

January 01, 2005

Uninformed and probabilistic distributed agent combinatorial searches for the unary NP-complete disassembly line balancing problem

Surendra M. Gupta
Northeastern University

Seamus M. McGovern
Northeastern University

Recommended Citation

Gupta, Surendra M. and McGovern, Seamus M., "Uninformed and probabilistic distributed agent combinatorial searches for the unary NP-complete disassembly line balancing problem" (2005). . Paper 117. <http://hdl.handle.net/2047/d10009875>

This work is available open access, hosted by Northeastern University.



Laboratory for Responsible Manufacturing

Bibliographic Information

McGovern, S. M. and Gupta, S. M., "Uninformed and Probabilistic Distributed Agent Combinatorial Searches for the Unary NP-Complete Disassembly Line Balancing Problem", *Proceedings of the SPIE International Conference on Environmentally Conscious Manufacturing V*, Boston, Massachusetts, pp. 81-92, October 23-24, 2005.

Copyright Information

Copyright 2005, Society of Photo-Optical Instrumentation Engineers.

This paper was published in Proceedings of SPIE (Volume 5997) and is made available as an electronic reprint with permission of SPIE. One print or electronic copy may be made for personal use only. Systematic or multiple reproduction, distribution to multiple locations via electronic or other means, duplication of any material in this paper for a fee or for commercial purposes, or modification of the content of the paper are prohibited.

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**

Uninformed and Probabilistic Distributed Agent Combinatorial Searches for the Unary NP-Complete Disassembly Line Balancing Problem

Seamus M. McGovern and Surendra M. Gupta^{*}
Laboratory for Responsible Manufacturing
334 SN, Department of MIE
Northeastern University
360 Huntington Avenue
Boston, MA, 02115 USA

ABSTRACT

Disassembly takes place in remanufacturing, recycling and disposal, with a line 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 disassembly line balancing problem, which is proven here to belong to the class of unary NP-complete problems. Probabilistic (ant colony optimization) and uninformed (H-K) search methods are presented and compared. Numerical results are obtained using a recent case study to illustrate the search implementations and compare their performance. Conclusions drawn include the consistent generation of near-optimal solutions, the ability to preserve precedence, the speed of the techniques, and their practicality due to ease of implementation.

Keywords: Disassembly, disassembly line balancing, combinatorial optimization, H-K general-purpose heuristic, ant colony optimization, complexity theory, product recovery

1. INTRODUCTION

More and more manufacturers are acting to recycle and remanufacture their post-consumed products due to new and more rigid environmental legislation, increased public awareness, and extended manufacturer responsibility. In addition, the economic attractiveness of reusing products, subassemblies and 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.

In order to minimize the amount of waste sent to landfills, product recovery seeks to obtain materials and parts from old or outdated products through recycling and remanufacturing – this includes the 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 defined as the methodical extraction of valuable parts/subassemblies and materials from discarded products through a series of operations. 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.

Recently, disassembly has 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

^{*} gupta@neu.edu; phone 1 617 373-4846; fax 1 617 373-2921; URL <http://www.coe.neu.edu/~smgupta/>

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. For example, an automobile slated for disassembly contains a variety of parts that are dangerous to remove and/or present a hazard to the environment such as the battery, airbags, fuel, and oil. Various demand sources may also lead to complications in disassembly line balancing. The reusability of parts creates a demand for these parts, however, the demands and availability of the reusable parts is significantly less predictable than what is found in the assembly process. Most products contain parts that are installed (and must be removed) in different attitudes, from different areas of the main structure, or in different directions. Since any required directional changes increases the setup time for the disassembly process, it is desirable to minimize the number of directional changes in the final disassembly sequence. Finally, 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.

This paper first defines the disassembly line balancing problem (DLBP) then proves for the first time that it is NP-complete in the strong sense, 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 two different combinatorial optimization methods. These include a probabilistic distributed intelligent agent metaheuristic (Ant Colony Optimization or ACO) and an uninformed deterministic search methodology (Hunter-Killer or H-K). The application of these techniques is instrumental in rapidly obtaining solutions to the DLBP's intractably large solution space. The ACO considered here is an ant system algorithm known as the ant-cycle model [1]. A DLBP implementation of the H-K general-purpose optimization heuristic – a deterministic search technique based on an enhanced exhaustive search – is then applied and compared. Both techniques seek to preserve precedence relationships. A case study example is considered to illustrate implementation of the methodologies. The conclusions drawn from the study include the consistent generation of near-optimal solutions, the ability to preserve precedence relationships, the speed of the methods, and their practicality due to the ease of implementation in solving the DLBP.

