Multi-Objective Optimization in Computational Intelligence: Theory and Practice

Lam Thu Bui
University of New South Wales, Australia

Sameer Alam
University of New South Wales, Australia

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE

Hershey · New York

Acquisitions Editor:	Kristin Klinger
Development Editor:	Kristin Roth
Senior Managing Editor:	Jennifer Neidig
Managing Editor:	Jamie Snavely
Assistant Managing Editor:	Carole Coulson
Copy Editor:	Erin Meyer
Typesetter:	Amanda Appicello
Cover Design:	Lisa Tosheff
Printed at:	Yurchak Printing Inc.

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

and in the United Kingdom by
Information Science Reference (an imprint of IGI Global)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: http://www.eurospanbookstore.com

Library of Congress Cataloging-in-Publication Data

Multi-objective optimization in computational intelligence : theory and practice / Lam Thu Bui and Ricardo Sameer Alam, editors.

p. cm.

Summary: “This book explores the theoretical, as well as empirical, performance of MOs on a wide range of optimization issues including combinatorial, real-valued, dynamic, and noisy problems. It provides scholars, academics, and practitioners with a fundamental, comprehensive collection of research on multi-objective optimization techniques, applications, and practices”--Provided by publisher.

ISBN-13: 978-1-59904-498-9 (hardcover)

ISBN-13: 978-1-59904-500-9 (e-book)

1. Computational intelligence. 2. Evolutionary computation. 3. Mathematical optimization. 4. Artificial intelligence. I. Bui, Lam Thu. II. Alam, Ricardo Sameer.

Q342.M85 2008

519.6--dc22

2007040640

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter VI
Lexicographic Goal Programming and Assessment Tools for a Combinatorial Production Problem

Seamus M. McGovern
U.S. DOT National Transportation Systems Center, USA

Surendra M. Gupta
Northeastern University, USA

ABSTRACT

NP-complete combinatorial problems often necessitate the use of near-optimal solution techniques including heuristics and metaheuristics. The addition of multiple optimization criteria can further complicate comparison of these solution techniques due to the decision-maker's weighting schema potentially masking search limitations. In addition, many contemporary problems lack quantitative assessment tools, including benchmark data sets. This chapter proposes the use of lexicographic goal programming for use in comparing combinatorial search techniques. These techniques are implemented here using a recently formulated problem from the area of production analysis. The development of a benchmark data set and other assessment tools is demonstrated, and these are then used to compare the performance of a genetic algorithm and an H-K general-purpose heuristic as applied to the production-related application.

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. A crucial first step is disassembly. Disassembly is defined as the methodical extraction of valuable parts, subassemblies, and materials from discarded products through a series of operations. Recently, disassembly has gained a great deal of attention in the literature due to its role in environmentally conscious manufacturing. A disassembly line 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. These many challenges are reflected in its formulation as a multicriteria decision making problem.

Line balancing (ordering assembly/disassembly tasks on a line to achieve some objective) is critical in minimizing the use of valuable resources (e.g., time and money) invested and in maximizing the level of automation and the quality of parts or materials recovered (Figure 1). The Disassembly Line Balancing Problem (DLBP) seeks a sequence of parts for removal from an end of life product that minimizes the resources for disassembly and maximizes the automation of the process and the quality of the parts or materials recovered. This chapter first mathematically models the multicriteria DLBP, which belongs to the class NP-complete, necessitating use of specialized solution techniques. Combinatorial optimization is an emerging field that combines techniques from applied mathematics, operations research, and computer science to solve optimization problems over discrete structures. Due to the suboptimal nature of these searches, a method is needed to access different combinatorial optimization techniques. Lexicographic goal programming is proposed to provide a hierarchical search structure, while quantitative tools including a benchmark data set are introduced. The DLBP is then solved using two combinatorial optimization methods: a genetic algorithm (GA) and the hunter-killer (H-K) general-purpose heuristic.

LITERATURE REVIEW

Key to addressing any engineering problem is to understand how complex or easy it is, what it shares with similar problems, and appropriate methods to obtain reasonable solutions. For these reasons, a background in optimization and algorithms is valuable. Tovey (2002) provides a well-structured review of complexity, NP-hardness, NP-hardness proofs (including the concise style of Garey & Johnson, 1979), typical NP-hard problems, the techniques of specialization, forcing, padding, and gadgets, mathematical programming versus heuristics, and other complexity classifications. Rosen (1999) provides a useful text in the general area of discrete mathematics including set theory, logic, algorithms, graph theory, counting, set theory and proofs. Papadimitriou and Steiglitz (1998) is the de-facto text on combinatorial optimization as is Garey and Johnson (1979) in the area of NP-completeness. Holland (1975) is credited with developing the genetic algorithm. Osman and Laporte (1996) provide a well-researched paper on all forms of metaheuristics, the basic concepts of each, and references to applications. A follow-on paper by Osman (2004) is more compact and also more current.

A major part of manufacturing and assembly operations, the *assembly line* is a production line where material moves continuously at a uniform rate through a sequence of workstations where assembly work is performed. With research papers going back to the 1950's, the Assembly Line Balancing problem is well defined and fairly well understood. While having significant differences from assembly line balancing, the recent development of DLBP requires that related problems be fully investigated and understood in order to better define DLBP and to obtain guidance in the search for appropriate methodologies to solve it. Gutjahr and Nemhauser (1964) first described a solution to the Assembly Line Balancing problem, while Erel and Gokcen (1964) developed a modified version by allowing for *mixed-model* lines (assembly lines used to assemble different models of the same product). Suresh, Vinod, and Sahu (1996) first presented a genetic algorithm to provide a near-optimal solution to the Assembly Line Balancing problem. *Tabu search* is used in balancing assembly lines in Lapierre, Ruiz, and Soriano (2006) using SALB-I with instances from the literature (*Arcus 1* and *2*) and a case study from industry. Hackman, Magazine, and Wee (1989) proposed a *branch-and-bound* heuristic for the SALB-I problem. Ponnambalam, Aravindan, and Naidu (1999) compared line-balancing heuristics with a quantitative evaluation of six assembly line balancing techniques.

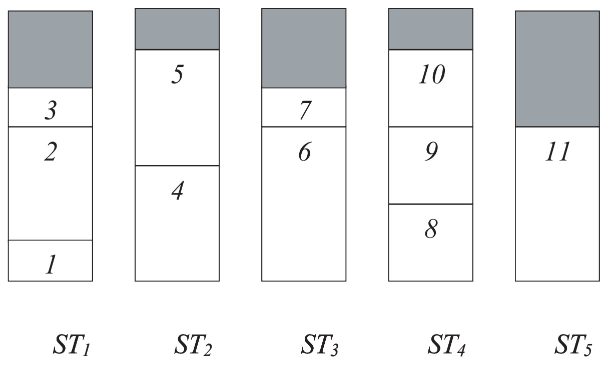
Many papers have discussed the different aspects of product recovery. Brennan, Gupta, and Taleb (1994) and Gupta and Taleb (1994) investigated the problems associated with disassembly planning and scheduling. Torres, Gil, Puente, Pomares, and Aracil (2004) reported a study for nondestructive automatic disassembly of personal computers. Gungor and Gupta (1999b, 1999c, 2002, 2001) presented the first introduction to disassembly line balancing 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. For a review of environmentally conscious manufacturing and product recovery see Gungor and Gupta (1999a). For a comprehensive review of disassembly sequencing see Lambert (2003) and Lambert and Gupta (2005). McGovern, Gupta, and Kamarthi (2003) first proposed combinatorial optimization techniques for the DLBP.

MODELING THE MULTI-CRITERIA PRODUCTION PROBLEM

The desired solution to a DLBP instance consists of an ordered sequence (i.e., *n*-tuple) of work elements (also referred to as tasks, components, or parts). For example, if a solution consisted of the eight-tuple ⟨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.

While different authors use a variety of definitions for the term "balanced" in reference to assembly (Elsayed & Boucher, 1994) and disassembly lines, we propose the following definition (McGovern et al., 2003; McGovern & Gupta, 2003) that considers the total number of workstations *NWS* and the station times (i.e., the total process-

Figure 1. Multicriteria selection procedure



ing time requirement in workstation j) ST_j; this definition will be used consistently throughout this chapter:

Definition: *A disassembly line is optimally balanced when the fewest possible number of workstations is needed and the variation in idle times between all workstations is minimized. This is mathematically described by*

Minimize NWS

then

Minimize $[\max(ST_x) - \min(ST_y)]\ \forall\, x, y \in \{1, 2, \ldots, NWS\}$

This is done while meeting any constraints, including precedence constraints. Line balancing can be visualized in Figure 1 with the boxes representing workstations (five here), the total height of the boxes indicating cycle time CT (the maximum time available at each workstation), the numbered boxes representing each part (1 through 11 here) and proportionate in height to each part removal time, and the gray area indicative of the idle time.

Minimizing the sum of the workstation idle times I, which will also minimize the total number of workstations, is described by

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

Line balancing seeks to achieve perfect balance (i.e., 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 (Elsayed & Boucher, 1994). 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 ST_j instead of to CT. (SI is very similar in format to the sample standard deviation from the field of statistics, but using $\max(ST_j) \mid j \in \{1, 2, \ldots, NWS\}$ rather than the mean of the station times.) LE rewards the minimum number of workstations but allows unlimited variance in idle times between workstations because no comparison is made between ST_js. The balancing method developed by McGovern et al. (2003; McGovern & Gupta, 2003) seeks to simultaneously minimize the number of workstations while ensuring that idle times at each workstation are similar, though at the expense of the generation of a nonlinear objective function. A resulting minimum numerical value is the more desirable solution, indicating both a minimum number of workstations and similar idle times across all workstations. The measure of balance F is represented as

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

Note that mathematically, Formula (2) effectively makes Formula (1) redundant due to the fact that it concurrently minimizes the number of workstations. This new method should be effective with traditional assembly line balancing problems as well.

Theorem*: Let PRT_k be the part removal time for the kth of n parts where CT is the maximum amount of time available to complete all tasks assigned to each workstation. Then for the most efficient distribution of tasks, the minimum (optimal) number of workstations, NWS^* satisfies*

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

Proof: *If the inequality is not satisfied, then there must be at least one workstation completing tasks requiring more than CT of time, which is a contradiction.* □

Subsequent bounds are shown to be true in a similar fashion and are presented throughout the chapter without proof.

The upper bound (worst case) for the number of workstations is given by

$$NWS_{nom} = n \tag{4}$$

Therefore

$$\left\lceil \frac{\sum_{k=1}^{n} PRT_k}{CT} \right\rceil \leq NWS \leq n \tag{5}$$

The lower bound on F is given by

$$F^* \geq \left(\frac{I}{NWS^*} \right)^2 \cdot NWS^* \tag{6}$$

while the upper bound is described by

$$F_{nom} = \sum_{k=1}^{n} (CT - PRT_k)^2 \tag{7}$$

therefore

$$\left(\frac{I}{NWS^*} \right)^2 \cdot NWS^* \leq F \leq \sum_{k=1}^{n} (CT - PRT_k)^2 \tag{8}$$

A hazard measure was developed to quantify each solution sequence's performance, with a lower calculated value being more desirable. This measure is based on binary variables that indicate whether a part is considered to contain hazardous material (the binary variable is equal to one if the part is hazardous, else zero) and its position in the sequence. A given solution sequence hazard measure is defined as the sum of hazard binary flags multiplied by their position in the solution sequence, thereby rewarding the removal of hazardous parts early in the part removal sequence. This measure H is represented as

$$H = \sum_{k=1}^{n} (k \cdot h_{PS_k})$$

$$h_{PS_k} = \begin{cases} 1, hazardous \\ 0, otherwise \end{cases} \tag{9}$$

where PS_k identifies the kth part in the solution sequence; that is, for solution ⟨3, 1, 2⟩, $PS_2 = 1$. The lower bound on the hazardous part measure is given by

$$H^* = \sum_{p=1}^{|HP|} p \tag{10}$$

where the set of hazardous parts is defined as

$$HP = \{k : h_k \neq 0 \ \forall \ k \in P\} \tag{11}$$

where P set of n part removal tasks, and its cardinality can be calculated with

$$|HP| = \sum_{k=1}^{n} h_k \tag{12}$$

For example, a product with three hazardous parts would give an H^* value of 1 + 2 + 3 = 6. The upper bound on the hazardous part measure is given by

$$H_{nom} = \sum_{p=n-|HP|+1}^{n} p \tag{13}$$

or alternatively

$$H_{nom} = (n \cdot |HP|) - |HP| \qquad (14)$$

For example, three hazardous parts in a product having a total of twenty would give an H_{nom} value of 18 + 19 + 20 = 57 or equivalently, H_{nom} = (20 · 3) – 3 = 60 – 3 = 57. Formulae (10), (13), and (14) are combined to give

$$\sum_{p=1}^{|HP|} p \le H \le \sum_{p=n-|HP|+1}^{n} p = (n \cdot |HP|) - |HP| \qquad (15)$$

Also, a demand measure was developed to quantify each solution sequence's performance, with a lower calculated value being more desirable. This measure is based on positive integer values that indicate the quantity required of this part after it is removed—or zero if it is not desired—and its position in the sequence. Any given solution sequence demand measure is defined as the sum of the demand value multiplied by their position in the sequence, rewarding the removal of high demand parts early in the part removal sequence. This measure D is represented as

$$D = \sum_{k=1}^{n} (k \cdot d_{PS_k})$$

$$d_{PS_k} \in \mathrm{N}, \forall PS_k \qquad (16)$$

The lower bound on the demand measure (D^*) is given by Formula (16) where

$$d_{PS_1} \ge d_{PS_2} \ge \ldots \ge d_{PS_n} \qquad (17)$$

For example, three parts with demands of 4, 5, and 6 respectively would give a best-case value of (1 · 6) + (2 · 5) + (3 · 4) = 28. The upper bound on the demand measure (D_{nom}) is given by Formula (16) where

$$d_{PS_1} \le d_{PS_2} \le \ldots \le d_{PS_n} \qquad (18)$$

For example, three parts with demands of 4, 5 and 6 respectively would give a worst-case value of (1 · 4) + (2 · 5) + (3 · 6) = 32.

