heuristic are 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] L. Brennan, S. M. Gupta, and K. N. Taleb, "Operations planning issues in an assembly/disassembly environment", *International Journal of Operations and Production Planning*, Vol 14, No. 9, pp. 57-67, 1994.

[2] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, The MIT Press, Cambridge, MA, 2001.

[3] E. A. Elsayed, and T. O. Boucher, *Analysis and Control of Production Systems*, Prentice Hall, Upper Saddle River, NJ, 1994.

[4] A. Gungor, and S. M. Gupta, "Issues in environmentally conscious manufacturing and product recovery: a survey", *Computers and Industrial Engineering*, Vol 36, No. 4, pp. 811-853, 1999.

[5] A. Gungor, and S. M. Gupta, "Disassembly Line Balancing", *Proceedings of the 1999 Annual Meeting of the Northeast Decision Sciences Institute*, Newport, Rhode Island, March 24-26, pp. 193-195, 1999.

[6] A. Gungor, and S. M. Gupta, "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 29-April 1, pp. 70-73, 1999.

[7] A. Gungor, and S. M. Gupta, "A solution approach to the disassembly line problem in the presence of task failures", *International Journal of Production Research*, Vol 39, No. 7, pp. 1427-1467, 2001.

[8] A. Gungor, and S. M. Gupta, "Disassembly line in product recovery", *International Journal of Production Research*, Vol 40, No. 11, pp. 2569-2589, 2002.

[9] S. M. Gupta, and K. N. Taleb, "Scheduling disassembly", *International Journal of Production Research*, Vol 32, pp. 1857-1866, 1994.

[10] A. A. Hopgood, *Knowledge-based systems for engineers and scientists*, CRC Press, Boca Raton, FL, 1993.

[11] S. M. M^cGovern, S. M. Gupta, and S. V. Kamarthi, "Solving disassembly sequence planning problems using combinatorial optimization", *Proceedings of the 2003 Annual Meeting of the Northeast Decision Sciences Institute*, Providence, Rhode Island, pp. 178-180, March 2003.

[12] Y. Tang, M. Zhou, and R. Caudill, "A Systematic Approach to Disassembly Line Design", *Proceedings of the 2001 IEEE International Symposium on Electronics and the Environment*, Denver, Colorado, May 7-9, pp. 173-178, 2001.