2. LITERATURE REVIEW

Product recovery involves a number of steps [2]. 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 [3], [4]. 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], [7] 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 [8]. McGovern and Gupta developed a greedy/2-Opt hybrid algorithm for the DLBP [9] and a greedy/hill climbing hybrid algorithm [10]. Kongar and Gupta applied a genetic algorithm to the disassembly sequencing problem [11]. McGovern *et al.* first considered combinatorial optimization techniques for the DLBP [12].

3. NOTATION

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

\leq_p	polynomial-time transformation
$<$	partial order; i.e., x precedes y is written $x < y$
α	weight of existing pheromone (trail) in path selection
β	weight of the edges in path selection

η_{pq}	ACO visibility value of edge (directed edge for DLBP) pq at time t
ρ	variable such that $1 - \rho$ represents the pheromone evaporation rate
$\tau_{pq}(NC)$	amount of trail on edge pq (directed edge for DLBP) during cycle NC
ψ_k	k^{th} element's skip measure (i.e., for the solution's third element, visit every 2^{nd} possible task if $\psi_3 = 2$); subscript is left blank if k is constant for all elements
a	MULTIPROCESSOR SCHEDULING problem task variable
A	MULTIPROCESSOR SCHEDULING problem task set
B	MULTIPROCESSOR SCHEDULING problem deadline bound
c	initial amount of pheromone on each of the paths
CT	cycle time; maximum time available at each workstation
d_k	demand; quantity of part k requested
D	demand rating for a given solution sequence; also, demand bound for the decision version of DLBP
F	measure of balance for a given solution sequence
F_{nr}	measure of balance of ant r 's sequence at time n (the end of a tour)
F_{tr}	measure of balance of ant r 's sequence at time t
h_k	binary value; 1 if part k is hazardous, else 0
H	hazard rating for a given solution sequence; also, hazard bound for the decision version of DLBP
I	total idle time for a given solution sequence
j	workstation count ($1, \dots, NWS$)
k	part identification or sequence position ($1, \dots, n$)
$l(a)$	MULTIPROCESSOR SCHEDULING problem task length
L_r	ACO delta trail divisor value; set equal to F_{nr} for DLBP
m	number of ants; also, number of processors in the MULTIPROCESSOR SCHEDULING problem
n	number of part removal tasks
\mathbf{N}	the set of natural numbers ($0, 1, 2, \dots$)
NC_{max}	maximum number of cycles for ACO
NWS	number of workstations required for a given solution sequence
NWS^*	optimum (minimum) number of workstations
NWS_{nom}	worst-case (maximum) number of workstations
p	edge variable
P	the set of part removal tasks
$p_{pq}^r(t)$	probability of ant r taking an edge (directed edge for DLBP) pq at time t during cycle NC
PRT_k	part removal time required for k^{th} task
PS_k	k^{th} part in a solution sequence (i.e., for solution "3, 1, 2" $PS_2 = 1$)
q	edge variable
Q	amount of pheromone added if a path is selected
r	ant count
r_k	integer value corresponding to part removal direction
R	direction rating for a given solution sequence; also, direction bound for the decision version of DLBP
R_k	binary value; 0 if part k can be removed in the same direction as part $k + 1$, else 1
ST_j	station time; total processing time requirement in workstation j
t	time within a cycle; ranges from 0 to n
V	maximum range for a workstation's idle time
x	counter variable
y	counter variable
\mathbf{Z}	the set of all integers ($\dots, -2, -1, 0, 1, 2, \dots$)
\mathbf{Z}^+	the set of positive integers ($1, 2, \dots$)

4. THE DLBP MODEL DESCRIPTION

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

1. Minimize the number of disassembly workstations and hence, minimize the total idle time.
2. Ensure the idle times at each workstation are similar.

3. Remove hazardous parts early in the disassembly sequence.
4. Remove high demand parts before low demand parts.
5. Minimize the number of direction changes required for disassembly.