Finally, a direction measure was developed to quantify each solution sequence's performance, with a lower calculated value indicating minimal direction changes and a more desirable solution. This measure is based on a count of the direction changes. Integer values represent each possible direction (typically r = {+ x, – x, + y, – y, + z, – z}; in this case $|r|$ = 6). These directions are expressed as

$$r_{PS_k} = \begin{cases} +1, \mathit{direction} & +x \\ -1, \mathit{direction} & -x \\ +2, \mathit{direction} & +y \\ -2, \mathit{direction} & -y \\ +3, \mathit{direction} & +z \\ -3, \mathit{direction} & -z \end{cases} \qquad (19)$$

and are easily expanded to other or different directions in a similar manner. The direction measure R is represented as

$$R = \sum_{k=1}^{n-1} R_k$$

$$R_k = \begin{cases} 1, r_{PS_k} \ne r_{PS_{k+1}} \\ 0, \mathit{otherwise} \end{cases} \qquad (20)$$

The lower bound on the direction measure is given by

$$R^* = |r| - 1 \qquad (21)$$

For example, for a given product containing six parts that are installed/removed in directions r_k = (– y, + x, – y, – y, + x, + x) the resulting best-case value would be 2 – 1 = 1 (e.g., one possible R^* solution containing the optimal, single-change of product direction would be: $\langle -y, -y, -y, +x, +x, +x\rangle$). In the specific case where the number of unique direction changes is one less than the total number of parts n, the upper bound on the direction measure would be given by

$$R_{nom} = |r| \quad \text{where } |r| = n - 1 \tag{22}$$

Otherwise, the measure varies depending on the number of parts having a given removal direction and the total number of removal directions. It is bounded by

$$|r| \le R_{nom} \le n - 1 \quad \text{where } |r| < n - 1 \tag{23}$$

For example, six parts installed/removed in directions $r_k = (+x, +x, +x, -y, +x, +x)$ would give an R_{nom} value of 2 as given by the lower bound of Formula (23) with a solution sequence of $\langle +x, +x, -y, +x, +x, +x\rangle$. Six parts installed/removed in directions $r_k = (-y, +x, -y, -y, +x, +x)$ would give an R_{nom} value of $6 - 1 = 5$ as given by the upper bound of Formula (23) with a solution sequence of $\langle -y, +x, -y, +x, -y, +x\rangle$ for example.

In the special case where each part has a unique removal direction, the measures for R^* and R_{nom} are equal and are given by

$$R^* = R_{nom} = n - 1$$

$$\text{where } |r| = n \tag{24}$$

Note that the optimal and nominal hazard, demand, and direction formulae are dependent upon favorable precedence constraints that will allow for generation of these optimal or nominal measures. Finally, note that McGovern and Gupta (2006a) have proven that the DLBP is NP-complete.

The combinatorial optimization techniques described here make use of these many criteria in a lexicographic form (detailed in the next section) to address the multicriteria aspects of DLBP. Since measure of balance is the primary consideration in this chapter, additional objectives are only considered subsequently; that is, the methodologies first seek to select the best performing measure of balance solution; equal balance solutions are then evaluated for hazardous part removal positions; equal balance and hazard measure solutions are evaluated for high-demand part removal positions; and equal balance, hazard measure and high-demand part removal position solutions are evaluated for the number of direction changes. This priority ranking approach was selected over a weighting scheme for its simplicity, ease in reranking the priorities, ease in expanding or reducing the number of priorities, due to the fact that other weighting methods can be readily addressed at a later time, and primarily to enable unencumbered *efficacy* (a method's effectiveness in finding good solutions) *analysis* of the combinatorial optimization methodologies and problem data instances under consideration.

LEXICOGRAPHIC GOAL PROGRAMMING FOR USE IN SOLUTION METHODOLOGY EVALUATION

One of the ways in which the complexity of DLBP manifests itself is with the multiple, often conflicting objectives (as defined in the previous section). The field of *multiple-criteria decision making* (MCDM) provides a variety of means for addressing the selection of a solution where several objectives exist. The bulk of MCDM methods involve multicriteria versions of linear programming (LP) problems. Since DLBP requires integers exclusively as its solution, it cannot be formulated as an LP. Additionally, since the objective described by Formula (2) is nonlinear, DLBP is not linear either (a requirement of LP, though this can be remedied using a version of the descriptions in the previous formal Definition). Also, many MCDM methods rely on *weighting*. These weights are in proportion to the importance of each objective. Weights were not desirable for this study since any results would be expected to be influenced by the weights selected. While this is appropriate for an application of the methodologies in this study to an applied problem and using experts to select the appropriate weights, here weighting would only serve to add an additional level of complexity to the comprehension of the problem and the proposed solutions. In addition, since the research in this study is not applied to a particular, unique disassembly situation but rather to the DLBP in general, the assignment of weighting values would be completely arbitrary and hence add little if any value to the final analysis of any results. Finally, the use of weights may not adequately reflect the generalized performance of the combinatorial optimization methods being studied; nuances in the methods, the data, and the weights themselves may generate atypical, unforeseen, or non-repeatable results. For these reasons, a simplified process was developed to select the best solution. Based on the priorities listed, the balance is the primary objective used to search for an optimal solution (note the use of "less than" and "less than or equal to" signs in Figure 2 indicating the desire for the better solution to be on the left side of the inequality since we are seeking to minimize all measures). Given multiple optimum extreme points in *F*, early removal of hazardous parts is then considered. Given multiple optimum extreme points in *F* and *H*, early removal of high-demand parts is considered next. Finally, given multiple optimum extreme points in *F*, *H*, and *D*, adjacent removal of parts with equivalent part removal directions is considered. This process is shown in pseudo-code format in Figure 2 where the most recent solution

Figure 2. Multicriteria selection procedure

```
Procedure BETTER_SOLUTION (new_solution, best_solution) {
IF      (new_solution.F < best_solution.F
        ∨
        (new_solution.F ≤ best_solution.F ∧
        new_solution.H < best_solution.H)
        ∨
        (new_solution.F ≤ best_solution.F ∧
        new_solution.H ≤ best_solution.H ∧
        new_solution.D < best_solution.D)
        ∨
        (new_solution.F ≤ best_solution.F ∧
        new_solution.H ≤ best_solution.H ∧
        new_solution.D ≤ best_solution.D∧
        new_solution.R < best_solution.R)){
                RETURN (TRUE)
        }
        RETURN (FALSE)
}
```

generated is given by *new_solution* while the best solution visited thus far in the search is given by *best_solution* with *.F*, *.H*, *.D*, and *.R* indicating the respective solution's numerical measures from formulae (2), (9), (16), and (20).

This process has its basis in two MCDM techniques:

The *feasible region* in an LP problem (and in DLBP) is usually a multidimensional subset of $\mathbf{R}^z$ containing an infinite (finite in DLBP due to its integer nature) number of points. Because it is formed by the intersection of a finite number of *closed half-spaces* (defined by ≤ and ≥, that is, the inequality constraints) and *hyperplanes* (equality constraints) it is *polyhedral*. Thus, the feasible region is *closed* and *convex* with a finite number of extreme points. The Simplex method (Hillier & Lieberman, 2005) for solving LPs exploits the polyhedral properties in the sense that the optimal solutions can be found without having to examine all of the points in the feasible region. Taking the steepest *gradient* following each point examined accomplishes this. Conventional LP algorithms and software terminate their search once the first optimal extreme point is found (in our example, once the first F^* is found). They fail to identify alternative optima if they exist. In general, an LP instance may have one or more optimal extreme points and one or more *unbounded edge*(*s*) (though the latter would not be expected of DLBP since it should be contained within the *convex hull* of a *polytope*, that is, a finite region of *n*-dimensional space enclosed by a finite number of hyperplanes). The optimal set is the *convex combination* (i.e., the set of all the points) of all optimal extreme points and points on unbounded edges. It is therefore desired to test all optimal extreme points. This can be done by *pivoting* and is performed using what is known as a *Phase III bookkeeping system*. Determining all alternative optima is enabled in a similar way using the routine shown in Figure 2 (as long as the combinatorial optimization technique in question is able to visit those extreme points).

A second MCDM technique that the process in Figure 2 borrows from is *preemptive (lexicographic) goal programming* (GP). GP was initially conceived by Charnes, Cooper, and Ferguson (1955) and Charnes and Cooper (1961) and conceptualizes objectives as *goals* then assigns priorities to the achievement of these goals. In preemptive GP, goals are grouped according to priorities. The goals at the highest priority level are considered to be infinitely more important than goals at the second priority level, and the goals at the second priority level are considered to be infinitely more important than goals at the third priority level, and so forth (note that a search can effectively terminate using GP if a high priority goal has a unique solution; as a result, lower order goals would not have the opportunity to influence the GP-generated solution). This process can be readily seen in Figure 2 where "infinitely more important" is enforced using the "less than or equal to" (≤) symbol.

This chapter makes use of the term "optimal" to describe the best solution. It should be noted that in the field of MCDM this term is changed to "efficient" (also, *noninferior*, *nondominated*, or *Pareto-optimum*) where there is no unique solution that maximizes all objectives simultaneously. With this understanding, "optimal" will continue to be used to refer to the best answer possible for a given instance and meeting the criteria set in Figure 2.

ASSESSMENT TOOLS

While the disassembly line is the best choice for automated disassembly of returned products, finding the optimal balance for a disassembly line is computationally intensive with exhaustive search quickly becoming prohibitively large. Combinatorial optimization techniques provide a general algorithmic framework that can be applied to this optimization problem. Although combinatorial optimization holds promise in solving DLBP, one of the concerns when using heuristics is the idea that very little has been rigorously established in reference to their performance; developing ways of explaining and predicting their performance is considered to be one of the most important challenges currently facing the fields of optimization and algorithms (Papadimitriou & Steiglitz, 1998). These challenges exist in the variety of evaluation criteria available, a lack of data sets for testing (disassembly-specific instances are addressed in Section 5.5), and a lack of performance analysis tools. In this section, mathematical and graphical tools for quantitative and qualitative performance analysis are developed and reviewed, focusing on analytical methodologies used in evaluating both of the combinatorial optimization searches used here.

Graphical Analysis Tools

Charts and tables provide an intuitive view into the workings and performance of solution-generating techniques. Both are used here to enhance the qualitative understanding of a methodology's execution and status of its terminal state as well as to allow for a comparison of relative performance with instance size and when compared to other methodologies.

The tables are used to observe the solution of any single instance. The tables used here present a solution in the following format: the sequence *n*-tuple is listed in the first row, followed by the corresponding part removal times, then the workstation assignments, then the hazard values, followed by the demand values, and finally the direction values (note that the direction representation $\{+x, -x, +y, -y, +z, -z\}$ is changed from $\{+1, -1, +2, -2, +3, -3\}$ as portrayed in the McGovern and Gupta (2006b) formulae to $\{0, 1, 2, 3, 4, 5\}$ for purposes of software engineering). To improve readability, the columns are shaded corresponding to the workstation assignment using alternating shades of gray. Use of this format (i.e., table) allows for study of the final solution state as well as potentially enabling improvements in algorithm performance due to insights gained by this type of observation.

The second graphical format used to allow for qualitative study of techniques and their solutions consists of a graphical comparison of known best- and worst-case results with the results/averaged results (deterministic techniques/stochastic techniques; since methodologies with a probabilistic component—such as would be found with evolutionary algorithms—can be expected to generate different answers over multiple runs) of a solution technique under consideration. The charts are used to observe multiple, varying-size solutions of the DLBP *A Priori* instances. Multiple charts are used to display the various performance measures, which are demonstrated here with the DLBP *A Priori* benchmark data sets of sizes $8 \leq n \leq 80$. The near-optimal solutions coupled with the known optimal and nominal solutions for all instance sizes under study provides a method, not only for comparing the methodologies to the best and worst cases, but to other methodologies as well. Computational complexity is portrayed using time complexity (analysis of the time required to solve a particular size instance of a given problem) while *space complexity* (analysis of the computer memory required to solve a particular size instance of a given problem; Rosen, 1999) is not considered. All time complexities are provided in *asymptotic notation* (*big-Oh*, *big-Omega*, and *big-Theta*) when commonly known or when calculated where able. "Time complexity" typically refers to worst-case runtimes, while in the numerical results portion of this chapter, the runtimes provide a qualitative description of the studied methodologies, so the experimentally determined time complexities are presented with the understanding that the information is the average-case time complexity of the particular software written for the problem used with a specific instance. In this chapter the charts used include: number of workstations with optimal, balance measure with optimal, normalized balance measure with optimal and nominal, hazard measure with optimal and nominal, demand measure with optimal and nominal, direction measure with optimal and nominal, average-case time complexity with third-order and exhaustive growth curves, and average-case time complexity curves in detail. Note that "number of workstations" and "idle time" measures are analogous (e.g., one can be calculated from the other) so only "number of workstations" is calculated and displayed here. Also, while "number of workstations" and "balance" are both calculated in various ways and displayed in separate graphs, they are strongly related as well. Both are presented to allow insight into the search processes and further quantify the efficacy of their solutions; however, it should be noted that, for example, a solution optimal in balance must also obviously be optimal in the number of workstations.

Note that with the graphs depicting third-order (i.e., $O(n^3)$) and exhaustive (i.e., $O(n!)$) growth curve graphics (as seen in Figure 14), the actual average-case time complexity curve under consideration is often not even readily visible. Even so, average-case time complexity with the third-order and exhaustive growth curves helps to show how relatively fast all of these techniques are, while the average-case time complexity graphics (Figure 15) defines the methodologies' speed and rate of growth in even more detail.

