SOLVING SEQUENCE-DEPENDENT DISASSEMBLY LINE BALANCING PROBLEM USING A HYBRID GENETIC ALGORITHM

Can B. KALAYCI, Pamukkale University
Department of Industrial Engineering
Kinikli Kampusu, Denizli, Massachusetts 20070 TURKEY
cbkalayci@pau.edu.tr, +90(258)-296-3141

Surendra M. GUPTA, Northeastern University
Department of Mechanical and Industrial Engineering, 334 Snell Engineering Center
360 Huntington Avenue, Boston, Massachusetts 02115 U.S.A.
gupta@neu.edu, (617)-373-4846

ABSTRACT

In this paper, we consider a sequence-dependent disassembly line balancing problem (SDDLBP) with multiple objectives that requires the assignment of disassembly tasks to a set of ordered disassembly workstations while satisfying the disassembly precedence constraints and optimizing the effectiveness of several measures considering sequence-dependent time increments. A hybrid genetic algorithm is proposed to solve the SDDLBP.

Keywords: disassembly, sequence-dependent disassembly line balancing, metaheuristics, hybrid genetic algorithm

1. INTRODUCTION

Product recovery becomes increasingly significant. Gungor and Gupta [8], Ilgin and Gupta [12] provide an extensive review of product recovery which involves the recovery of materials and parts from returned or end-of-life (EOL) products. The most critical and time consuming step of product recovery is disassembly. Due to its high productivity and suitability for automation, disassembly line is the most suitable layout for disassembly operations [9].

Disassembly Line Balancing Problem (DLBP) is a multi-objective problem that is described by Gungor and Gupta [10] and has mathematically been proven to be NP-complete by McGovern and Gupta [23] making the goal to achieve the optimal balance computationally expensive. Exhaustive search works well enough in obtaining optimal solutions for small sized instances; however its exponential time complexity limits its application on the large sized instances. An efficient search method needs to be employed to attain a (near) optimal condition with respect to objective functions. Although some researchers have formulated the DLBP using mathematical programming techniques [3] [4] [19], it quickly becomes unsolvable for a practical sized problem due to its combinatorial nature. For this reason, there is an increasing need to use metaheuristic techniques such as genetic algorithms (GA) [14] [23], ant colony optimization (ACO) [1] [7] [21], simulated annealing (SA) [18], tabu search (TS) [15], artificial bee colony (ABC) [17] and particle swarm optimization (PSO) [16]. See McGovern and Gupta [24] for more information on DLBP.
In this paper, we consider a sequence-dependent disassembly line balancing problem (SDDLBP) with multiple objectives that requires the assignment of disassembly tasks to a set of ordered disassembly workstations while satisfying the disassembly precedence constraints and optimizing the effectiveness of several measures considering sequence-dependent time increments. A hybrid genetic algorithm (HGA) is proposed to solve the SDDLBP.

The rest of the paper is organized as follows: In Section 2, notation used in this paper is presented. Problem definition and formulation is given in Section 3. Section 4 describes the proposed HGA algorithm for the multi-objective SDDLBP. The computational experience to evaluate its performance on numerical examples is provided in Section 5. Finally some conclusions are pointed out in Section 6.

2. NOTATION

\( c \quad \) Cycle time (Maximum time available at each workstation)
\( cr \quad \) Crossover rate
\( d_i \quad \) Demand; quantity of part \( i \) requested
\( er \quad \) Elitism rate
\( h_i \quad \) Binary value; 1 if part \( i \) is hazardous, else 0.
\( IP \quad \) Set \((i,j)\) of parts such that task \( i \) must precede task \( j \)
\( i \quad \) Part identification, task count \((1,\ldots,n)\)
\( j \quad \) Part identification, task count \((1,\ldots,n)\)
\( k \quad \) Workstation count \((1,\ldots,m)\)
\( m \quad \) Number of workstations required for a given solution sequence
\( m^* \quad \) Minimum possible number of workstations
\( M \quad \) Sufficiently large number
\( mr \quad \) Mutation rate
\( n \quad \) Number of parts for removal
\( N \quad \) The set of natural numbers
\( ps \quad \) Population size
\( PS_i \quad \) \text{\textit{i}}^{th} \text{ part in a solution sequence}
\( r \quad \) Uniformly distributed random number between 0 and 1.
\( sd_{ij} \quad \) Sequence dependent time increment influence of \( i \) on \( j \)
\( t_i \quad \) Part removal time of part \( i \)
\( t_i' \quad \) Part removal time of part \( i \) considering sequence dependent time increment
\( t_{limit} \quad \) Time limit of the algorithm to be executed
\( ts \quad \) Tournament size
2. PROBLEM DEFINITION AND FORMULATION

The sequence dependent disassembly line balancing problem (SDDLBPP) investigated in this paper is concerned with a paced disassembly line for a single model of product that undergoes complete disassembly. The difference between disassembly line balancing problem (DLBP) and sequence-dependent disassembly line balancing problem (SDDLBPP) is task time interactions. If task \( j \) is performed before task \( i \), its standard time \( t_j \) is incremented by \( sd_{ij} \). This sequence dependent increment measures the prolongation of task \( j \) forced by the interference of already waiting task \( i \).

