Journal of Cleaner Production xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of Cleaner Production

journal homepage: www.elsevier.com/locate/jclepro

An efficient metaheuristics for a sequence-dependent disassembly planning

Yaping Ren [a, *], Leilei Meng [b], Chaoyong Zhang [c, **], Fu Zhao [d, e], Ulah Saif [f], Aihua Huang [e], Gamini P. Mendis [e], John W. Sutherland [e]

[a] School of Intelligent Systems Science and Engineering, Jinan University (Zhuhai Campus), Zhuhai, 519070, China
[b] School of Computer Science, Liaocheng University, Liaocheng, 252059, China
[c] Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China
[d] School of Mechanical Engineering, Purdue University, West Lafayette, IN, 47907, USA
[e] Environmental and Ecological Engineering, Purdue University, West Lafayette, IN, 47907, USA
[f] Department of Industrial Engineering, University of Engineering and Technology, Taxila, Pakistan

ARTICLE INFO

Article history:
Received 20 May 2018
Received in revised form
14 September 2019
Accepted 28 September 2019
Available online xxx

Handling Editor: Yutao Wang

Keywords:
Remanufacturing
Disassembly planning
Sequence-dependent
AND/OR graph
Metaheuristics

ABSTRACT

Disassembly planning (DP) is critical in remanufacturing and value recovery from end-of-life products and has attracted increasing attention due to the recent resurgence of research on circular economy. DP problem is NP-hard and its complexity increases exponentially with the size of problem. Sequence-dependent cost due to varying quality of the parts to be retrieved further increases the complexity of DP problems. This paper investigates the DP considering sequence-dependent costs among disassembly operations. A mathematical model is proposed with the objective to maximize the recovery profit using an AND/OR graph (AOG) subject to sequence-dependent costs. A novel two-phase heuristic method is developed to effectively generate feasible disassembly sequence according to the AOG in reasonable computation time. In addition, an improved genetic algorithm (IGA) is proposed to solve the problem, in combination with the presented two-phase heuristic. The performance of IGA is measured on a series of test problem instances against exact methods including CPLEX and an iterative method. Results indicate that IGA successfully find the near-optimal/optimal solutions and outperforms the other methods in terms of computation time. Finally, the proposed method is applied to compute the disassembly solution of a HG5-20 triaxial five speed mechanical transmission. Compared to the existing disassembly solutions of the transmission, the obtained solutions by IGA can shorten about 11% disassembly time and increase by approximately 7% recovery profit.

1. Introduction

Due to the recent resurgence of research on circular economy, remanufacturing has received increasing attention. Remanufacturing is devoted to the value recovery from end-of-life (EOL) products (Harivardhini et al., 2017), which preserve part geometry and function along with the materials and energy consumed during manufacturing of these parts (Ren et al., 2017a). Remanufacturing also reduces waste generation by diverting end-of-life products from landfill (Colledani and Battaïa, 2016). During remanufacturing, the first key step is to disassemble EOL products into components or parts and retrieve usable or repairable subassemblies (Ren et al., 2018a). Therefore, disassembly not only contributes to reducing environmental pollution but also brings economic benefits to industries by reusing/recycling the parts/materials from EOL products (Ondemir and Gupta, 2014).

Disassembly process can be seen as an inverse process of assembly process, in which the subassemblies are disassembled from products (Ren et al., 2018b). Precedence constraints between disassembly operations must be satisfied during the disassembly process in order to generate feasible disassembly sequences (Ren et al., 2017b). Disassembly planning (DP) aims to select the optimal disassembly sequence of an EOL product with the maximization of recovery value and/or processing efficiency (Ren et al., 2017a). A DP involves product representation, disassembly

* Corresponding author.
** Corresponding author.
E-mail addresses: renyp1@163.com (Y. Ren), zcyhust@hust.edu.cn (C. Zhang).