Though not demonstrated in this chapter, it is often of value to make use of overlaid linear or polynomial fitted regression lines to better provide graphical information for analysis of some of these very fast heuristics. When a heuristic's software is configured to time down to 1/100th of a second or slower, it should be recognized that many applications of heuristics are able to run on that order (or in some cases even faster); therefore, average-case time complexity curves may give the appearance of making dramatic steps up or down when this is actually more of an aberration of the order of the timing data that is collected. For that reason, showing the average-case time complexity with its regression line displays both the actual data and more importantly, the shape of the time growth in n.

In order to make the balance results comparable in magnitude to all other measures and to allow for more legible graphical comparisons with worst-case calculations in the charts, the effects of squaring portions of Formula (2) can be compensated for by taking the square root of the resulting F, F^*, or F_{nom}. This will subsequently be referred to in this study as *normalizing* (to reflect the concept of a reduction in the values to a common magnitude). While using Formula (2) is desirable to emphasize the importance of a solution's balance as well as to drive stochastic search processes towards the optimal solution, normalization allows for a more intuitive observation of the relative merits of any two solutions. For example, two solutions having an equal number of workstations (e.g., $NWS = 3$) but differing balance such as $I_j = \langle 1, 1, 4 \rangle$ and $I_j = \langle 2, 2, 2 \rangle$ (optimal balance) would have balance values of 18 and 12 respectively, while the normalized values would stand at 4.24 and 3.46, still indicating better balance with the latter solution, but also giving a sense of the relative improvement that solution provides, which the measure generated by Formula (2) lacks.

Efficacy Index Equations

The primary mathematical tool developed for quantitative analysis is shown in Formula (25). This will subsequently be referred to as the *efficacy index* (McGovern & Gupta, 2006b). The efficacy index EI_x (where x is some metric under consideration, e.g., F) is the ratio of the difference between a calculated measure and its worst-case measure to the measure's *sample range* (i.e., the difference between the best-case measure and the worst-case measure as given by: $\max(X_y) - \min(X_z) \mid y, z \in \{1, 2, \ldots, |X|\}$ from the area of statistical quality control) expressed as a percentage and described by

$$EI_x = \frac{100 \cdot (x_{nom} - x)}{x_{nom} - x^*} \qquad (25)$$

This generates a value between 0% and 100%, indicating the percentage of optimum for any given measure and any given combinatorial optimization methodology being evaluated. For example, the efficacy index formula for balance would read

$$EI_F = \frac{100 \cdot (F_{nom} - F)}{F_{nom} - F^*}$$

where the subscript *nom* represents the worst-case bound (nominal) for a given data set and the superscript * represents the best-case bound (optimal) for a given data set.

For the study of multiple data sets, probability theory presents us with the concept of a *sample mean*. The sample mean of a method's efficacy index can be calculated using

$$\overline{EI}_x = \left(\sum_{i=1}^{y} \frac{100 \cdot (x_{nom} - x_i)}{x_{nom} - x^*} \right) \Big/ y \qquad (26)$$

where y is the sample size (the number of data sets). While Formula (25) provides individual data set size efficacy indices—especially useful in demonstrating worst and best case as well as trends with instance size—Formula (26) allows a single numerical value that provides a quantitative measure of the location of the data center in a sample.

Statistical Regression

Performed but not demonstrated in here, an additional quantitative tool may be borrowed from the field of statistics. *Simple linear regression* and *correlation* (using the *sample coefficient of determination*), and *polynomial regression* and its associated *coefficient of multiple determination* can be used to quantify the accuracy of the curves and to provide the regression equation. The chart containing the combinatorial optimization methodologies with the third-order and exhaustive growth curves was used not only to provide a qualitative, graphical comparison, but also (along with the detailed time complexity curves) to determine the *degree* (*order*) of the fitted polynomial regression curve. Once the order was observed by comparison, either a linear or a polynomial regression model was selected and the regression equation was then automatically calculated by mathematical software (using, for example, an EXCEL 2000 spreadsheet software function), as was the coefficient of determination. The heuristic methodologies used here were either first (linear) or second order. As part of the quantitative portion of this research, this corresponds to average-case time complexities of O(n) or O(n^2).

Statistical Coefficient of Determination

The coefficient of determination was the final portion of the quantitative component of the study. This value represents the portion of the *variation* in the collected data explained by the fitted curve. The coefficient of determination is then multiplied by 100 to illustrate the adequacy of the fitted regression model, indicating the percentage of variation time that can be attributed to the size of the data set. The closer the value comes to 100%, the more likely it is an accurate model. While the coefficients of the polynomial regression model are of interest in presenting as accurate a growth curve model as possible, of greater value in this study is the order of the model since the largest exponent is the only variable of interest in complexity theory.

Performance Assessment Experimental Benchmark Data Set

Any solution methodology needs to be applied to a collection of test cases to demonstrate its performance as well as its limitations. Benchmark data sets are common for many NP-complete problems, such as *Oliver30* and *RY48P* for application to the Traveling Salesman Problem and *Nugent15/20/30*, *Elshafei19*, and *Krarup30* for the Quadratic Assignment Problem. Unfortunately, because of their size and their design, most of these existing data sets have no known optimal answer and new solutions are not compared to the optimal solution, but rather the best solution to date. In addition, since DLBP is a recently defined problem, no appropriate benchmark data sets exist.

This size-independent *a priori* benchmark data set was generated (McGovern & Gupta, 2004) based on the following. Since, in general, solutions to larger and larger instances cannot be verified as optimal (due to the time complexity of exhaustive search), it is proposed that instances be generated in such a way as to always provide a known solution. This was done by using part removal times consisting exclusively of prime numbers further selected to ensure that no permutations 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 (PRT_k, where k typically identifies a part or sequence position) 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 permutations of 3, 5, 7, and 11 at each workstation with idle times of zero. (Note that the cardinality of the set of part removal times $|PRT| \leq n$ since PRT_k is *onto* mapped to *PRT*, though not necessarily *one-to-one*, since multiple parts may have equal part removal times; that is, PRT_k is a *surjection* and may or may not be a *bijection* to *PRT*.)

As demonstrated in Table 1, 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 (the last one listed in the original data). In addition, one part (the last listed, second largest part removal time component) was listed as being demanded. This was done so that only the hazardous and the demand sequencing would be demonstrated while providing a slight solution sequence disadvantage to any purely greedy methodology (since two parts with part removal times of 3 and 5 are needed along with the parts with the larger part removal times to reach the optimal balance F^*, assigning hazard and high-demand attributes to those parts with smaller part removal times may prevent some methodologies from artificially obtaining an F^* sequence). From each part removal time size, the first listed part was selected to have a removal direction differing from the other parts with the same part removal time. This was done to demonstrate direction selection while requiring any solution-generating methodology to move these first parts of each part removal time size encountered to the end of the sequence (i.e., into the last workstation) in order to obtain the optimal part direction value of $R^* = 1$ (assuming the solution technique being evaluated is able to successfully place the hazardous and demanded parts towards the front of the sequence).

Also, there were no precedence constraints placed on the sequence, a deletion that further challenges any method's ability to attain an optimal solution (by maximizing the feasible search space). This has the added benefit of more precisely modeling the restricted version of the decision version (i.e., non-optimization) of DLBP seen in McGovern and Gupta (2006a).

Table 1. DLBP A Priori data for n = 12

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
Hazardous	0	0	0	0	0	0	0	0	0	0	0	1
Demand	0	0	0	0	0	0	0	0	1	0	0	0
Direction	1	0	0	1	0	0	1	0	0	1	0	0

Hazard values are given by

$$h_k = \begin{cases} 1, & k = n \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

with demand values given by

$$d_k = \begin{cases} 1, & k = \dfrac{n \cdot (|PRT| - 1)}{|PRT|} \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

and part removal direction values given by

$$r_k = \begin{cases} 1, & k = 1, \dfrac{n}{|PRT|} + 1, \dfrac{2n}{|PRT|} + 1, \ldots, \dfrac{(|PRT| - 1) \cdot n}{|PRT|} + 1 \\ 0, & \text{otherwise} \end{cases} \tag{29}$$

Since $|PRT|$ = 4 in this chapter, each part removal time is generated by

$$PRT_k = \begin{cases} 3, 0 < k \le \dfrac{n}{4} \\ 5, \dfrac{n}{4} < k \le \dfrac{n}{2} \\ 7, \dfrac{n}{2} < k \le \dfrac{3n}{4} \\ 11, \dfrac{3n}{4} < k \le n \end{cases} \tag{30}$$

Known optimal results include balance measure $F^* = 0$, hazardous part measure $H^* = 1$, demanded part measure $D^* = 2$, and part removal direction measure $R^* = 1$.

A data set containing parts with equal part removal times and no precedence constraints will result in multiple optimum extreme points. Using probability theory (counting sample points using the generalized multiplication rule covering *n* operations), it can be seen in Table 2 (and detailed in McGovern & Gupta, 2004) that the number of solutions optimal in all objectives goes from less than 8.3% of *n* at *n* = 4, to 0.12% at *n* = 8, dropping to effectively 0% at *n* = 16; as *n* grows, the percentage of optimal solutions gets closer and closer to zero.

The final configuration of the benchmark as used here was 19 instances with instance size distributed from *n* = 8 to *n* = 80 in steps of $|PRT|$ = 4. The size and range of the instances is considered appropriate, with small *n*s tested—which decreases the *NWS* value and tends to exaggerate less than optimal performance—as well as large, which demonstrates time complexity growth and efficacy changes with *n*.

H-K HEURISTIC

Heuristic Search Background

Exhaustive search techniques (e.g., pure depth-first or pure breadth-first) will fail to find a solution to any but the smallest instances within any practical length of time. *Blind search*, *weak search*, *naïve*

Table 2. Comparison of possible solutions to optimal solutions for a given n using the DLBP A Priori data

n	*n*!	Number optimal in balance	Number optimal in all	Percentage optimal in balance	Percentage optimal in all
4	24	24	2	100.00%	8.33%
8	40,320	9,216	48	22.86%	0.12%
12	479,001,600	17,915,904	10,368	3.74%	0.00%
16	2.09228E+13	1.10075E+11	7,077,888	0.53%	0.00%

search, and *uninformed search* are all terms used to refer to algorithms that use the simplest, most intuitive method of searching through a search space, whereas *informed search* algorithms use heuristics to apply knowledge about the structure of the search space. An uninformed search algorithm is one that does not take into account the specific nature of the problem. This allows uninformed searches to be implemented in general, with the same implementation able to be used in a wide range of problems. Uninformed searches include exhaustive search and H-K. H-K seeks to take advantage of the benefits of uninformed search while addressing the exhaustive search drawbacks of runtime growth with instance size.

Heuristic Motivation and Introduction

Exhaustive search is optimal because it looks at every possible answer. While an optimal solution can be found, this technique is impractical for all but the simplest combinatorial problems due to the explosive growth in search time. In many physical search applications (e.g., antisubmarine warfare, search and rescue) exhaustive search is not possible due to time or sensor limitations. In these cases, it becomes practical to sample the search space and operate under the assumption that, for example, the highest point of land found during the conduct of a limited search is either is the highest point in a given search area or is reasonably near the highest point. The proposed search technique (McGovern & Gupta, 2004) in this chapter works by sampling the exhaustive solution set; that is, search the solution space in a method similar to an exhaustive search but in a pattern that skips solutions (conceptually similar to the STEP functionality in a FOR loop as found in computer programming) to significantly minimize the search space (Figure 3; the shading indicates solutions visited, the border represents the search space).

This pattern is analogous to the radar acquisition search pattern known as "spiral scan," the search and rescue pattern of the "expanding square," or the antisubmarine warfare aircraft "magnetic anomaly detector (MAD) hunting circle." Once the solution is generated, the space can be further searched with additional applications of the H-K heuristic (with modifications from the previous H-K) or the best-to-date solution can be further refined by performing subsequent local searches (such as 2-opt or smaller, localized H-K searches). Depending on the application, H-K can be run once, multiple times on subsequent solutions, multiple times from the same starting point using different skip measure (potentially as a multiprocessor application using parallel algorithms or as a grid computing application), multiple times from a different starting point using the same skip measure (again, potentially as a

Figure 3. Exhaustive search space and the H-K search space and methodology

multiprocessor or grid computing application), or followed up with an H-K or another, differing local search on the best or several of the best suboptimal solutions generated. While termination normally takes place after all sequences are generated for a given skip size, termination can also be effected based on time elapsed or once finding a solution that is within a predetermined bound. H-K can also be used as the first phase of a hybrid algorithm or to hot start another methodology (e.g., to provide the initial population in a GA). One interesting use for H-K is application to the unusual problem where quantifying a small improvement (i.e., a greedy decision, such as would be found in ant colony optimization where the ant agents build a solution incrementally and, therefore, need to know which of the available solution elements reflects an improvement) is not possible or is not understood, or where the incremental greedy improvements may not lead to a global optima. Finally, H-K would also be useful in quickly gathering a sampling of the solution space to allow for a statistical or other study of the data (e.g., H-K could enable the determination of the approximate worst-case and best-case solutions as well as solution efficacy indices *mean*, *median*, and *mode*).