**Illustrative example:** The precedence relationships (solid line arrows) and sequence dependent time increments (dashed line arrows) for an 8 part PC disassembly process are illustrated in Figure 1 and their knowledge database is given in Table 1. This example is modified from Gungor and Gupta [10].

![Figure 1: 8-part PC example](image)

<table>
<thead>
<tr>
<th>Table 1 Knowledge database for the PC example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part</td>
</tr>
<tr>
<td>PC top cover</td>
</tr>
<tr>
<td>Floppy drive</td>
</tr>
<tr>
<td>Hard drive</td>
</tr>
<tr>
<td>Back plane</td>
</tr>
<tr>
<td>PCI cards</td>
</tr>
<tr>
<td>RAM modules</td>
</tr>
<tr>
<td>Power supply</td>
</tr>
<tr>
<td>Motherboard</td>
</tr>
</tbody>
</table>

Sequence dependencies for the PC example are given as follows: \( sd_{32} = 2, sd_{32} = 4, sd_{56} = 1 \), \( sd_{65} = 3 \). For a feasible sequence \( \{1, 2, 3, 6, 5, 8, 7, 4\} \); since part 2 is disassembled before part 3, sequence dependency \( sd_{32} = 4 \) takes place because when part 2 is disassembled, the obstructing part 3 is still not taken out, i.e., the part removal time for part 2 is increased which results in \( t'_2 = t_2 + sd_{32} = 14 \); similarly since part 6 is disassembled before part 5, sequence dependency \( sd_{56} = 1 \) takes place because when part 6 is disassembled, the obstructing part 5 is still not taken out, i.e., the part removal time for part 6 is increased which results in \( t'_6 = t_6 + sd_{56} = 17 \).

In this paper, the precedence relationships considered are of AND type and are represented using the immediately preceding matrix \([y_{ij}]_{n \times n}\), where

\[
y_{ij} = \begin{cases} 
1 & \text{if task } i \text{ is executed after task } j \\
0 & \text{if task } i \text{ is executed before task } j 
\end{cases}
\] (1)
In order to state the partition of total tasks, we use the assignment matrix \( x_{jk} \), where

\[
x_{jk} = \begin{cases} 
1 & \text{if part } j \text{ is assigned to station } k \\
0 & \text{otherwise}
\end{cases}
\]  

(2)

The mathematical formulation of SDDLBP is given as follows:

\[
\min f_1 = m \\
\min f_2 = \sum_{i=1}^{m} (c - t_i)^2 \\
\min f_3 = \sum_{i=1}^{n} i \times h_{PS_i}, \quad h_{PS_i} = \begin{cases} 
1 & \text{hazardous} \\
0 & \text{otherwise}
\end{cases} \\
\min f_4 = \sum_{i=1}^{n} i \times d_{PS_i}, \quad d_{PS_i} \in N, \forall PS_i
\]  

(3)  

(4)  

(5)  

(6)

Subject to:

\[
\sum_{k=1}^{m} x_{jk} = 1, \quad j = 1, \ldots, n \]  

(7)

\[
\left[ \sum_{i=1}^{n} t_i \right] \leq m^* \leq n \]  

(8)

\[
\sum_{j=1}^{n} \left( t_j + \sum_{i=1}^{n} sd_{ij} \times y_{ij} \right) \times x_{jk} \leq c \]  

(9)

\[
x_{ik} \leq m \sum_{k=1}^{m} x_{jk}, \quad \forall (i, j) \in IP \]  

(10)

The first objective given in equation (3) is to minimize the number of workstations for a given cycle time \( c \) [5]. The second objective given in equation (4) is to aggressively ensure that idle times at each workstation are similar, though at the expense of the generation of a non-linear objective function [23]. As the third objective (see equation (5)), a hazard measure developed to quantify each solution sequence’s performance, with a lower calculated value being more desirable, thereby rewarding the removal of hazardous parts early in the part removal sequence. [23]. As the fourth objective (equation (6)), a demand measure was developed to quantify each solution sequence’s performance, thereby rewarding the removal of high demand parts early in the part removal sequence [23]. The constraints given in; equation (7) ensures that all tasks are assigned to at least and at most one workstation (the complete assignment of each task), equation (8) guarantees that the number of work stations with a workload does not exceed the permitted number, equation (9) ensures that the work content of a workstation cannot exceed the cycle time and equation (10) imposes the restriction that all the disassembly precedence relationships between tasks should be satisfied.
3. PROPOSED HYBRID GENETIC ALGORITHM APPROACH