A major constraint is the requirement to provide a feasible disassembly sequence for the product being investigated. The result is an integer, deterministic, multiple-criteria decision-making problem with an exponential search space. 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 and is described by

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

which attains objective 1.

Line balancing seeks to achieve “perfect balance” (all idle times equal to zero). When this is not achievable, either Line Efficiency (LE) or the Smoothness Index (SI) is often used as a performance evaluation tool [13]. We use a measure of balance that combines the two and is easier to calculate. SI rewards similar idle times at each workstation, but at the expense of allowing for a large (sub-optimal) 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 [14]. The balancing method simultaneously minimizes the number of workstations while aggressively ensuring that idle times at each workstation are similar, though at the expense of the generation of a nonlinear objective function. The method is computed based on the number of workstations required as well as the sum of the square of the idle times at all of 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. The measure of balance is represented as

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

with perfect balance is indicated by $F = 0$. Note that mathematically, formula (2) effectively makes formula (1) redundant due to the fact that it concurrently minimizes the number of workstations.

In addition, we find

$$NWS^* = \left\lceil \frac{\sum_{k=1}^n PRT_k}{CT} \right\rceil \quad (3)$$

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

A hazard measure has been developed and is represented as

$$H = \sum_{k=1}^n (k \cdot h_{PS_k}), \quad h_{PS_k} = \begin{cases} 1, & \text{hazardous} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

The demand measure can be represented as

$$D = \sum_{k=1}^n (k \cdot d_{PS_k}), \quad d_{PS_k} \in \mathbb{N}, \forall PS_k \quad (6)$$

McGovern and Gupta [9] provide further explanation of formulae (1) through (6). A measure based on a count of the direction changes has also been developed. Part removal directions are expressed as

$$r_{PS_k} = \begin{cases} +1, & \text{direction} & : +x \\ -1, & \text{direction} & : -x \\ +2, & \text{direction} & : +y \\ -2, & \text{direction} & : -y \\ +3, & \text{direction} & : +z \\ -3, & \text{direction} & : -z \end{cases} \quad (7)$$

The direction measure is then represented as

$$R = \sum_{k=1}^{n-1} R_k, \quad R_k = \begin{cases} 1, & r_{PS_k} \neq r_{PS_{k+1}} \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

The overall goal is to minimize each of these measures, with a priority of F , then H , then D , then R .

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

Problem assumptions include the following:

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

5. UNARY NP-COMPLETENESS PROOF

The decision version of DLBP has been shown to be NP-complete [15]. DLBP is also unary NP-complete (often referred to as “NP-complete in the strong sense”). Using the concise style of Garey and Johnson [16] this is shown in the following proof by restriction to MULTIPROCESSOR SCHEDULING. MULTIPROCESSOR SCHEDULING is described by:

INSTANCE: A finite set A of tasks, a length $l(a) \in \mathbb{Z}^+$ for each $a \in A$, a number $m \in \mathbb{Z}^+$ of processors, and a deadline $B \in \mathbb{Z}^+$.

QUESTION: Is there a partition $A = A_1 \cup A_2 \cup \dots \cup A_m$ of A into m disjoint sets such that

$$\max \left\{ \sum_{a \in A_i} l(a) : 1 \leq i \leq m \right\} \leq B ?$$

In DLBP, NWS is equivalent to m in MULTIPROCESSOR SCHEDULING, while CT is equivalent to B . The following theorem provides the proof:

Theorem 1 DLBP is unary NP-complete.

INSTANCE: A finite set P of tasks, partial order $<$ on P , task time $PRT_k \in \mathbf{Z}^+$, hazardous part binary value $h_k \in \{0, 1\}$, part demand $d_k \in \mathbf{N}$, and part removal direction $r_k \in \mathbf{Z}$ for each $k \in P$, workstation capacity $CT \in \mathbf{Z}^+$, number $NWS \in \mathbf{Z}^+$ of workstations, difference between largest and smallest idle time $V \in \mathbf{N}$, hazard measure $H \in \mathbf{N}$, demand measure $D \in \mathbf{N}$, and direction change measure $R \in \mathbf{N}$.

QUESTION: Is there a partition of P into disjoint sets P_A, P_B, \dots, P_{NWS} such that the sum of the sizes of the tasks in each P_X is CT or less, the difference between largest and smallest idle times is V or less, the sum of the hazardous part binary values multiplied by their sequence position is H or less, the sum of the demanded part values multiplied by their sequence position is D or less, the sum of the number of part removal direction changes is R or less, and it obeys the precedence constraints?