The *skip size* ψ, or more generally ψ_k (the kth element's skip measure; i.e., for the solution's third element, visit every 2nd possible task for $\psi_3 = 2$) can be as small as $\psi = 1$ or as large as $\psi = n$. Since $\psi = 1$ is equivalent to exhaustive search and $\psi = n$ generates a trivial solution (it returns only one solution, that being the data in the same sequence as it is given to H-K, that is, $PS_k = \langle 1, 2, 3, \ldots, n \rangle$; also, in the single-phase H-K this solution is already considered by any value of ψ), in general all skip values can be further constrained as

$$2 \le \psi_k \le n - 1 \qquad (31)$$

Depending on structural decisions, H-K can take on a variety of forms, from a classical optimization algorithm in its most basic form, to a general evolutionary algorithm with the use of multiple H-K processes, to a *biological* or *natural process algorithm* by electing random functionality. In order to demonstrate the method and show some of its limitations, in this chapter the most basic form of the H-K heuristic is used: one process (though visiting the data twice, in forward and in reverse order), constant starting point of $PS_k = \langle 1, 1, 1, \ldots, 1 \rangle$ (since the solution set is a permutation, there are no repeated items; therefore, the starting point is effectively $PS_k = \langle 1, 2, 3, \ldots, n \rangle$), constant skip type (i.e., each element in the solution sequence is skipped in the same way), constant maximum skip size (although different skip sizes are used throughout each H-K run, and no follow-on solution refinement.

The H-K Process and DLBP Application

As far as the H-K process itself, since it is a modified exhaustive search allowing for solution sampling, it searches for solutions similar to depth-first search iteratively seeking the next permutation iteration—allowing for skips in the sequence—in lexicographic order. In the basic H-K and with $\psi = 2$, the first element in the first solution would be 1, the next element considered would be 1, but since it is already in the solution, that element would be incremented and 2 would be considered and be acceptable. This is repeated for all of the elements until the first solution is generated. In the next iteration, the initial part under consideration would be incremented by 2 and, therefore, 3 would be considered and inserted as the first element. Since 1 is not yet in the sequence, it would be placed in the second position, 2 in the third, and so forth. For DLBP H-K this is further modified to test the proposed sequence part addition for precedence constraints. If all possible parts for a given solution position fail these checks, the remainder of the positions are not further inspected, the procedure falls back to the previously successful solution addition, increments it by 1, and continues. These processes are repeated until all allowed items have been visited in the first solution position (and by default, due to the nested nature of the search, all subsequent solution positions). For example, with $n = 4$, $P = \{1, 2, 3, 4\}$, 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 $PS_k = \langle 3, 4, 1, 2\rangle$. With $n = 5$, $P = \{1, 2, 3, 4, 5\}$, and no precedence constraints, instead of considering the 5! = 120 possible permutations, only 16 are considered by the single-phase H-K with $\psi = 2$ and using forward-only data as demonstrated in Figure 4.

All of the parts are maintained in a tabu-type list. Each iteration of the DLBP H-K generated solution is considered for feasibility. If it is ultimately feasible in its entirety, DLBP H-K then looks at each element in the solution and places that element using the Next-Fit (NF) rule (from the Bin-Packing problem application; once a bin has no space for a given item attempted to be packed into it, that bin is never used again even though a later, smaller item may appear in the list and could fit in the bin (see Hu & Shing, 2002). DLBP H-K puts the element under consideration into the current workstation if it fits. If it does not fit, a new workstation is assigned and previous workstations are never again considered. Although NF does not perform as well as First-Fit, Best-Fit, First-Fit-Decreasing, or Best-Fit-Decreasing when used in the general Bin-Packing problem, it is the only one of these rules that will work with

Figure 4. DLBP H-K results at $n = 5$ and $\psi = 2$

$$PS_k = \langle 1, 2, 3, 4, 5\rangle$$
$$PS_k = \langle 1, 2, 5, 3, 4\rangle$$
$$PS_k = \langle 1, 4, 2, 3, 5\rangle$$
$$PS_k = \langle 1, 4, 5, 2, 3\rangle$$
$$PS_k = \langle 1, 4, 5, 3, 2\rangle$$
$$PS_k = \langle 3, 1, 2, 4, 5\rangle$$
$$PS_k = \langle 3, 1, 4, 2, 5\rangle$$
$$PS_k = \langle 3, 1, 4, 5, 2\rangle$$
$$PS_k = \langle 3, 4, 1, 2, 5\rangle$$
$$PS_k = \langle 3, 4, 1, 5, 2\rangle$$
$$PS_k = \langle 3, 4, 5, 1, 2\rangle$$
$$PS_k = \langle 5, 1, 2, 3, 4\rangle$$
$$PS_k = \langle 5, 1, 4, 2, 3\rangle$$
$$PS_k = \langle 5, 3, 1, 2, 4\rangle$$
$$PS_k = \langle 5, 3, 1, 4, 2\rangle$$
$$PS_k = \langle 5, 3, 4, 1, 2\rangle$$

a DLBP solution sequence due to the existence of precedence constraints (see McGovern & Gupta, 2005 for a DLBP implementation of First-Fit-Decreasing). 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. Although the actual software implementation for this study consisted of a

Figure 5. The DLBP H-K procedure

```
Procedure DLBP_H-K {
    SET $ISS_k$ := 0 $\forall$ $k \in P$
    SET $FS$ := 1
    ┌
    $PS_1$ := 1 to $n$, skip by $\psi_1$
        SET $ISS_{PS1}$ := 1
        ┌
        $PS_2$ := 1 to $n$, skip by $\psi_2$
            WHILE     ($ISS_{PS2}$ = = 1 $\vee$
                       PRECEDENCE_FAIL $\wedge$
                       not at $n$)
                Increment $PS_2$ by 1

            IF    $ISS_{PS2}$ = = 1
            THEN SET $FS$ := 0
            ELSE  SET $ISS_{PS2}$ := 1
                :
            ┌   :
            IF    $FS$ = = 1
                    $PS_n$ := 1 to $n$ skip by $\psi_n$
                        WHILE     ($ISS_{PSn}$ = =1 $\vee$
                                   PRECEDENCE_FAIL $\wedge$
                                   not at $n$)
                            Increment $PS_n$ by 1

                        IF    $ISS_{PSn}$ = = 0
            └           THEN evaluate solution $PS$
                :
                :
        IF    $FS$ = = 1
        THEN SET $ISS_{PS2}$ := 0
        ELSE  SET $FS$ := 1
        └
    SET $ISS_{PS1}$ := 0
    SET $FS$ := 1
    └
}
```

very compact recursive algorithm, in the interest of clarity, the general DLBP H-K procedure is presented here as a series of nested loops (Figure 5, where ISS_k is the binary flag representing the tabu-type list; set to 1 if part k is in the solution sequence, and FS is the feasible sequence binary flag; set to 1 if the sequence is feasible).

Skip size affects various measures including the efficacy indices and time complexity. The general form of the skip-size to problem-size relationship is formulated as

$$\psi_k = n - \Delta\psi_k \tag{32}$$

where $\Delta\psi$ represents the kth element's delta skip measure; difference between problem size n and skip size ψ_k (i.e., for $\Delta\psi = 10$ and $n = 80$, $\psi = 70$).

Early tests of time complexity growth with skip size suggest another technique to be used as part of H-K search. Since any values of ψ that are larger than the chosen skip value for a given H-K instance were seen to take significantly less processing time, considering all larger skip values should also be considered in order to increase the search space at the expense of a minimal increase in search time. In other words, H-K can be run repeatedly on a given instance using all skip values from a smallest ψ (selected based upon time complexity considerations) to the largest (i.e., $n - 1$ per Formula (31)) without a significant time penalty. In this case, any ψ_k would be constrained as

$$n - \Delta\psi_k \le \psi_k \le n - 1$$

$$\text{where } 1 \le \Delta\psi_k \le n - 2 \tag{33}$$

If this technique is used (as it is here), it should also be noted that multiples of ψ visit the same solutions; for example, for $n = 12$ and $2 \le \psi \le 10$, the four solutions considered by $\psi = 10$ are also visited by $\psi = 2$ and $\psi = 5$.

In terms of time complexity, the rate of growth has been observed to be exponential in the inverse of ψ. The average-case time complexity of H-K is then listed as $O(b^b)$ in skip size, where $b = 1/\psi$. Due to the nature of H-K, the number of commands executed in the software do not vary based on precedence constraints, data sequence, greedy or probabilistic decision making, improved solutions nearby, and so forth, so the worst case is also $O(b^b)$, as is the best case (big-Omega of $\Omega(b^b)$), and, therefore, a tight bound exists, which is $\Theta(b^b)$. As used in the Numerical Comparison section below (forward and reverse data, $1 \le \Delta\psi \le 10$, and resulting skip sizes of $n - 10 \le \psi \le n - 1$), the average-case time complexity of DLBP H-K curve is listed as $O(n^2)$ or polynomial complexity. The deterministic, single iteration nature of H-K also indicates that the process would be no faster than this, so it is expected that the time complexity lower bound is $\Omega(n^2)$ and, therefore, the H-K appears to have an asymptotically tight bound of $\Theta(n^2)$ as configured here.

GENETIC ALGORITHM

GA Model Description

A genetic algorithm (a parallel neighborhood, stochastic-directed search technique) provides an environment where solutions continuously crossbreed, mutate, and compete with each other until they evolve into an optimal or near-optimal solution (Holland, 1975). Due to its structure and search method, a GA is often able to find a global solution, unlike many other heuristics that use hill climbing to find a best solution nearby resulting only in a local optima. In addition, a GA does not need specific details about a problem nor is the problem's structure relevant; a function can be linear, nonlinear, stochastic, combinatorial, noisy and so forth.

GA has a solution structure defined as a *chromosome*, which is made up of *genes* and generated by two *parent* chromosomes from the *pool* of solutions, each having its own measure of *fitness*. New solutions are generated from old using the techniques of *crossover* (sever parents genes and swap severed sections) R_x and *mutation* (randomly vary genes within a chromosome) R_m. Typically, the main challenge with any genetic algorithm implementation is determining a chromosome representation that remains valid after each generation.

For DLBP the chromosome (solution) consisted of a sequence of genes (parts). A pool, or *population*, of size N was used. Only feasible disassembly sequences were allowed as members of the population or as offspring. The fitness was computed for each chromosome using the method for solution performance determination (i.e., in lexicographic order using F, H, D, then R).

The time complexity is a function of the number of generations, the population size N, and the chromosome size n. As such, the runtime is seen to be on the order of $n \cdot N \cdot$(number of generations). Since both the population and the number of generations are considered to stay constant with instance size, the best-case time complexity of GA is seen to have an asymptotic lower bound of $\Omega(n)$. Because the worst-case runtime also requires no more processing time than $T(n) \propto n \cdot N \cdot$(number of generations), the worst-case time complexity of GA has the asymptotic upper bound $O(n)$, so GA therefore exhibits a tight time complexity bound of $\Theta(n)$.

DLBP-Specific Genetic Algorithm Architecture

The GA for DLBP was constructed as follows (McGovern & Gupta, 2007). An initial, feasible population was randomly generated and the fitness of each chromosome in this generation was calculated. An even integer of $R_x \cdot N$ parents was randomly selected for crossover to produce $R_x \cdot N$ offspring (offspring make up $(R_x \cdot N \cdot 100)$% of each generation's population). (Note that often GA's use fitness values rather than random selection for crossover; the authors found that random selection made creation of children that were duplicates of each other or of parents less likely and allowed for a more diverse population.) An elegant crossover, the precedence preservative crossover (PPX) developed by Bierwirth, Mattfeld, and Kopfer (1996) was used to create the offspring. As shown in Figure 6, PPX first creates a mask (one for each child, every generation). The mask consists of random 1s and 2s indicating which parent part information should be taken from. If, for example, the mask for child 1 reads 22121112, 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 next three available parts in parent 1 would make up genes five, six, and seven of child 1; the last part in parent 2 would make up gene eight of child 1. This technique is repeated using a new mask for child 2.

After crossover, mutation is randomly conducted. Mutation was occasionally (based on the R_m value) performed by randomly selecting a single child then exchanging two of its disassembly tasks while ensuring precedence is preserved. The $R_x \cdot N$ 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 formed using PPX, the solution set could contain 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 the study in this chapter. To avoid this, DLBP GA was modified to treat duplicate solutions as

Figure 6. PPX example

Parent 1:	1 3 2 6 5 8 7 4		Parent 1:	1 3 2 6 5 8 7 4	
Parent 2:	1 2 3 5 6 8 7 4	⇒	Parent 2:	**1 2** 3 5 6 8 7 4	⇒
Mask:	2 2 1 2 1 1 1 2		Mask:	2 2 1 2 1 1 1 2	
Child:			Child:	1 2	
Parent 1:	x **3** x 6 5 8 7 4		Parent 1:	x x x 6 5 8 7	
Parent 2:	x x 3 5 6 8 7 4	⇒	Parent 2:	x x x **5** 6 8 7 4	⇒
Mask:	1 2 1 1 1 2		Mask:	2 1 1 1 2	
Child:	1 2 3		Child:	1 2 3 5	
Parent 1:	x x x **6** x **8** **7** 4		Parent 1:	x x x x x x x 4	
Parent 2:	x x x x 6 8 7 4	⇒	Parent 2:	x x x x x x x **4**	
Mask:	1 1 1 2		Mask:	2	
Child:	1 2 3 5 6 8 7		Child:	1 2 3 5 6 8 7 4	

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 $(1 - R_x) \cdot N$ parents were kept along with all of the $R_x \cdot N$ offspring to make up the next generation then the process was repeated.

DLBP-Specific GA Qualitative Modifications

DLBP GA was modified from a general GA in several ways (Figure 7). Instead of the worst portion of the population being selected for crossover as is often the case in GA, in DLBP GA all of the population was (randomly) considered for crossover. This better enables the selection of nearby solutions (i.e., solutions similar to the best solutions to-date) common in many scheduling problems. Also, mutation was performed only on the children in DLBP GA, not the worst parents as is typical in a general GA. This was done to address the small population used in DLBP GA and to counter PPX's tendency to duplicate parents. Finally, duplicate children are sorted in DLBP GA to make their deletion from the population likely since there is a tendency for the creation of duplicate solutions (due to PPX) and due to the small population saved from generation to generation.