https://doi.org/10.1016/j.jclepro.2019.118644
0959-6526/

Nomenclature

A_j	Set of immediate predecessors of operation j
B_j	Set of exclusive OR operations of operation j
C	Cost per unit time.
c_j	Cost of performing operation j
ct_j	Consumed time of performing operation j
E	Exclusive matrix
e_{jk}	The cell in exclusive matrix E, denoting the exclusive relationship between operations j and k
f	The objective function value i.e. the recovery profit of a product
f_a	The average f value over 20 runs of IGA
f_{best}	The best f value over 20 runs of IGA
f_h	The fitness value of chromosome h
f^*	The f value of the optimal solution obtained from the CPLEX software
Gap	The gap between f^* and f_a
h	The hth chromosome in the current population
MaxIter	The maximum iteration number of IGA
m, n	The index used to represent the position of disassembly operations in the disassembly sequence, m, $n = 1, 2, \ldots, J$
i	The index of a subassembly, $i = 1, 2, \ldots, N$
j, k	The index of an operation, j, $k = 0, 1, 2, \ldots, J$
P	Precedence matrix
PopSize	The population size.
P_c	The crossover probability
P_m	The mutation probability
p_{jk}	The cell in precedence matrix P, denoting the precedence relationship between operations j and k
T	Transition matrix
t_{ij}	The cell in transition matrix T, denoting the relationship between subassembly i and operation j
R_i	Revenue of subassembly i
r	A random number between [0, 1]
s_j	The jth operation in v^1
sp_h	The selection probability of the hth chromosome in the current population
q_j	The jth binary number in v^2
v	Chromosome i.e. a disassembly solution, $v = \{v^1, v^2\}$
v^1	Disassembly operation sequence, $v^1 = \{s_1, \ldots, s_j, \ldots, s_J\}$
v^2	Binary vector, $v^2 = \{q_1, \ldots, q_j, \ldots, q_J\}$
v'	Solution after the first adjustment
v''	Solution after the second adjustment
W1, W2	Subsequence in v
x_j	Binary decision variable, $x_j = 1$, if operation j is performed otherwise, $x_j = 0$
y_{jm}	Binary decision variable, $y_{jm} = 1$, if operation j is the mth performing operation in the disassembly sequence; otherwise, $y_{jm} = 0$
β	The percentage of obtaining the optimum solution in 20 runs of IGA

modeling, and disassembly sequencing. The product representation in the DP is generally depicted via a disassembly graph (DG), such as a directed graph, disassembly tree, AND/OR graph (AOG) and Petri net, etc. These graphs are desired to show all possible disassembly sequences of the given disassembly operations or components/parts. The selection of DGs is utilized to model a feasible disassembly sequence. Based on disassembly modeling, disassembly sequencing can be explored in which possible disassembly sequences of a product are identified and the best one is chosen to be the resulting disassembly solution.

The DP has been studied in the past decades and there are a number of papers on this problem. The DP can be modeled and addressed by mathematical programing, in which the optimal disassembly sequence could be identified by exact solution approaches. For example, Johnson and Wang (1998) presented an integer linear programming problem using a two-commodity network flow formulation and obtained the optimal solution with maximum profit. Kang et al. (2001) established an integer programming model to treat disassembly sequencing as the shortest path problems. Lambert (2007) proposed an iterative procedure based on a binary integer linear programming to tackle a sequence dependent DP, in which the AOG is used to represent a disassembly process. The iterative method enables the search for optima with gradual increase of product complexity.