Proof: DLBP \in NP. Given an instance, it can be verified in polynomial time if the answer is “yes” by: counting the number of disjoint sets and showing that they are NWS or less, summing the sizes of the tasks in each disjoint set and showing that they are CT or less, examining each of the disjoint sets and noting the largest and the smallest idle times then subtracting the smallest from the largest and showing that this value is less than V , summing the hazardous part binary values multiplied by their sequence position and showing this is H or less, summing the demanded part values multiplied by their sequence position and showing that this is D or less, summing the number of changes in part removal direction and showing that this is R or less, and checking that each task has no predecessors listed after it in the sequence.

MULTIPROCESSOR SCHEDULING \leq_p DLBP. Restrict to MULTIPROCESSOR SCHEDULING by allowing only instances in which $V = CT$, $<$ is empty, and $h_x = h_y$, $d_x = d_y$, $r_x = r_y \forall x, y$.

Therefore, DLBP is unary NP-complete. \square

6. PRECEDENCE RELATIONSHIP DIAGRAM FORMAT FOR DISASSEMBLY PROBLEMS

This paper proposes a modified disassembly task/part node representation for use in precedence diagrams. It is based on the familiar format found in assembly line balancing problems (as depicted in Elsayed and Boucher [13]), which consists of a circular symbol with the task identification number in the center and the part removal time listed outside and above. This is selectively enhanced to provide greater usability in application to disassembly problems. The part removal time is kept outside of the circle, but in the upper right corner. Working around in a clockwise manner, the additional considerations found in disassembly are listed using, as a default, the order of priority as given in this paper; i.e., “H” (only if the part is labeled as hazardous, else blank), the part’s demand, and the part’s removal direction (Fig. 1).

This representation provides a reference to all of a given disassembly problem’s quantitative data in one location while still enabling all of the graphical relationship information that a traditional precedence diagram portrays. Edges continue to be depicted as solid lines terminated with arrows. The arrow’s direction points (top to bottom, or left to right) towards subsequent disassembly tasks; i.e., the arrows leave the predecessor parts. OR relations are connected to each other with a broken line. AND and OR relationships are depicted in Fig. 2; i.e., remove (1AND2) OR (2AND3) prior to 4 (note that, for example, part 1 takes 2 seconds to remove, is hazardous, has a demand of 7 and can only be removed in direction $-y$).

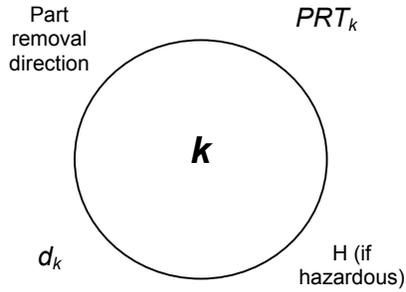


Fig. 1. Proposed disassembly task/part node representation.

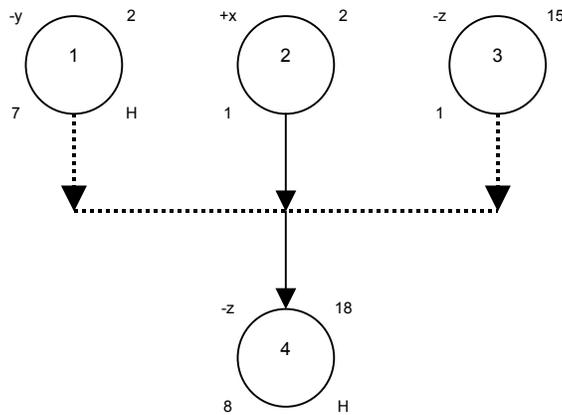


Fig. 2. AND and OR example depicting the requirement to remove (1AND2) OR (2AND3) prior to 4.

7. THE H-K GENERAL-PURPOSE HEURISTIC

A recently described search approach, H-K, was adapted for use with the DLBP. This search technique provides a near-optimal solution to combinatorial optimization problems using a modified exhaustive search method that provides a deterministic (in H-K's most general form) data sampling of all solutions.