DLBP-Specific GA Quantitative Modifications

While the matter of population sizing is a controversial research area in evolutionary computing, here a small population was used (20) to minimize data storage requirements and simplify analysis while a large number of generations were used (10,000) to compensate for this small population while not being so large as to take an excessive amount of processing time. This was also done to avoid solving all cases to optimality since it was desirable to determine the point at which the DLBP GA performance begins to break

Figure 7. The DLBP GA procedure

```
Procedure DLBP_GA {
          INITIALIZE_DATA                 {Load data: part removal times, etc.}

          SET count := 1                  {count is the generation counter}

          FOR N DO:                       {Randomly create an initial population}
                    DO:
                              RANDOMLY_CREATE_CHROMOSOME
                    WHILE (PRECEDENCE_FAIL)
                    CALC_FITNESS          {Establish the initial solution's fitness}

          SET num_parents := N * $R_x$        {Determine number of parents for reproduction}
                                          {Ensure an even number of parents}
          SET num_parents := 2 * (num_parents / 2)
                                          {Note: num_parents is typed as an integer}
          DO:                             {Run GA for MAX_GENERATIONS}
                    RANDOMIZE_PARENTS     {Randomly order the parents for breeding}

                    FOR num_parents DO:   {Perform crossover using PPX}
                              CROSSOVER

                    IF FLIP($R_m$) DO:        {FLIP equals 1 $R_m$ % of the time, else 0}
                              MUTATE_CHILD {Randomly select and mutate a child}

                    REMOVE_LEAST_FIT      {Sort: best parents to the last positions}
                    REMOVE_REPEATS        {Duplicate answers to the front and resort}
                    CHILDREN_TO_POOL      {Add children to the population}

                    FOR N DO:
                              CALC_FITNESS {Calculate each solution's fitness}

                    count := count + 1    {Next generation}
          WHILE (count < MAX_GENERATIONS)

          SAVE the last chromosome as the best_solution
}
```

down and how that breakdown manifests itself. A 60% crossover was selected based on test and analysis. Developmental testing indicated that a 60% crossover provided better solutions and did so with one-third less processing time than, for example 90%. Previous assembly line balancing literature that indicated best results have typically been found with crossover rates of from 0.5 to 0.7 also substantiated the selection of this lower crossover rate. A mutation was performed about one percent of the time. Although some texts recommend 0.01% mutation while applications in journal papers have used as much as 100% mutation, it was found that 1.0% gave excellent algorithm performance for the Disassembly Line Balancing Problem.

NUMERICAL COMPARISON AND QUALITATIVE ANALYSIS

The evolutionary algorithm and the uninformed heuristic were run on the DLBP *A Priori* experimental instances (with size varying between $8 \leq n \leq 80$) with the results comparatively analyzed in this section. For H-K, all cases were calculated using a varying delta skip of $1 \leq \Delta\psi \leq 10$ with the resulting skip sizes of $n - 10 \leq \psi \leq n - 1$; the software was set up so that it would not attempt any skip size smaller than $\psi = 3$ (to avoid exhaustive or near-exhaustive searches with small instances). In addition, the DLBP *A Priori* data sets were run with the data in forward and reverse for H-K. For GA, all tests were performed using a population size of 20, mutation rates of 1.0%, 10,000 generations, and crossover rates of 60%. Note that, due to the averaging of the results given by the process with stochastic characteristics (DLBP GA), many of the reported results are not purely discrete.

Table-Based Qualitative Assessment

From Section 5.1, the table-based qualitative analysis tool is demonstrated. The DLBP H-K technique can be seen below (Table 3) on the DLBP *A Priori* data presented forward and reverse at $n = 12$ and $3 \leq \psi \leq 11$ (note that without the minimum allowed value of $\psi = 3$, skip values would include $2 \leq \psi \leq 11$). DLBP H-K was able to find a solution optimal in the number of workstations, balance, and hazard (though not demand or direction). The solution found came from the reverse set and consisted of $NWS^* = 3$, $F^* = 0$, $H^* = 1$, $D = 10$ (optimal value is $D^* = 2$), and $R = 2$ (optimal value is $R^* = 1$).

An example of a GA metaheuristic solution can be seen below with $n = 12$ (Table 4). While there is more than one optimal solution for the *A Priori* instances, GA was able to regularly find one of the multiple optimum extreme points (three workstations and $F^* = 0$, $H^* = 1$, $D^* = 2$, and $R^* = 1$).

Graph-Based Qualitative Assessment and Quantitative Analysis Using Efficacy Indices and Regression

Next, a qualitative comparison was performed using the graph-based format. Quantitative analysis

Table 3. H-K solution using the A Priori instance at n = 12 and 3 ≤ ψ ≤ 11 (data presented in forward and reverse order)

Part ID	12	2	5	8	11	1	4	7	10	9	6	3
PRT	11	3	5	7	11	3	5	7	11	7	5	3
Workstation	1	1	1	1	2	2	2	2	3	3	3	3
Hazardous	1	0	0	0	0	0	0	0	0	0	0	0
Demand	0	0	0	0	0	0	0	0	0	1	0	0
Direction	0	0	0	0	0	1	1	1	1	0	0	0

Table 4. Typical GA solution using the A Priori instance at n = 12

Part ID	12	9	5	3	8	11	6	2	4	10	7	1
PRT	11	7	5	3	7	11	5	3	5	11	7	3
Workstation	1	1	1	1	2	2	2	2	3	3	3	3
Hazardous	1	0	0	0	0	0	0	0	0	0	0	0
Demand	0	1	0	0	0	0	0	0	0	0	0	0
Direction	0	0	0	0	0	0	0	0	1	1	1	1

was enabled first by calculating the efficacy indices and then through the experimental determination of average-case time complexity using linear or polynomial regression. Note that the curves also appear to rather sharp increases and decreases. This is due to a combination of discrete values being connected by straight lines (i.e., with no smoothing) and due to the scale of those parameters' charts, rather than being indicative of any data anomalies.

The first study performed was a measure of each of the technique's calculated number of workstations as compared to the optimal. As shown in Figure 8, both of the methods performed very well in workstation calculation, staying within 2 workstations of optimum for all data set sizes tested. On the full range of data ($8 \leq n \leq 80$), DLBP H-K found solutions with NWS^* workstations up to $n = 12$, then solutions with $NWS^* + 1$ workstations through data set 11 ($n = 48$), after which it stabilized at $NWS^* + 2$. From Formula (25) H-K's efficacy index in number of workstations started at an optimal $EI_{NWS} = 100\%$, dropped to a low of $EI_{NWS} = 92\%$, then continuously climbed through to $EI_{NWS} = 97\%$ with an efficacy index sample mean in number of workstations of $\overline{E}_{NWS} = 96\%$. DLBP GA's efficacy index in number of workstations went from a high of 100%, down to 94% then stabilized between 97% and 98%. Overall, as given by Formula (26), DLBP GA shows an efficacy index sample mean in number of workstations of 97%.

In terms of the calculated measure of balance, again, both of the methods are seen to perform very well and significantly better than the worst case (as given by formula (7)). Best case is found uniformly at $F = 0$, as illustrated by Figure 9. However, examining the balance in greater detail provides some insight into the different techniques.

DLBP H-K and GA tend to decrease similarly and in a step function fashion (note, however, that even their normalized balance performance actually improves overall with instance size as a percentage of worst case as indicated by their improved efficacy indices with instance size; see Figure 10). This is to be expected; for H-K this has to do with the fact that as the instance size grows, a lower and lower percentage of the search space is investigated (assuming the skip size range is not allowed to increase with instance size, which is the case in this chapter). For GA, efficacy falls off with instance size because (unlike, for example, many hill-climbing and r-optimal processes that continue to search until no better nearby solution can be found) GA is set up to terminate after a fixed number of generations (in addition, GA's population does not increase with increases in instance size).

While increases in H-K's balance measure are seen with increases in data set size, as a percentage

Figure 8. Workstation calculations for each DLBP combinatorial optimization method

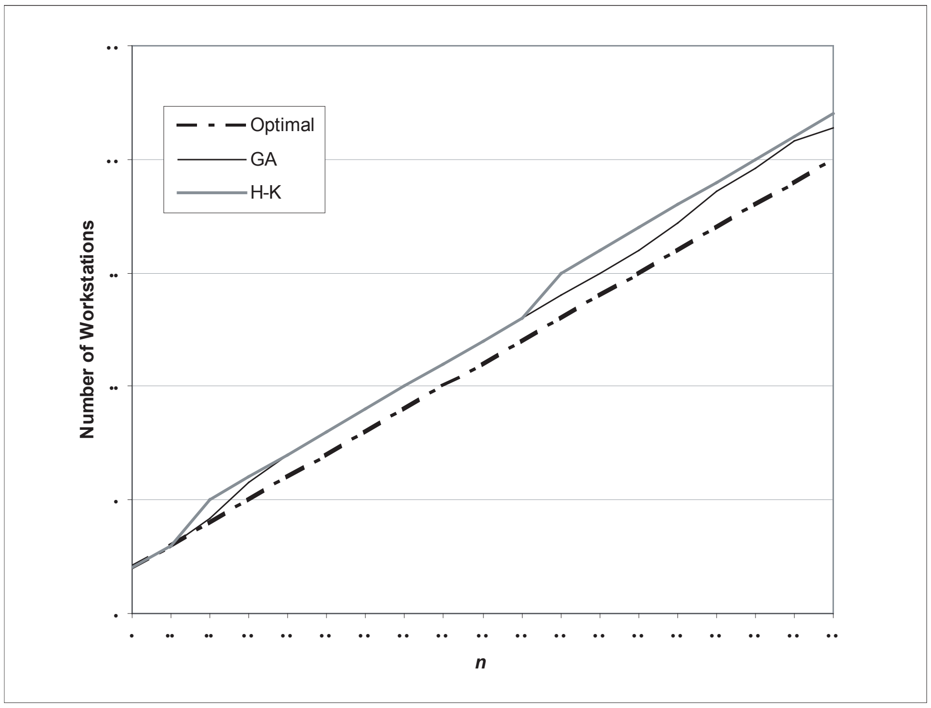


Figure 9. Detailed DLBP combinatorial optimization methods' balance performance

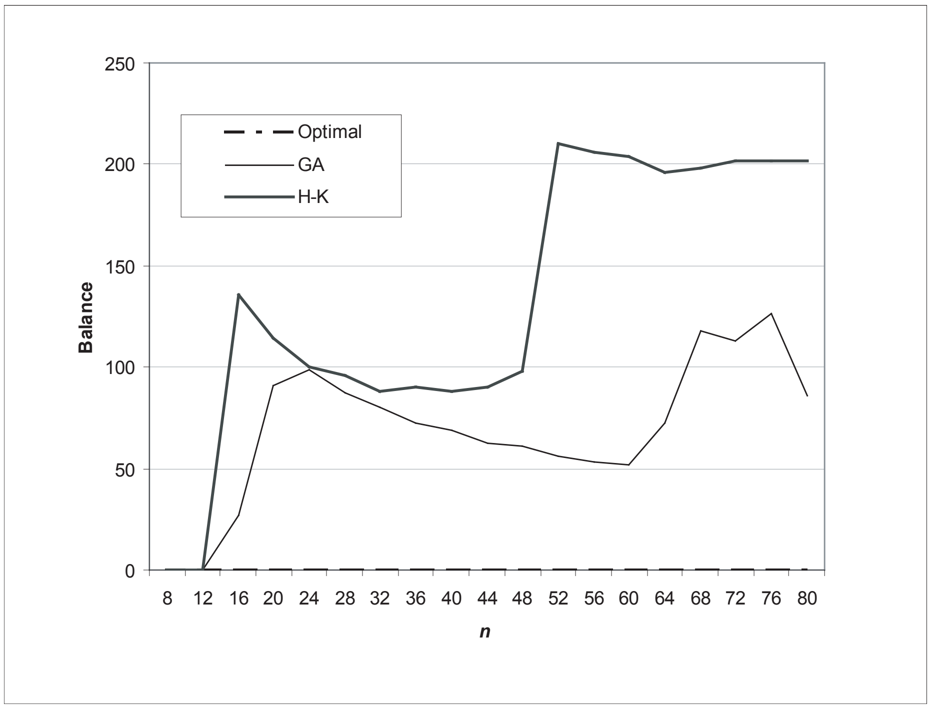
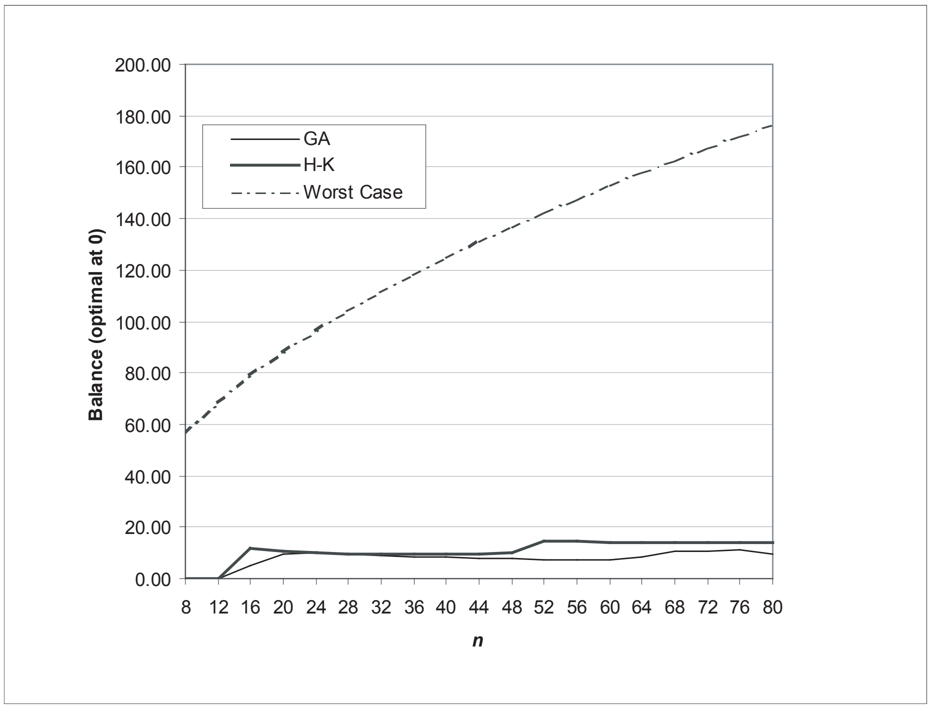


Figure 10. Normalized DLBP combinatorial optimization methods' balance performance



of the overall range from best case to worst case, the normalized balance measure tends to decrease (i.e., improve) with increases in the data set size. The normalized balance efficacy index dropped from a high of $EI_F = 100\%$ to a low of $EI_F = 85\%$ at data set 3 ($n = 16$) then slowly climbed to $EI_F = 92\%$ giving a sample mean of $\overline{E}_F = 92\%$. With DLBP GA, the normalized balance efficacy index started as high as 100% then dropped to 93% to 95%; it was never lower than 89% and had a sample mean of 94%. An instance size of $n = 16$ was seen to be the point at which the optimally balanced solution was not consistently found for the selected N and number of generations. Although DLBP GA's performance decreased with instance size, it can be seen in Figure 10 that the solution found, while not optimal, was very near optimal and when normalized, roughly paralleled the optimal balance curve.