Since SDDLBP falls into the NP-Complete class of combinatorial optimization problems, it is necessary to use alternative methods in order to reach (near) optimal solutions faster. Metaheuristics such as GA seem to be particularly suited for this task because they process a set of solutions in parallel, possibly exploiting similarities of solutions by recombination that provides an alternative to traditional optimization techniques to locate (near) optimum solutions in complex landscapes.

In the HGA, we use a task based representation. The length of the chromosome is defined by the number of tasks and each gene of the chromosome represents a task. Tasks are assigned to workstations using next fit algorithm according to the task sequence in the chromosome, as long as the predetermined cycle time is not exceeded. Once the cycle time is exceeded, a new work station is opened for assignment, and the procedure is repeated until there are no more tasks to assign. Flow diagram of proposed HGA is depicted in Figure 2.

![Flow diagram of the proposed hybrid genetic algorithm](image)

**Figure 2:** Flow diagram of the proposed hybrid genetic algorithm

Initial solutions are randomly generated for the HGA. We use station-oriented procedure for a solution constructing strategy in which solutions are generated by filling workstations successively one after the other [7].

The fitness functions provide a measure of an individual’s performance in the search space. The proposed HGA algorithm tries to minimize the fitness functions according to the priority of first (1), second (2), third (3) and fourth (4) objective functions, respectively.

The individuals for mating are selected by tournament selection. According to the tournament size defined, tournament size times randomly selected chromosomes are compared to each other and fitter chromosome becomes the parent to be mated in crossover operations.
Three part fragment reordering crossover [2] [20] is used as crossover operation as follows: Two points, which cut each of the parent into three parts (first part, middle part and last part), are generated randomly. As demonstrated in Figure 3, parent-1 is recombined with parent-2 in order to form new children to ensure that the resulting offspring are always feasible. In the disassembly line balancing problem, recombination must guarantee feasibility because of the precedence constraints. The offspring keeps the head and tail parts of the first parent. The middle part is filled in by adding all missing tasks in the order in which they are contained in the second parent.

In the proposed HGA algorithm, interchanging two tasks (SWAP) or inserting a task to a different work station (INSERT) is implemented as a mutation strategy such that the new neighboring solutions are ensured to be feasible. By guaranteeing feasibility in each operation, the necessity of the repair function is prevented. In SWAP, two randomly selected tasks from two randomly selected workstations are exchanged and in INSERT, a randomly selected task from a randomly selected workstation is inserted into another randomly selected workstation while satisfying the precedence constraints.

In the local search step; the best individual of the population from the HGA is used as the starting solution for an embedded tabu search (TS) function. The output of the TS operation is fed to the new population selection process. Whenever an improved solution is found, this solution is taken under the consideration of elitism strategy. The elitism strategy chooses a predetermined number of the best individuals at each generation is used as replacement strategy to create the next generation. The individuals of the new generation may be individuals from the current generation, offspring produced by crossover operator, individuals that underwent mutation or an improved solution found by local search regarding their fitness values.

4. NUMERICAL RESULTS

The proposed algorithm was coded in MATLAB and tested on Intel Core2 1.79 GHz processor with 3GB RAM. After engineering, the program is investigated on two different scenarios for verification and validation purposes. After the full factorial set of experiments, the parameter set with $er = .2$, $cr = .8$, $mr = .2$, $ts = 2$ was determined as the best parameter set while $ps$ parameter was fixed to be 600. The first scenario is for a product consisting of $n=10$ components. The knowledge database and precedence relationships for the components are given in Table 2.
Figure 4, respectively. The problem and its data were modified from McGovern and Gupta [22] with a paced disassembly line operating at a speed which allows \( c = 40 \) s for each workstation to perform its required disassembly tasks. The sequence dependencies for the 10 part product are given as follows: \( sd_{14} = 1, sd_{23} = 2, sd_{41} = 3, sd_{44} = 4, sd_{45} = 4, sd_{54} = 2, sd_{55} = 2, sd_{65} = 4, sd_{60} = 3, sd_{66} = 1 \). While the exhaustive search method was able to find optimal solutions in 215 time on average, the proposed approach was able to successfully find the optimal solution in just over 5 time on average under the restriction of the system specifications given above. Table 3 depicts an optimal solution sequence. The fitness function values of the optimal solution are found to be: \( f_1 = 5, f_2 = 67, f_3 = 5, f_4 = 9605 \). According to this sequence, sequence dependent time increments \( sd_{56}, sd_{69}, sd_{41}, sd_{45}, sd_{32} \) are added to the part removal times of part 6, 6, 1, 5, 2 respectively.