7.1. H-K Heuristic Background and Motivation

In some physical search applications (anti-submarine warfare, search and rescue) exhaustive search is not possible due to time or sensor limitations. In these cases, it becomes necessary to sample the search space and operate under the assumption that the best point found is either the optimal point or is reasonably near. First introduced by McGovern and Gupta [17], the H-K general-purpose heuristic works by sampling the exhaustive solution set. It searches the solution space in a method similar to exhaustive search but using a pattern that skips some solutions to significantly minimize the search space (Fig. 3).

Once the solution is generated, it can be further refined by performing subsequent local searches (such as 2-Opt or smaller, localized H-K searches; i.e., using the solution as the starting point and decreasing the skip size, ψ , while limiting the range of each element in the subsequent search). Depending on the application, H-K can be run once, multiple times from the same starting point using a different ψ , multiple times from a different starting point using the same ψ , or followed up with a differing local search on the best or on several of the best solutions generated.



Fig. 3. Exhaustive and H-K search space.

In this paper, H-K is run alone to a single-phase solution with multiple applications from the same starting point using different ψ s. Because H-K is exceptionally deterministic and performs no preprocessing of the data, the order in which the data is presented can be expected to affect the solutions generated. For this reason, the analysis done in this paper was run twice; first with the data in the sequence that it has been provided in, and then run again but this time with the data presented in reverse order. The better of the two results was then maintained as the sole, best solution. The multiple applications used all skip sizes of $n - 20$ and larger, i.e., $n - 20 \leq \psi \leq n - 1$ ($\psi = n$ is trivial). In this basic form, there is one H-K process run on forward and reverse data, multiple times using different ψ values. The forward starting point is represented by the part solution sequence n -tuple $PS_k = \langle 1, 1, 1, \dots, 1 \rangle$ (given $k = \{1, 2, 3, \dots, n\}$). Note that since the solution set is a permutation, there are no repeated items, therefore the starting point is effectively $PS_k = \langle 1, 2, 3, \dots, n \rangle$. The reverse starting point is represented by $PS_k = \langle n, n, n, \dots, n \rangle$ (effectively $PS_k = \langle n, n - 1, n - 2, n - 3, \dots, 1 \rangle$). To summarize, the H-K used here has two constant (vs. randomized) starting points (i.e., forward and reverse), constant skip type (i.e., each element in the sequence is skipped in the same way), constant skip size at each solution element, varying skip size ($5 \leq \psi \leq 24$) between runs, and no follow-on solution refinement.

7.2. H-K Process and DLBP Application

The general process works as follows: in the basic H-K, and with $\psi = 2$, the first element in the initial solution sequence would be 1. In the next element position, 1 would be considered, but since it is already in the solution, the second element's value would be incremented by one, and so 2 would be considered and be acceptable. This is repeated for all of n element positions until the first solution is generated. It then repeats in a nested fashion, attempting to increment all solution elements by ψ , until the first element is the only one remaining to be incremented. At that point, the first element under consideration would be incremented by 2 (since $\psi = 2$) and therefore, 3 would be considered and inserted in the first element position. Since 1 is not yet in the sequence, it would be placed in the second position, 2 in the third, 4 in the fourth, etc. For DLBP H-K these evolutions are further modified to test the proposed sequence for precedence constraints. This is performed as each element is incremented. If all remaining parts for a given element position fail these precedence checks, the remainder of the positions are not further inspected, the procedure falls back to the previously successful solution element addition, increments it by 1, and continues. These processes are repeated until all allowed items have been visited in the first element position (and by default, due to the nested nature of the search, all subsequent solution positions). For example, with $n = 4$, $PRT_k \in \{1, 2, 3, 4\}$ and no precedence constraints, instead of considering the $4! = 24$ possible permutations, only five are considered by the single-phase H-K with $\psi = 2$ and using forward-only data: $PS_k = \langle 1, 2, 3, 4 \rangle$, $PS_k = \langle 1, 4, 2, 3 \rangle$, $PS_k = \langle 3, 1, 2, 4 \rangle$, $PS_k = \langle 3, 1, 4, 2 \rangle$, and $PS_k = \langle 3, 4, 1, 2 \rangle$. All of the tasks are maintained in a Tabu-type list to ensure no sequences are visited multiple times. After a feasible solution sequence is generated, DLBP H-K places each element into a workstation using the Next-Fit rule from bin packing. DLBP H-K considers at each element in the solution and puts that element into the current workstation if it fits; if it does not fit, a new workstation is assigned. When all of the work elements have been assigned to a workstation, the process is complete and the balance, hazard, demand, and direction measures are calculated. The best of all of the inspected solution sequences is then saved as the problem solution.