Nonetheless, these mathematical programming-based approaches inevitably run into challenges as the problem size increases (Meng et al., 2019a). Due to the NP-hardness of the DP, finding global optimum becomes computationally intractable for large scale problems (Ren et al., 2018c). For this reason, heuristics or metaheuristics have been developed in recent years as efficient computation methods to generate near optimal solutions in order to deal with large scale or complex instances (Meng et al., 2019b). For example, Smith et al. (2016) utilized expert rules and cost-benefit analysis to reduce the search space of disassembly solutions. Then, an optimized partial disassembly sequence with maximal economic benefits and minimal environmental costs was determined. Sanchez and Haas (2018) studied the disassembly process of building components. A selective disassembly sequence planning method with rule-based recursive analyses was proposed to achieve the minimization of environmental impact and removal costs during the recovery of target components from buildings. With regards to the metaheuristics in solving DP problems, the most common methods are intelligent algorithms in recent years. For example, Kheder et al. (2015) used a genetic algorithm (GA) to optimize a disassembly process by various criteria including maintainability of components, part volume, tool changes, and disassembly direction changes. Due to the multiple decision criteria, the DP was able to obtain satisfactory results in a short CPU time. Ren et al. (2018b) studied a complicated asynchronous parallel DP. Two matrices, i.e. precedence and collision matrices, were defined to determine feasible disassembly solutions. Based on the matrices, a GA was adapted to efficiently tackle the DP. Tseng et al. (2018) presented a block-based GA to deal with the DP. The GA can efficiently compute the solutions of the problem based on a priority precedence grahph, but it only takes into account the disassembly cost/penalty function. Ren et al. employed a hierarchical disassembly tree to model a parallel DP. A heuristic method was presented to rapidly create feasible disassembly solutions, while an artificial bee colony (ABC) approach was developed to find the Pareto solutions for maximum profit and minimum makespan of the disassembly process. Percoco and Diella (2013) also adopted an ABC to solve a multi-objective DSP, while they simplified the multiple objectives to a weighted sum equation. Guo et al. (2016) modeled a selective DP using Petri nets and solve this problem by a scatter search (SS) algorithm.

Through analyzing the existing literature, we can see that the mathematical programming and exact approaches have been studied earlier for DP problems and can produce optimum

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

solutions for small sized products (Lu et al., 2020). However, they may not be suitable for current industrial practices since there are large-scale or complex EOL products to be disassembled for remanufacturing industries as they might become computationally intractable. To address this issue, heuristics and metaheuristics have been developed to solve large-scale DP problems with reasonable computational cost. However, purely heuristics algorithms might not be able to find high-quality solutions because they could be easily trapped in local optima resulting from simple heuristic rules. Fortunately, heuristics can further be improved by combining with metaheuristics (Lambert and Gupta, 2008).

The AOG-based disassembly modeling has been adopted as one of most widely used approaches to solve the DP, e.g., De Mello and Sanderson (1990), Lambert (2007), Edmunds et al. (2011), and Guo et al. (2018), which can depict the complete set of possible sequences while clearly show the assembly relations among subassemblies (Edmunds et al., 2011). However, limited research work is found which used metaheuristics with an AOG modeling to address DP problems. In addition, most existing studies do not provide an effective method that could efficiently produce possible disassembly sequences by the AOG. Edmunds et al. (2011) described a hierarchical genetic algorithm to automatically generate feasible disassembly sequences using the AOG, but a corresponding mathematical model was not explicitly given. Further, sequence-dependent disassembly costs was not taken into account by Edmunds et al. (2011). In practice, sequence-dependent costs are frequently encountered in the disassembly of a product, and it becomes more complex and challenging if the DP is assumed to be sequence-dependent (Lambert, 2007). Therefore, there is a need to address the sequence-dependent disassembly planning (SDDP) problem with an efficient metaheuristic approach so a near optimal solution can be found in reasonable computational time. This motivates authors to address the SDDP using an AOG based on an improved GA (IGA). The goal of the SDDP is to maximize the recovery profit of an EOL product during the disassembly process. Current research has following contributions:

- In order to describe the complex relationships in an AOG, current research established two matrices, i.e., precedence matrix *P* and exclusive OR matrix *E* are simultaneously defined to represent an assembly product. Moreover, in term of matrices *P* and *E*, a new binary integer programming model is formulated for the SDDP;
- A two-phase heuristic approach is proposed to quickly generate feasible disassembly sequences according to the AOG;
- An efficient metaheuristic i.e. IGA is developed and integrated with the proposed two-phase heuristic to solve the SDDP;
- The proposed algorithm is applied to a series of disassembly problem instances and the performance of IGA is compared with two famous exact methods. Furthermore, a real case study is employed to demonstrate the practical value of IGA.