The hazard measure results are as expected since hazard performance is designed to be deferential to balance and affected only when a better hazard measure can be attained without adversely affecting balance. The hazard measure tends to get worse with problem size using DLBP GA, with its efficacy index dropping relatively constantly from 100% to 60% and having a sample mean of $\overline{E}_H = 84\%$. The hazardous part was regularly suboptimally placed by DLBP H-K as well. Hazardous-part placement stayed relatively consistent with problem size (though effectively improving as compared to the worst case, as illustrated by Figure 11). The hazard measure's efficacy index fluctuates, similarly to a sawtooth wave function, between $EI_H = 57\%$ and $EI_H = 100\%$, giving a sample mean of $\overline{E}_H = 90\%$.

With DLBP GA, the demand measure gets worse at a slightly more rapid rate than the hazardous-part measure—as is expected due to the multicriteria priorities—with its efficacy index dropping from 100% to 45% (with a low of 42%) and having a sample mean of $\overline{E}_D = 78\%$ (Figure 12). DLBP H-K also suboptimally placed the high-demand part, and also at a higher rate than the hazardous part. Its efficacy index fluctuates between $EI_D = 7\%$ and $EI_D = 103\%$ (due to better than optimal placement in position $k = 1$ at the expense of hazard placement). Its resulting demand measure sample mean is $\overline{E}_D = 49\%$.

With part removal direction structured as to be deferential to balance, hazard, and demand, the two methodologies were seen to decrease in performance in a haphazard fashion. This decrease in performance is seen both when compared to the best case and when compared to the worst case (Figure 13). Again, these results are as expected due the prioritization of the multiple objectives. Though the part removal direction efficacy gets as high as $EI_R = 86\%$ with the H-K implementation, by data set 5 ($n = 24$) it has dropped to $EI_R = 0\%$ and never rises higher again than $EI_R = 43\%$, resulting in a sample mean of $\overline{E}_R = 20\%$. Using the GA, the part removal direction measure gets worse at a more rapid rate than the demand measure, again attributed to the multicriteria priorities, with its efficacy index dropping as low as 17% (at data set 15 where $n = 64$) and having a sample mean of $\overline{E}_R = 49\%$.

Finally, time complexity was examined using the *A Priori* data. Both of the techniques were seen to be very fast (Figure 14) and each is seen to be faster than third order.

Using DLBP GA, runtime increased very slowly with instance size. A linear model was used to fit the curve with the linear regression equation calculated to be $T(n) = 0.0327n + 1.5448$ with a

Figure 11. DLBP combinatorial optimization methods' hazard performance

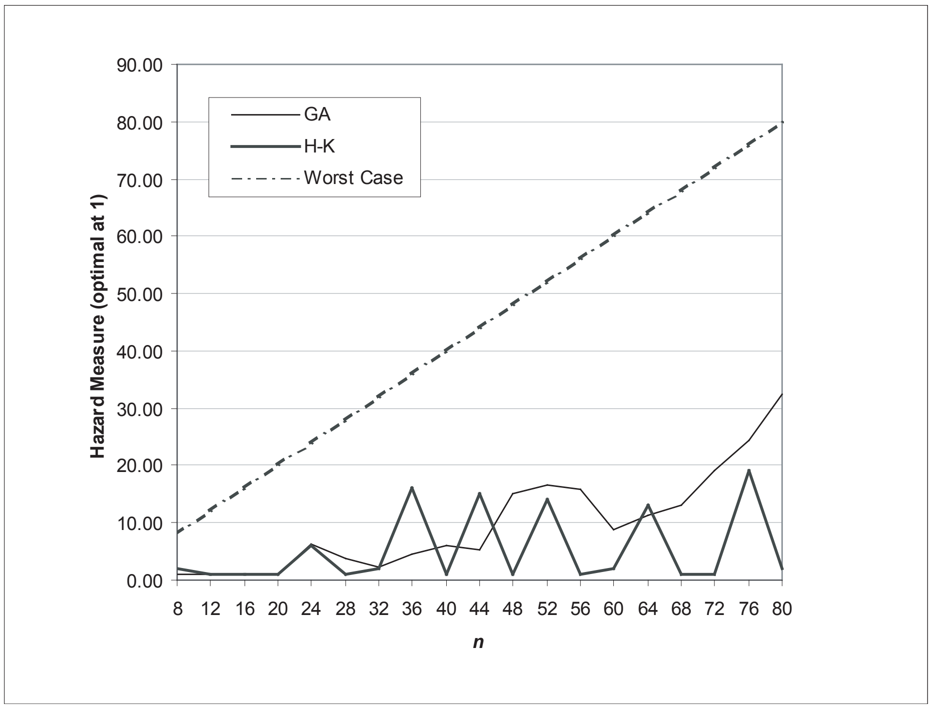


Figure 12. DLBP combinatorial optimization methods' demand performance

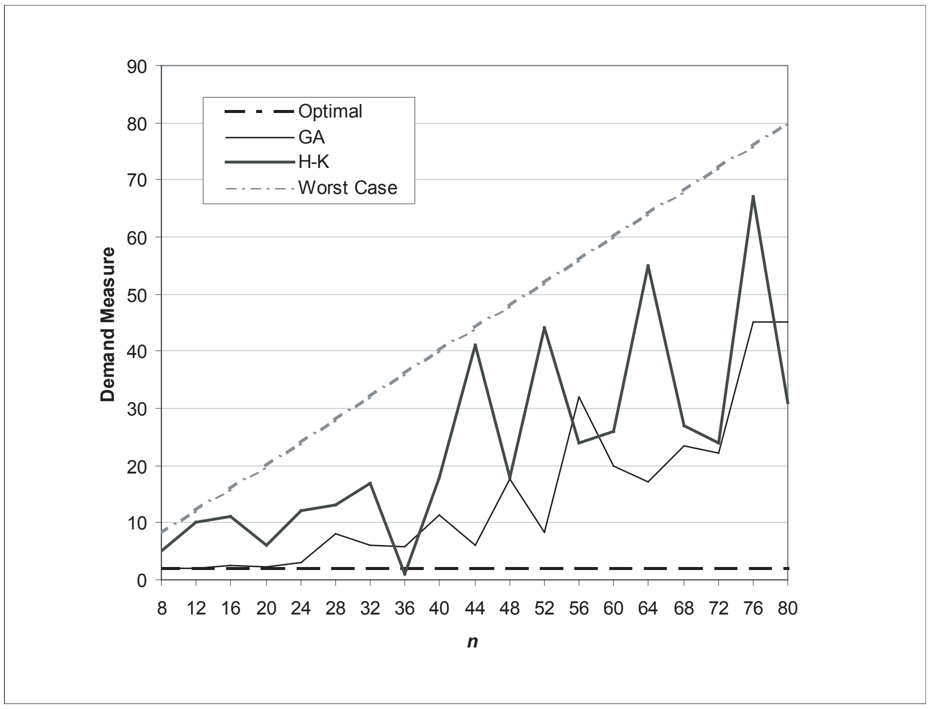
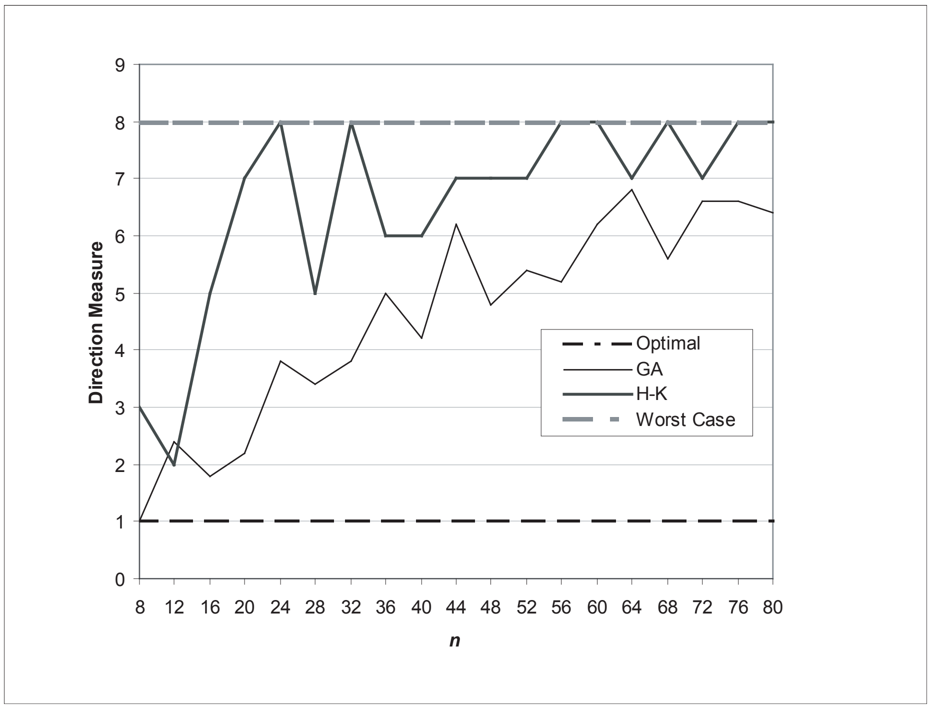


Figure 13. DLBP combinatorial optimization methods' part removal direction performance

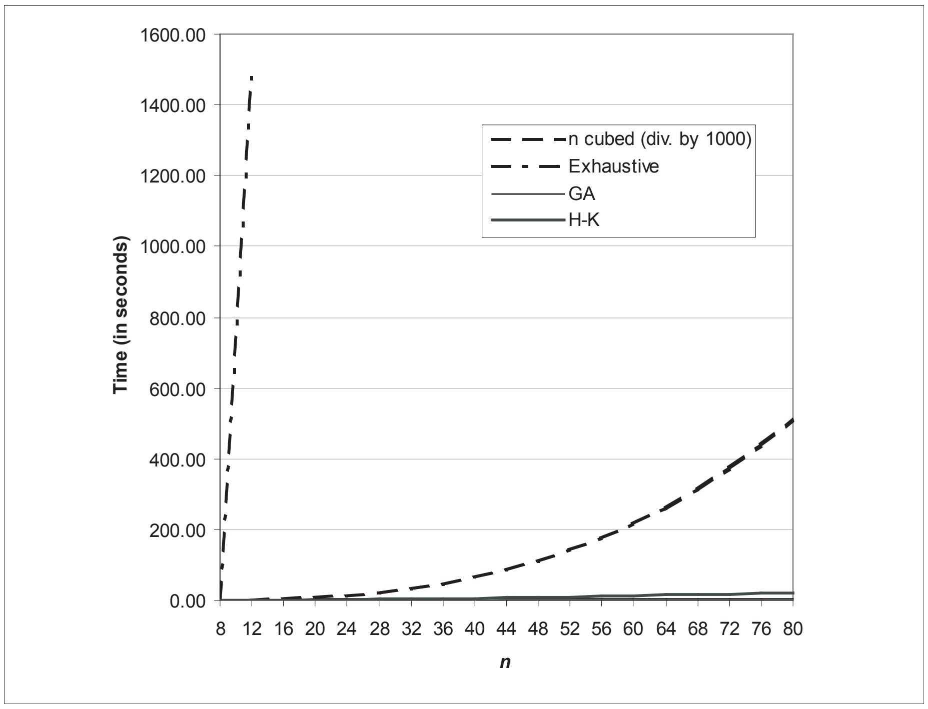


coefficient of determination of 0.9917 indicating 99.17% of the total variation is explained by the calculated linear regression curve. The growth of $0.0327n + 1.5448$ provides an experimentally derived result for the average-case time complexity of DLBP GA on the DLBP *A Priori* data sets as $O(n)$ or *linear complexity* (Rosen, 1999). This is in agreement with the theoretical calculations.

With DLBP H-K, the regression equation was calculated to be $T(n) = 0.0033n^2 - 0.0002n + 0.2893$. The small coefficients are indicative of the slow runtime growth in instance size. The coefficient of determination is calculated to be 0.9974, indicating 99.74% of the total variation is explained by the calculated regression curve. With a 2nd-order polynomial regression model used to fit the H-K curve, the average-case time complexity of DLBP H-K curve (with forward and reverse data, $1 \le \Delta\psi \le 10$, and resulting skip sizes of $n - 10 \le \psi \le n - 1$) is listed as $O(n^2)$ or polynomial complexity. The deterministic, single iteration nature of H-K also indicates that the process would be no faster than this, so it is expected that the time complexity lower bound is $\Omega(n^2)$ and, therefore, the H-K appears to have an asymptotically tight bound of $\Theta(n^2)$ as configured here. This empirical result is also in agreement with the theoretical complexity determination.