8. THE ACO METAHEURISTIC

8.1. ACO Model Description

ACO is a probabilistic evolutionary algorithm based on a distributed autocatalytic process that makes use of agents called ants. Just as a colony of ants can find the shortest distance to a food source, in ACO they work cooperatively

towards a solution to a combinatorial optimization problem. Ants are placed at multiple starting nodes, such as cities for the traveling salesman problem (TSP). Each of the m ants is allowed to visit all remaining unvisited edges as indicated by the previously described Tabu-type list. Each ant's possible subsequent steps (from some node p to a node q giving edge pq) are evaluated for desirability and each is assigned a proportionate probability given by formula (9). Based on these, the next edge is randomly selected for each ant. After completing an entire tour, all feasible ants are given the equivalent of additional pheromone (in proportion to tour desirability) that is added to each step visited by that ant. All paths are then decreased in their pheromone strength according to a measure of evaporation. This process is repeated for NC_{max} or until stagnation behavior (where all ants make the same tour).

8.2. DLBP-Specific Modifications

DLBP ACO is modified to account for feasibility constraints and provide for multiple objectives [18]. In DLBP ACO, each task is a node, with the number of ants initially being set equal to the number of tasks, and one ant on each task. Each ant is allowed to visit all tasks not already in its solution. Each ant's possible subsequent steps are evaluated for feasibility and the measure of balance then assigned a proportionate probability. Infeasible steps receive a probability of 0. Any ants having only infeasible subsequent task options are effectively ignored for the remainder of the cycle. At tour completion, the best solution found is saved and the process is repeated. Equivalent solutions are further evaluated based on hazard, demand, and direction measures. Due to the nature of the DLBP (i.e., whether or not a task should be added to a workstation depends on that workstation's idle time and precedence constraints), the probability in formula (9) is not calculated once, at the beginning at each tour as is typical with ACO, but is calculated dynamically, generating new probabilities at each increment of t . The probability of ant r taking edge pq at time t during cycle NC is

$$p_{pq}^r(t) = \begin{cases} \frac{[\tau_{pq}(NC)]^\alpha \cdot [\eta_{pq}(t)]^\beta}{\sum_{r \in allowed_r} [\tau_{pr}(NC)]^\alpha \cdot [\eta_{pr}(t)]^\beta} & \text{if } q \in allowed_r \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

The amount of trail on each edge is calculated after each cycle using

$$\tau_{pq}(NC+1) = \rho \cdot \tau_{pq}(NC) + \Delta \tau_{pq} \quad (10)$$

where

$$\Delta \tau_{pq} = \sum_{r=1}^m \Delta \tau_{pq}^r, \text{ and: } \Delta \tau_{pq}^r = \begin{cases} \frac{Q}{L_r} & \text{edge}(p,q) \text{ used} \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

For DLBP ACO the visibility η_{pq} is also calculated dynamically in each cycle and is defined as

$$\eta_{pq} = \frac{1}{F_{tr}} \quad (12)$$

In formula (11) L_r is set to F_{nr} , where a small final value for ant r 's measure of balance at time n (the end of the tour) provides a large measure of trail added to each edge. Though L_r and η_{pq} are related here, this is not unusual in ACO applications (in TSP, L_r is the tour length, while η_{pq} is the reciprocal of the distance between cities p and q). However, this method of selecting η_{pq} (effectively a short-term greedy choice) may not always translate into the best final solution for a complex problem like DLBP. Also, since all edges are directed edges in DLBP (unlike the undirected edges in TSP), pq is evaluated separately from qp . In addition, NC_{max} is set to 300 since larger problems than that studied here were shown to reach their best solution by that count. The process is not run until no improvements were shown but, as is the norm with many combinatorial optimization techniques, is run continuously until NC_{max} [19]. This also enabled

the probabilistic component of ACO an opportunity to leave any local minima. Per the best ACO performance experimentally determined by Dorigo *et al.* [20], the following is used: $\alpha = 1.00$, $\beta = 5.00$, $\rho = 0.50$, $Q = 100.00$, and $c = 1.00$.