The rest of this paper is organized as follows. Section 2 describes an AOG-based disassembly problem and establishes its mathematical model. Section 3 describes the proposed approach. Section 4 presents computational experiments and results. Section 5 concludes the research work and describes some future research directions.

2. Problem statement

Disassembly is a process which can be defined as a sequence of operations to separate components and parts from an assembly (Lambert, 1999). Disassembly can be classified into complete and partial disassembly. A complete disassembly dismantles all high-value and all low-value components of the assembly product while a partial disassembly can only removes specific high-value or high-impact components form the assembly product (Smith and Hung, 2015). This partial disassembly leads to better net revenue than the recovery of a complete set of operations which is called selective disassembly. Selective disassembly involves less number of disassembly operations and can get subassemblies which can be used directly during assembly of remanufactured product. Hence, Selective disassembly is significant to save lots of time and cost as compared to complete disassembly. During disassembly of the product, some auxiliary operations are generally needed between two operations, such as tool transformation and product reorientation, etc., which result in the increase of the disassembly cost and depend on the sequence of the disassembly operations, i.e., a sequence-dependent cost (Lambert, 2007).

2.1. Representation of disassembly

In the DP there are different kinds of disassembly graphs (DGs) which are used to represent the node graph of products such as an undirected graph, directed graph, disassembly precedence graph, AOG and Petri net. These are significant to determine graphical representation of the set of all possible sequences of operations involved in the product structure. In the current disassembly sequencing problem there are two conditions which are necessary to meet during representation of the disassembly:

1) The disassembly representation is required to show all the feasible disassembly sequences in a product;
2) The assembly relations are needed to be specified between subassemblies in the graph.

The first condition can be satisfied by most of DGs, but many of them only involve parts and cannot express the assembly relations between subassemblies, such as a disassembly network graph, disassembly tree, and disassembly precedence graph. Similarly, Petri net has been used to optimize selective disassembly sequences and maximize disassembly profit by Guo et al. (2016) which included all subassemblies. However, it is relatively complex to construct a disassembly Petri net and build its disassembly model. DG is also required to be readable and describe disassembly relations as simple as possible. An AOG is able to conveniently depict the connection and precedence relations among subassemblies. In this paper, the AOG is applied to illustrate the disassembly process of a product. Note that the fasteners, e.g., screws and bolts, are not shown in the AOG. This simplification significantly reduces the complexity of the AOG while the representation of the disassembly information is not affected in the SDDP. The basic ideas of AOG are explained with Bourjault's ballpoint as seen in Fig. 1 which is converted to a disassembly AOG as shown in Fig. 2 (Lambert, 2007). It can be observed from Fig. 1 that the ballpoint consists of parts A, B, C, D and E.

In an AOG, each disassembly operation is expressed with a hyper arc, which consists of two connected arcs. The two connected arcs

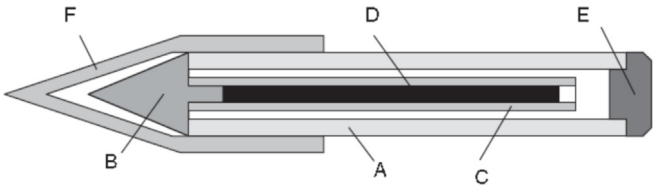


Fig. 1. Assembly drawing of Bourjault's ballpoint.