The runtimes can be examined in greater detail in Figure 15. While DLBP GA was seen to be very fast due to its linear growth, at some point it may be necessary to increase GA's population or number of generations to allow for the generation of adequate solutions and this will of course increase its runtime. DLBP H-K was also very fast, even with $\Delta\psi$ varying from 1 to 10 and with forward and reverse data runs (the anomaly seen in the H-K curve in Figure 15 is due to a software rule that dictated that all ψ could be as small as $n - 10$, but no less than $\psi = 3$, to prevent exhaustive

Figure 14. Time complexity of DLBP combinatorial optimization methods

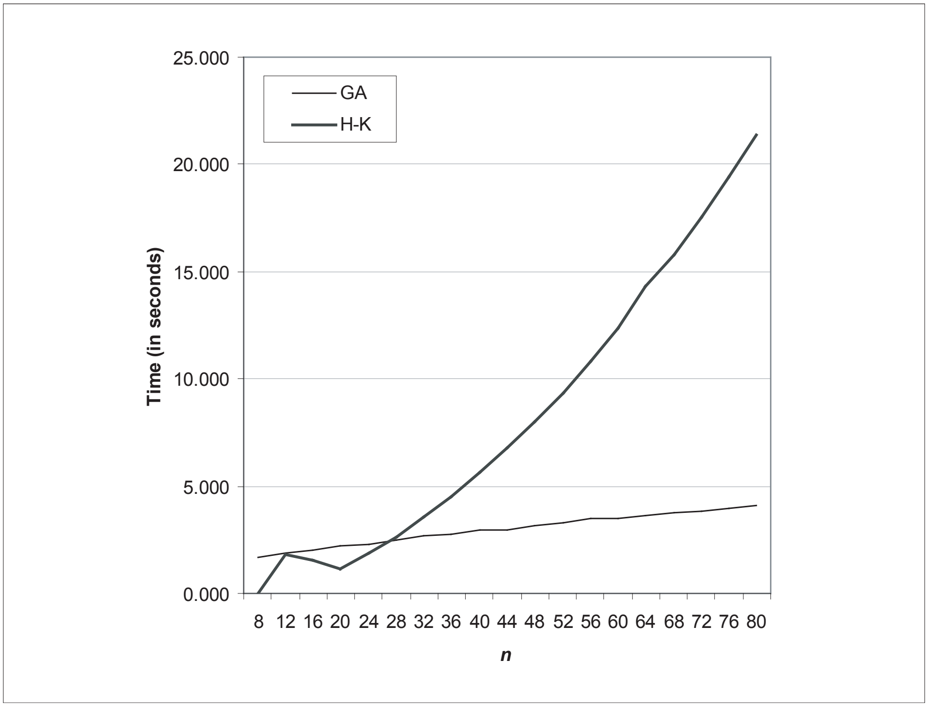


or near-exhaustive searches at small *n*). The H-K process grows approximately exponentially in 1/ψ, taking, for example from 0.02 seconds at ψ = 5 (actually 5 ≤ ψ ≤ 11) up to just under 20 seconds at ψ = 2 (i.e., 2 ≤ ψ ≤ 11 and 19.34 seconds) with *n* = 12 and forward and reverse data (exhaustive search was seen to take almost 25 minutes (24.61 minutes) on the same size data).

Although less than optimal, these results are not unusual for heuristics run against this data set. These suboptimal results are not indicative of poor heuristic performance but are, more likely, indicative of a successful DLBP *A Priori* benchmark data set design. The DLBP *A Priori* benchmark data is especially designed to challenge the solution-finding ability of a variety of combinatoric solution-generating techniques to enable a thorough quantitative evaluation of various method's performances in different areas (McGovern & Gupta, 2004). Suboptimal solutions typically result when this specially designed set of instances is processed, even at seemingly small *n* values. For each of these methodologies, their DLBP-specific engineering element of searching exclusively for precedence-preserving solution *n*-tuples means that the inclusion or addition of precedence constraints will reduce the search space and increasingly move any of these methods towards the optimal solution.

A smaller ψ or (as with GA and other search techniques) the inclusion of precedence constraints will increasingly move the DLBP H-K method towards the optimal solution. The time complexity performance of DLBP H-K provides the tradeoff benefit with the technique's near-optimal performance, demonstrating the moderate increase in time required with problem size, which grows markedly slower than the exponential growth of

Figure 15. Detailed time complexity of DLBP combinatorial optimization methods



exhaustive search. Runtime performance can be improved by increasing skip size and by running the data in one direction only while the opposite (i.e., decreasing skip size and running different versions of the data, including running the data in forward and reverse order) would be expected to increase solution efficacy. Note that, as a newly developed methodology, other modifications may decrease runtime and/or increase solution efficacy as well. With DLBP GA runtime performance can be improved by reducing the number of generations or reducing the population, while the opposite is true for improving other measures of performance; that is, increase the number of generations or increase the population to improve efficacy indices.

Each of the collected quantitative values was then compiled into one table for reference and comparison (Table 5). This table contains each of the combinatorial optimization methodologies researched here and lists their number of workstations, balance, hazard, demand and part removal direction sample mean efficacy indices; regression model average-case experimentally determined time complexity; and associated asymptotic upper bound (experimentally determined average case using the DLBP *A Priori* data).

The shading provides a quick reference to performance, with darker shades indicating worsening performance. While Exhaustive Search is included in the table, none of its row elements are considered for shading since its purpose in the table is as the benchmark for comparison. Note that the balance measure sample mean efficacy index is based on the normalized balance. This is done in the interest of providing a more appropriate and consistent scale across the table.

FUTURE DIRECTIONS

Directions for future research can be described as follows:

- It may be of interest to vary the multicriteria ordering of the objectives; two possibilities include a re-ordering of the objectives based on expert domain knowledge, and a comparison of all permutations of the objectives in search of patterns or unexpected performance improvements or decreases
- While the multiple-criteria decision making approach used here made use of preemptive goal programming, many other methodologies are available and should be considered to decrease processing time, for an overall or selective efficacy improvement, or to examine other promising methods including weighting schemes
- It may be of interest to make use of the promise of H-K in generating uninformed solutions from throughout the search space through the use of H-K to hot start GA
- Throughout the research, complete disassembly was assumed; while this is of great theoretical interest (since it allows for a worst-case study in terms of problem complexity and it provides consistency across

Table 5. Summary of quantitative measures for both methodologies

	$\overline{EI}_{NWS}$	$\overline{EI}_{F(norm)}$	$\overline{EI}_{H}$	$\overline{EI}_{D}$	$\overline{EI}_{R}$	$T(n)$	big-Oh
Exhaustive	100%	100%	100%	100%	100%	$1.199n!$	$O(n!)$
GA	97%	94%	84%	78%	49%	$0.033n$	$O(n)$
H-K	96%	92%	90%	49%	20%	$0.003n^2$	$O(n^2)$

the problem instances and methodologies), in practical applications it is not necessarily desired, required, practical, or efficient—recommend that future studies consider the more applied problem that allows for incomplete or partial disassembly

- Per the prior recommendation and an earlier one, different multicriteria ordering of the objectives could simplify the determination of the optimal level of incomplete disassembly; for example, if the main objective is to remove several demanded parts of a product containing hundreds of parts, by making the demand measure a higher priority objective than the balance, it may be found that these parts can be removed relatively early on in the disassembly process thereby allowing the process to terminate significantly earlier in the case of partial disassembly

This concludes some suggestions for future research. In addition to the items listed above, any further developments and applications of recent methodologies to multicriteria decision making, the H-K algorithm or GA or the Disassembly Line Balancing Problem that will help to extend the research in these areas may be appropriate.

CONCLUSION

In this chapter, a new multi-objective optimization problem was reviewed and used to demonstrate the development of several performance assessment techniques and the use of lexicographic goal programming for comparison and analysis of combinatoric search methodologies. From the field of combinatorial optimization, an evolutionary algorithm and an uninformed general-purpose search heuristic were selected to demonstrate these tools. Quantitative and qualitative comparison was performed using tables, graphs and efficacy measures with varying instance sizes and using exhaustive search as a time and performance benchmark. Each of the methodologies performed very well—though consistently suboptimally—with the different performance assessment techniques demonstrating a variety of performance subtleties that show no one technique to be ideal in all applications. While these results are solely dependent on the *A Priori* data set instances used, this data set appears to have successfully posed—even with multiple optimum extreme points—a nontrivial challenge to the methodologies and in all instance sizes, providing at least an initial indication that the data set does meet its design expectations and does not contain any unforeseen nuances that would render it a weak benchmark for this type of NP-complete combinatorial problem.

REFERENCES

Bierwirth, C., Mattfeld, D. C., & Kopfer, H. (1996). On permutation representations for scheduling problems. In H. M. Voigt, W. Ebeling, I. Rechenberg, & H.-P. Schwefel (Eds.), *Parallel problem solving from nature -- PPSN IV, Lecture notes in computer science* (pp. 310-318). Berlin, Germany: Springer-Verlag.

Brennan, L., Gupta, S. M., & Taleb, K. N. (1994). Operations planning issues in an assembly/disassembly environment. *International Journal of Operations and Production management*, *14*(9), 57-67.

Charnes, A. & Cooper, W. W. (1961). *Management models and industrial applications of linear programming.* New York: John Wiley and Sons.

Charnes, A., Cooper, W. W., & Ferguson, R. O. (1955). Optimal estimation of executive compensation by linear programming. *Management Science*, *1*(2), 138-151.

Elsayed, E. A. & Boucher, T. O. (1994). *Analysis and control of production systems*. Upper Saddle River, New Jersey: Prentice Hall.

Erel, E. & Gokcen, H. (1964). Shortest-route formulation of mixed-model assembly line balancing problem. *Management Science*, *11*(2), 308-315.

Garey, M. & Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP completeness*. San Francisco, CA: W. H. Freeman and Company.

Güngör, A. & Gupta, S. M. (1999a). A systematic solution approach to the disassembly line balancing problem. In *Proceedings of the 25th International Conference on Computers and Industrial Engineering* (pp. 70-73), New Orleans, Louisiana.

Güngör, A. & Gupta, S. M. (1999b). Disassembly line balancing. In *Proceedings of the 1999 Annual Meeting of the Northeast Decision Sciences Institute* (pp. 193-195), Newport, Rhode Island.

Güngör, A. & Gupta, S. M. (1999c). Issues in environmentally conscious manufacturing and product recovery: A survey. *Computers and Industrial Engineering*, *36*(4), 811-853.

Güngör, A. & 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.

Güngör, A. & Gupta, S. M. (2002). Disassembly line in product recovery. *International Journal of Production Research, 40*(11), 2569-2589.

Gupta, S. M. & Taleb, K. (1994). Scheduling disassembly. *International Journal of Production Research*, *32*(8), 1857-1866.

Gutjahr, A. L. & Nemhauser, G. L. (1964). An algorithm for the line balancing problem. *Management Science*, *11*(2), 308-315.

Hackman, S. T., Magazine, M. J., & Wee, T. S. (1989). Fast, effective algorithms for simple assembly line balancing problems. *Operations Research*, *37*(6), 916-924.

Hillier, F. S. & Lieberman, G. J. (2005). *Introduction to operations research*. New York: McGraw-Hill.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Hu, T. C. & Shing, M. T. (2002). *Combinatorial algorithms*. Mineola, NY: Dover Publications.

Lambert, A. D. J. (2003). Disassembly sequencing: A survey. *International Journal of Production Research*, *41*(16), 3721-3759.

Lambert, A. J. D. & Gupta, S. M. (2005). *Disassembly modeling for assembly, maintenance, reuse, and recycling*. Boca Raton, FL: CRC Press (Taylor & Francis).

Lapierre, S. D., Ruiz, A., & Soriano, P. (2006). Balancing assembly lines with tabu search. *European Journal of Operational Research*, *168*(3), 826-837.

McGovern, S. M. & Gupta, S. M. (2003). Greedy algorithm for disassembly line scheduling. In *Proceedings of the 2003 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 1737-1744), Washington, D.C.

McGovern, S. M. & Gupta, S. M. (2004). Combinatorial optimization methods for disassembly line balancing. In *Proceedings of the 2004 SPIE International Conference on Environmentally Conscious Manufacturing IV* (pp. 53-66), Philadelphia, Pennsylvania.

McGovern, S. M. & Gupta, S. M. (2005). Local search heuristics and greedy algorithm for balancing the disassembly line. *The International Journal of Operations and Quantitative Management*, *11*(2), 91-114.

McGovern, S. M. & Gupta, S. M. (2006a). Computational complexity of a reverse manufacturing line. In *Proceedings of the 2006 SPIE International Conference on Environmentally*

Conscious Manufacturing VI (CD-ROM), Boston, Massachusetts.

McGovern, S. M. & Gupta, S. M. (2006b). Performance metrics for end-of-life product processing. In *Proceedings of the 17th Annual Production & Operations Management Conference* (CD-ROM) Boston, Massachusetts.

McGovern, S. M., & Gupta, S. M. (2007). A balancing method and genetic algorithm for disassembly line balancing. *European Journal of Operational Research*, *179*(3), 692-708.

McGovern, S. M., Gupta, S. M., & Kamarthi, S. V. (2003). Solving disassembly sequence planning problems using combinatorial optimization. In *Proceedings of the 2003 Northeast Decision Sciences Institute Conference* (pp. 178-180), Providence, Rhode Island.

Osman, I. H. (2004). Metaheuristics: Models, design and analysis. In *Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference* (pp. 1.2.1-1.2.16), Gold Coast, Australia.

Osman, I. H. & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, *63*, 513-623.

Papadimitriou, C. H. & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. Mineola, NY: Dover Publications.

Ponnambalam, S. G., Aravindan, P., & Naidu, G. M. (1999). A comparative evaluation of assembly line balancing heuristics. *The International Journal of Advanced Manufacturing Technology*, *15*, 577-586.