Table 2 Knowledge database for the 10-part product

<table>
<thead>
<tr>
<th>Task</th>
<th>Time</th>
<th>Hazardous</th>
<th>Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>No</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>No</td>
<td>750</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
<td>Yes</td>
<td>295</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>14</td>
<td>No</td>
<td>360</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>No</td>
<td>0</td>
</tr>
</tbody>
</table>

The second scenario consists of a cellular telephone instance with \( n = 25 \) components. The knowledge database and precedence relationships for the components are given in Table 4 and Figure 5, respectively. The problem and its data were modified from Gupta et al. [11] with a disassembly line operating at a speed which allows \( c = 18 \) for each workstation to perform its
required disassembly tasks. The sequence dependencies for the 25 part product are given as the follows: $sd_{45} = 2$, $sd_{34} = 1$, $sd_{67} = 1$, $sd_{69} = 2$, $sd_{76} = 2$, $sd_{78} = 1$, $sd_{87} = 2$, $sd_{96} = 1$, $sd_{13,14} = 1$, $sd_{14,13} = 2$, $sd_{14,15} = 2$, $sd_{15,14} = 1$, $sd_{20,21} = 1$, $sd_{21,20} = 2$, $sd_{22,25} = 1$, $sd_{25,22} = 2$.

Table 4 Knowledge database of cellular telephone instance

<table>
<thead>
<tr>
<th>Part</th>
<th>Task</th>
<th>Part Removal Time</th>
<th>Hazardous</th>
<th>Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antenna</td>
<td>1</td>
<td>3</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>Battery</td>
<td>2</td>
<td>2</td>
<td>Yes</td>
<td>7</td>
</tr>
<tr>
<td>Antenna guide</td>
<td>3</td>
<td>3</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Bolt (type 1) A</td>
<td>4</td>
<td>10</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Bolt (Type1) B</td>
<td>5</td>
<td>10</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Bolt (Type2) 1</td>
<td>6</td>
<td>15</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Bolt (Type2) 2</td>
<td>7</td>
<td>15</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Bolt (Type2) 3</td>
<td>8</td>
<td>15</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Bolt (Type2) 4</td>
<td>9</td>
<td>15</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Clip</td>
<td>10</td>
<td>2</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>Rubber Seal</td>
<td>11</td>
<td>2</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Speaker</td>
<td>12</td>
<td>2</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>White Cable</td>
<td>13</td>
<td>2</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Red/Blue Cable</td>
<td>14</td>
<td>2</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Orange Cable</td>
<td>15</td>
<td>2</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Metal Top</td>
<td>16</td>
<td>2</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Front Cover</td>
<td>17</td>
<td>2</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>Back Cover</td>
<td>18</td>
<td>3</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>Circuit Board</td>
<td>19</td>
<td>18</td>
<td>Yes</td>
<td>8</td>
</tr>
<tr>
<td>Plastic Screen</td>
<td>20</td>
<td>5</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Keyboard</td>
<td>21</td>
<td>1</td>
<td>No</td>
<td>4</td>
</tr>
<tr>
<td>LCD</td>
<td>22</td>
<td>5</td>
<td>No</td>
<td>6</td>
</tr>
<tr>
<td>Sub-keyboard</td>
<td>23</td>
<td>15</td>
<td>Yes</td>
<td>7</td>
</tr>
<tr>
<td>Internal IC Board</td>
<td>24</td>
<td>2</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>Microphone</td>
<td>25</td>
<td>2</td>
<td>Yes</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 5: Cellular telephone instance

Since within the vast search space ($25!$), the exhaustive search is prohibitive due to the exponential growth of the time complexity, the optimal solution is unknown. The proposed HGA algorithm was able to find the best solution given in Figure 6 within a reasonable time ($500t$) under the system specifications given above.
SDDLBP is a recently reported multi-objective NP-complete optimization problem. The main objective of this paper was to solve sequence-dependent disassembly line balancing problem (SDDLBP). A fast, near-optimal, hybrid genetic algorithm was developed and presented in this paper to solve multi-objective SDDLBP. The algorithm was tested on two different scenarios that include a small sized and a large sized disassembly instance. HGA was very fast to solve this small sized instance. For the second instance, it was found out that HGA performed well in terms of solution quality according to the predetermined objective priorities.

REFERENCES