9. NUMERICAL RESULTS

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.

9.1. Case Study

Both combinatorial optimization techniques were used to provide a solution to the disassembly line balancing problem based on the disassembly sequencing problem presented by Gupta *et al.* [21]. Here, the objective is to completely disassemble a cell phone consisting of $n = 25$ components with several precedence relationships. The data set includes a disassembly line operating at a speed which allows $CT = 18$ seconds for each workstation to perform its required disassembly tasks. The data for the problem can be found summarized in table 1.

Table 1. SCH-3500 cell phone parts and their properties

task	part	PRT_k	hazard	d_k	direction	predecessor
1	antenna	3	Y	4	+y	
2	battery	2	Y	7	-y	
3	antenna guide	3		1	-z	1,2
4	bolt (type1) a	10		1	-z	
5	bolt (type1) b	10		1	-z	
6	bolt (type2) 1	15		1	-z	2
7	bolt (type2) 2	15		1	-z	2
8	bolt (type2) 3	15		1	-z	2
9	bolt (type2) 4	15		1	-z	3
10	clip	2		2	+z	4,5
11	rubber seal	2		1	+z	10
12	speaker	2	Y	4	+z	11
13	white cable	2		1	-z	6,7,8,9
14	red/blue cable	2		1	+y	6,7,8,9
15	orange cable	2		1	+x	6,7,8,9
16	metal top	2		1	+y	6,7,8,9
17	front cover	2		2	+z	13,14
18	back cover	3		2	-z	15
19	circuit board	18	Y	8	-z	13,14,16,18
20	plastic screen	5		1	+z	17
21	keyboard	1		4	+z	17
22	LCD	5		6	+z	21
23	sub-keyboard	15	Y	7	+z	16,21
24	internal IC	2		1	+z	19,23
25	microphone	2	Y	4	+z	21

9.2. DLBP ACO and DLBP H-K Performance

DLBP ACO and DLBP H-K were both written in C++ and run on a 1.6GHz PM x86 family workstation. For the results in table 2, DLBP H-K was run three times to obtain an average computation time, while DLBP ACO was run five times (unlike H-K, ACO is highly probabilistic, so additional runs were conducted to achieve an average measure of performance, as well as best case and worst case).

DLBP H-K with $5 \leq \psi \leq 24$ and $n = 25$ was very quick, taking less than 5% of the time of DLBP ACO. DLBP ACO regularly generated the optimal number of workstations and appears to have averaged very close to optimal in F (due to

the size and nature of the cell phone problem, no optimal solutions have been proven to be found to date). DLBP H-K performed slightly better in terms of hazardous part placement, while DLBP ACO was slightly better at removing high demand parts early. The number of part removal directions was seen to be similar for both. For a more in-depth study of these solution methods and analysis on larger and more diverse problem sets, see [22] and [23].

Table 2. Comparison of the averaged H-K and ACO solutions for the cell phone example

Solution	Time	NWS	F	H	D	R
H-K	0.050	10	137	84	943	12
ACO	1.3392	9.0	10.2	87.2	924.4	12.0

DLBP ACO typically found feasible solutions to the cell phone data set in just over one second. Due to its probabilistic component, DLBP ACO was seen to select a solution sub-optimal in F about 40% of the time (with the assumption that $F = 9$ is optimal [21]) over ten runs. This can be attributed to formula (9) and ACO's requirement to evaluate partial solutions before making a solution element selection decision. In addition, it was observed that edges not selected are diluted very rapidly; it was not unusual to see the bulk of the edges (all initialized to $c = 1$) with trail values of 1×10^{-91} after the 300 cycles. DLBP H-K with $\psi = \{5, \dots, n - 1\}$ and forward and reverse data found its best feasible solution extremely quickly, regularly taking less than one tenth of a second. 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 cell phone problem to optimality using exhaustive search would take over 250-million years.

Although the preceding results are not optimal, this is more a reflection of the challenges posed even by seemingly simple disassembly problems more than it is an indication of any limitations of the search techniques. Note that the inclusion of additional precedence constraints will increasingly move both DLBP ACO and DLBP H-K towards the optimal solution.