Rosen, K. H. (1999). *Discrete mathematics and its applications*. Boston, MA: McGraw-Hill.

Suresh, G., Vinod, V. V., & Sahu, S. (1996). A genetic algorithm for assembly line balancing. *Production Planning and Control*, *7*(1), 38-46.

Torres, F., Gil, P., Puente, S. T., Pomares, J., & Aracil, R. (2004). Automatic PC disassembly for component recovery. *International Journal of Advanced Manufacturing Technology*, *23*(1-2), 39-46.

Tovey, C. A. (2002). Tutorial on computational complexity. *Interfaces*, *32*(3), 30-61.

ADDITIONAL READING

Agrawal, S. & Tiwari, M. K. (2008). A collaborative ant colony algorithm to stochastic mixed-model u-shaped disassembly line balancing and sequencing problem. *International Journal of Production Research, 46*(6), 1405-1429.

Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1974). *The design and analysis of computer programs*. Reading, MA: Addison-Wesley.

Altekín, F. T. (2005). *Profit oriented disassembly line balancing*. Unpublished doctoral dissertation, Middle East Technical University, Ankara, Turkey.

Altekin, F. T., Kandiller, L., & Ozdemirel, N. E. (2008). Profit-oriented disassembly-line balancing. *International Journal of Production Research, 46*(10), 2675-2693.

Bautista, J. & Pereira, J. (2002). Ant algorithms for assembly line balancing. In M. Dorigo (Ed.), *ANTS 2002, LNCS 2463* (pp. 65-75). Berlin, Germany: Springer-Verlag.

Das, S. K. & Naik, S. (2002). Process planning for product disassembly. *International Journal of Production Research*, *40*(6), 1335-1355.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, *26*(1), 1-13.

Duta, L., Filip, F. G., & Henrioud, J. M. (2002). Automated disassembly: Main stage in manufactured products recycling. In *Proceedings of the 4th International Workshop on Computer Science and Information Technologies* (CD-ROM), Patras, Greece.

Duta, L., Filip, F. G., & Henrioud, J. M. (2005). Applying equal piles approach to disassembly line balancing problem. In *Proceedings of the 16th IFAC World Congress* (CD-ROM), Prague, Czech Republic.

Franke, C., Basdere, B., Ciupek, M., & Seliger, S. (2006). Remanufacturing of mobile phones - Capacity, program and facility adaptation planning. *Omega, 34*(6), 562-570.

Glover, F. (1989). Tabu search, Part I. *ORSA Journal of Computing*, *1*(3), 190-206.

Glover, F. (1990). Tabu search, Part II. *ORSA Journal of Computing*, *2*(1), 4-32.

Güngör, A. & Gupta, S. M. (1997). An evaluation methodology for disassembly processes. *Computers and Industrial Engineering*, *33*(1), 329-332.

Güngör, A. & Gupta, S. M. (1998). Disassembly sequence planning for products with defective parts in product recovery. *Computers and Industrial Engineering*, *35*(1-2), 161-164.

Güngör, A. & Gupta, S. M. (2001). Disassembly sequence plan generation using a branch-and-bound algorithm. *International Journal of Production Research*, *39*(3), 481-509.

Gupta, S. M., Evren, E., & McGovern, S. M. (2004). Disassembly sequencing problem: A case study of a cell phone. In *Proceedings of the 2004 SPIE International Conference on Environmentally Conscious Manufacturing IV* (pp. 43-52), Philadelphia, Pennsylvania.

Gupta, S. M. & Güngör, A. (2001). Product recovery using a disassembly line: Challenges and solution. In *Proceedings of the 2001 IEEE International Symposium on Electronics and the Environment* (pp. 36-40), Denver, Colorado.

Gupta, S. M. & McGovern, S. M. (2004). Multiobjective optimization in disassembly sequencing problems. In *Proceedings of the 2nd World Conference on Production & Operations Management and the 15th Annual Production & Operations Management Conference* (CD-ROM), Cancun, Mexico.

Hong, D. S. & Cho, H. S. (1997). Generation of robotic assembly sequences with consideration of line balancing using simulated annealing. *Robotica*, *15*, 663-673.

Hopper, E. & Turton, B. C. H. (2000). An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, *128*(1), 34-57.

Huang, Y. M. & Liao, Y.-C. (2006). Optimum disassembly process with genetic algorithms for a compressor. In *Proceedings of the ASME 2006 Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (CD-ROM), Philadelphia, Pennsylvania.

Iori, M. (2003). *Metaheuristic algorithms for combinatorial optimization problems*. Unpublished doctoral dissertation, University of Bologna, Bologna, Italy.

Kekre, S., Rao, U. S., Swaminathan, J. M., & Zhang, J. (2003). Reconfiguring a remanufacturing line at Visteon, Mexico. *Interfaces*, *33*(6), 30-43.

Kongar, E. & Gupta, S. M. (2002a). A genetic algorithm for disassembly process planning. In *Proceedings of the 2002 SPIE International Conference on Environmentally Conscious Manufacturing II* (pp. 54-62), Newton, Massachusetts.

Kongar, E. & Gupta, S. M. (2002b). A multicriteria decision making approach for disassembly-to-order systems. *Journal of Electronics Manufacturing*, *11*(2), 171-183.

Kongar, E., Gupta, S. M., & McGovern, S. M. (2003). Use of data envelopment analysis for product recovery. In *Proceedings of the 2003 SPIE International Conference on Environmentally Conscious Manufacturing III* (pp. 219-231), Providence, Rhode Island.

Kotera, Y. & Sato, S. (1997). An integrated recycling process for electric home appliances. *Mitsubishi Electric ADVANCE*, *September*, 23-26.

Lambert, A. J. D. (1999). Linear programming in disassembly/clustering sequence generation. *Computers and Industrial Engineering*, *36*(4), 723-738.

Lambert, A. J. D. (2002). Determining optimum disassembly sequences in electronic equipment. *Computers and Industrial Engineering*, *43*(3), 553-575.

Lambert, A. J. D. & Gupta, S. M. (2002). Demand-driven disassembly optimization for electronic products. *Journal of Electronics Manufacturing*, *11*(2), 121-135.

Lapierre, S. D. & Ruiz, A. B. (2004). Balancing assembly lines: An industrial case study. *Journal of the Operational Research Society*, *55*(6), 589–597.

Lee, D.-H., Kang, J.-G., & Xirouchakis, P. (2001). Disassembly planning and scheduling: Review and further research. *Journal of Engineering Manufacture*, *215*(B5), 695-709.

McGovern, S. M., & Gupta, S. M. (2003). 2-opt heuristic for the disassembly line balancing problem. In *Proceedings of the SPIE International Conference on Environmentally Conscious Manufacturing III* (pp. 71-84), Providence, Rhode Island.

McGovern, S. M. & Gupta, S. M. (2004a). Demanufacturing strategy based upon metaheuristics. In *Proceedings of the 2004 Industrial Engineering Research Conference* (CD-ROM), Houston, Texas.

McGovern, S. M. & Gupta, S. M. (2004b). Metaheuristic technique for the disassembly line balancing problem. In *Proceedings of the 2004 Northeast Decision Sciences Institute Conference* (pp. 223-225), Atlantic City, New Jersey.

McGovern, S. M., & Gupta, S. M. (2004c). Multi-criteria ant system and genetic algorithm for end-of-life decision making. In *Proceedings of the 35th Annual Meeting of the Decision Sciences Institute* (pp. 6371-6376). Boston, Massachusetts.

McGovern, S. M. & Gupta, S. M. (2005a). Stochastic and deterministic combinatorial optimization solutions to an electronic product disassembly flow shop. In *Proceedings of the Northeast Decision Sciences Institute – 34th Annual Meeting* (CD-ROM), Philadelphia, Pennsylvania.

McGovern, S. M. & Gupta, S. M. (2005b). Uninformed and probabilistic distributed agent combinatorial searches for the unary NP-complete disassembly line balancing problem. In *Proceedings of the 2005 SPIE International Conference on Environmentally Conscious Manufacturing V* (pp. 81-92), Boston, Massachusetts.

McGovern, S. M. & Gupta, S. M. (2006a). Ant colony optimization for disassembly sequencing with multiple objectives. *The International Journal of Advanced Manufacturing Technology*, *30*(5-6), 481-496.

McGovern, S. M. & Gupta, S. M. (2006b). Deterministic hybrid and stochastic combinatorial optimization treatments of an electronic product disassembly line. In K. D. Lawrence, G. R. Reeves, & R. Klimberg (Eds.), *Applications of management science, Vol. 12*, (pp. 175-197). North-Holland, Amsterdam: Elsevier Science.

McGovern, S. M., Gupta, S. M., & Nakashima, K. (2004). Multi-criteria optimization for non-linear end of lifecycle models. In *Proceedings of the Sixth Conference on EcoBalance* (pp. 201-204). Tsukuba, Japan.

McMullen, P. R. & Frazier, G. V. (1998). Using simulated annealing to solve a multi-objective assembly line balancing problem with parallel workstations. *International Journal of Production Research*, *36*(10), 2717-2741.

McMullen, P. R. & Tarasewich, P. (2003). Using ant techniques to solve the assembly line balancing problem. *IIE Transactions*, *35*, 605-617.

Merkle, D. & Middendorf, M. (2000). An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In *Proceedings of Real-World Applications of Evolutionary Computing, EvoWorkshops 2000: EvoSTIM* (pp. 287-296), Edinburgh, Scotland.

Moore, K. E., Güngör, A., & Gupta, S. M. (1996). Petri net models of flexible and automated manufacturing systems: A survey. *International Journal of Production Research*, *34*(11), 3001-3035.

Moore, K. E., Güngör, A., & Gupta, S. M. (2001). Petri net approach to disassembly process planning for products with complex AND/OR precedence relationships. *European Journal of Operational Research*, *135*(2), 428-449.

O'Shea, B., Kaebernick, H., Grewal, S. S., Perlewitz, H., Müller, K., & Seliger, G. (1999). Method for automatic tool selection for disassembly planning. *Assembly Automation*, *19*(1), 47-54.

Pinedo, M. (2002). *Scheduling theory, algorithms and systems*. Upper Saddle River, New Jersey: Prentice-Hall.

Prakash & Tiwari, M. K. (2005). Solving a disassembly line balancing problem with task failure using a psychoclonal algorithm. In *Proceedings of the ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference* (CD-ROM), Long Beach, California.

Reingold, E. M., Nievergeld, J., & Deo, N. (1977). *Combinatorial algorithms: Theory and practice*. Englewood Cliffs, NJ: Prentice-Hall.

Scholl, A. (1995). *Balancing and sequencing of assembly lines*. Heidelberg, Germany: Physica-Verlag.

Schultmann, F. & Rentz, O. (2001). Environment-oriented project scheduling for the dismantling of buildings. *OR Spektrum*, *23*, 51-78.

Sodhi, M. S. & Reimer, B. (2001). Models for recycling electronics end-of-life products. *OR Spektrum*, *23*, 97-115.

Taleb, K. N., Gupta, S. M., & Brennan, L. (1997). Disassembly of complex products with parts and materials commonality. *Production Planning and Control*, *8*(3), 255-269.

Tang, Y., Zhou, M.-C., & Caudill, R. (2001a). A systematic approach to disassembly line design. In *Proceedings of the 2001 IEEE International Symposium on Electronics and the Environment* (pp. 173-178), Denver, Colorado.

Tang, Y., Zhou, M.-C., & Caudill, R. (2001b). An integrated approach to disassembly planning and demanufacturing operation. *IEEE Transactions on Robotics and Automation*, *17*(6), 773-784.

Thilakawardana, D., Driscoll, J., & Deacon, G. (2003a). A forward-loading heuristic procedure for single model assembly line balancing. In *Proceedings of the 17th International Conference on Production Research* (CD-ROM), Blacksburg, Virginia.

Thilakawardana, D., Driscoll, J., & Deacon, G. (2003b). Assembly line work assignment using a front loading genetic algorithm. In *Proceedings of the 17th International Conference on Production Research* (CD-ROM), Blacksburg, Virginia.

Tiacci, L., Saetta, S., & Martini, A. (2003). A methodology to reduce data collection in lean simulation modeling for the assembly line balancing problem. In *Proceedings of Summer Computer Simulation Conference 2003* (pp. 841-846), Montreal, Canada.

Tiwari, M. K., Sinha, N., Kumar, S., Rai, R., & Mukhopadhyay, S. K. (2001). A petri net based approach to determine the disassembly strategy of a product. *International Journal of Production Research*, *40*(5), 1113-1129.

Toffel, M. W. (2002). End-of-life product recovery: Drivers, prior research, and future directions. In *Proceedings of the INSEAD Conference on European Electronics Take-Back Legislation: Impacts on Business Strategy and Global Trade*, Fontainebleau, France.

Veerakamolmal, P. & Gupta, S. M. (1998). Optimal analysis of lot-size balancing for multiproducts selective disassembly. *International Journal of Flexible Automation and Integrated Manufacturing*, *6*(3&4), 245-269.

Veerakamolmal, P. & Gupta, S. M. (1999). Analysis of design efficiency for the disassembly of modular electronic products. *Journal of Electronics Manufacturing*, *9*(1), 79-95.

Wang, H., Niu, Q., Xiang, D., & Duan, G. (2006). Ant colony optimization for disassembly sequence planning. In *Proceedings of the ASME 2006 Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (CD-ROM), Philadelphia, Pennsylvania.

Wang, H., Xiang, D., Duan, G., & Song, J. (2006). A hybrid heuristic approach for disassembly/recycle applications. In *Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications* (pp. 985-995), Jinan, Shandong, China,.

Zeid, I., Gupta, S. M., & Bardasz, T. (1997). A case-based reasoning approach to planning for disassembly. *Journal of Intelligent Manufacturing*, *8*(2), 97-106.