10. 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 end-of-life problems, with one using a distributed agent ant system/ant-cycle model algorithm based on ACO and the other, an uninformed deterministic search. DLBP ACO and DLBP H-K were able to generate feasible sequences having a near-optimal measure of balance, while addressing hazardous materials, high demand parts, and a multitude of part removal directions. These diverse solution-generating techniques appear well suited to the multi-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 – a requirement of many disassembly problems – that generally do not lend themselves to application of traditional mathematical programming techniques.

REFERENCES

- [1] M. Dorigo and G. Di Caro, "The Ant Colony Optimization Meta-Heuristic," *New Ideas in Optimization*, McGraw-Hill, Maidenhead, UK, 1999.
- [2] 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.
- [3] 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.
- [4] S. M. Gupta and K. N. Taleb, "Scheduling Disassembly," *International Journal of Production Research*, vol. 32, pp. 1857-1866, 1994.
- [5] A. Gungor and S. M. Gupta, "Disassembly Line Balancing," *Proceedings of the 1999 Annual Meeting of the Northeast Decision Sciences Institute*, March 1999, Newport, Rhode Island, pp. 193-195.

- [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*, March 1999, New Orleans, Louisiana, pp. 70-73.
- [7] A. Gungor, and S. M. Gupta, "Disassembly Line in Product Recovery," *International Journal of Production Research*, vol. 40, no. 11, pp. 2569-2589, 2002.
- [8] 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.
- [9] S. M. McGovern and S. M. Gupta, "2-Opt Heuristic for the Disassembly Line Balancing Problem," *Proceedings of the 2003 SPIE International Conference on Environmentally Conscious Manufacturing III*, October 2003, Providence, Rhode Island, pp. 71-84.
- [10] S. M. McGovern and S. M. Gupta, "Greedy Algorithm for Disassembly Line Scheduling," *Proceedings of the 2003 IEEE International Conference on Systems, Man, and Cybernetics*, October 2003, Washington, DC, pp. 1737-1744.
- [11] E. Kongar and S. M. Gupta, "A Genetic Algorithm for Disassembly Process Planning," *Proceedings of the 2001 SPIE International Conference on Environmentally Conscious Manufacturing II*, October 2001, Newton, Massachusetts, pp. 54-62.
- [12] S. M. McGovern, 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*, March 2003, Providence, Rhode Island, pp. 178-180.
- [13] E. A. Elsayed and T. O. Boucher, *Analysis and Control of Production Systems*, Prentice Hall, Upper Saddle River, NJ, 1994.
- [14] S. M. McGovern and S. M. Gupta, "Metaheuristic Technique for the Disassembly Line Balancing Problem," *Proceedings of the 2004 Northeast Decision Sciences Institute Conference*, March 2004, Atlantic City, New Jersey, pp. 223-225.
- [15] S. M. McGovern and S. M. Gupta, "Combinatorial Optimization Methods for Disassembly Line Balancing," *Proceedings of the 2004 SPIE International Conference on Environmentally Conscious Manufacturing IV*, Philadelphia, Pennsylvania, October 2004, pp. 53-66.
- [16] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, NY, 1979.
- [17] S. M. McGovern and S.M. Gupta, "Demufacturing Strategy based upon Metaheuristics," *Proceedings of the 2004 IIE Industrial Engineering Research Conference*, May 2004, Houston, Texas, CD-ROM.
- [18] S. M. Gupta and S. M. McGovern, "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*, April 2004, Cancun, Mexico, CD-ROM.
- [19] A. A. Hopgood, *Knowledge-Based Systems for Engineers and Scientists*, CRC Press, Boca Raton, FL, 1993.
- [20] M. Dorigo, V. Maniezzo, and A. Colomi, "The Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, vol. 26, no. 1, pp. 1-13, 1996.
- [21] S. M. Gupta, E. Evren, and S. M. McGovern, "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, pp. 43-52.
- [22] S. M. McGovern and S. M. Gupta, "Multi-Criteria Ant System and Genetic Algorithm for End-Of-Life Decision Making," *Proceedings of the 35th Annual Meeting of the Decision Sciences Institute*, November 2004, Boston, Massachusetts, pp. 223-225.
- [23] S. M. McGovern, S. M., Gupta, and K. Nakashima, "Multi-Criteria Optimization for Non-Linear End of Lifecycle Models," *Proceedings of the Sixth Conference on EcoBalance*, Tsukuba, Japan, October 2004, pp. 201-204.