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

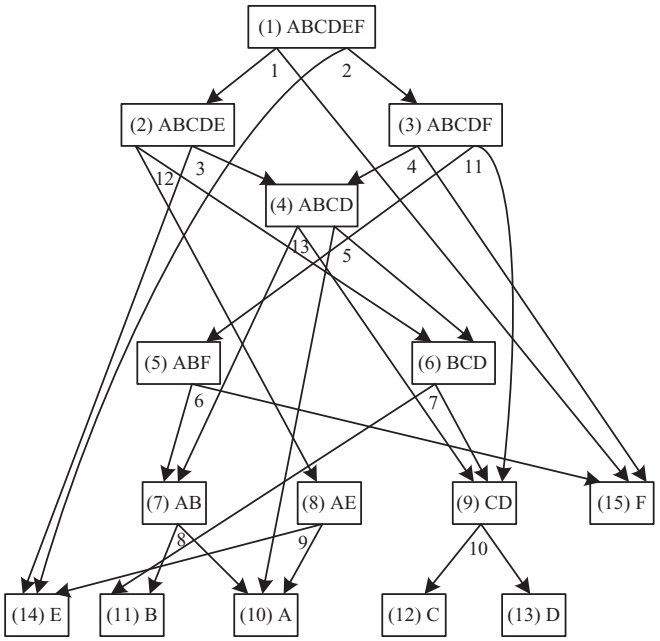


Fig. 2. The AOG of Bourjault's ballpoint.

beginning from a parent subassembly are directed towards two disassembles and these arcs are representing AND relation. Fig. 2 illustrates an example where each operation is led to two disassembles, however, the method presented is not limited to only two disassembles. This is compatible with practical situations in which more than two subassemblies may be resulted from one disassembly operation. A disassembly operation is numbered by 1, 2, …, and J, where J represents the number of operations involved in disassembly of the product. Nodes in the AOG represent existent subassemblies for an assembly in practice. Each node is in one-to-one correspondence to a subassembly of a product from (1) to (N) and N is the number of subassemblies. It can be seen from Fig. 2 that the disassembly AOG has multiple random disassembly paths (i.e., OR relation) each of which represents a feasible disassembly sequence which can be presented by the disassembly operations, e.g., 2 → 11 → 6 → 8 or 1 → 12 → 7 → 10.

A disassembly sequence can be represented by a series of operations in the AOG. Moreover, all feasible disassembly sequences are presented in the AOG and in order to determine a feasible sequence, the following conditions are considered.

1) The precedence relation

It can be seen from Fig. 2 that by performing operation 1, an assembly (1) is disassembled into (2) and (15), while (6) and (8) are obtained after separating (2) through operation 12. This indicates that operation 1 is the immediate predecessor of operation 12. A precedence matrix P=[p_{jk}] is constructed here to represent the immediate precedence relationship among operations, and p_{jk} is denoted as follows:

$$p_{jk} = \begin{cases} 1, & \text{if operation } k \text{ can be performed immediately after operation } j \\ 0, & \text{otherwise} \end{cases}.$$

Thus, precedence matrix P of the ballpoint can be given in Table 1. From precedence matrix P indicated in Table 1, it can be seen that operation j represented by row j precedes operation k represented by column k, where j and $k \in \{1, 2, \ldots, J\}$, if $p_{jk} = 1$. For

Table 1
Precedence matrix P of the ballpoint.
2) The exclusive OR (EOR) relation

p_{jk}	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	1	0	0	0	0	0	0	0	0	1	0
2	0	0	0	1	0	0	0	0	0	0	1	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	1
4	0	0	0	0	1	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	1	0	0	0	1	0	0	0
12	0	0	0	0	0	0	1	0	1	0	0	0	0
13	0	0	0	0	0	0	0	1	0	1	0	0	0

example, from Table 1, as shown in row 11, $p_{11,\,6} = 1$ and $p_{11,\,10} = 1$, which means that operations 6 and 10 can be done immediately after operation 11. Therefore, operation 11 is the immediate predecessor of operations 6 and 10, and operations 6 and 10 are the immediate successors of operation 11.

In a disassembly process, it is well noted that one parent subassembly can be only removed once, and one part can be obtained from only one parent in practice. For instance, it can be seen from Fig. 2 that subassembly (3) can be dismantled via operation 4 or 11, while subassembly (9) can be generated by operation 7, 11 or 13. In fact, the operations 4 and 11 (7, 11 and 13) are EOR, which indicates that only one of them can be carried out in one time. Operations 4 and 11 are defined as the EOR successors of operation 2, and operations 7, 11 and 13 are defined as the EOR predecessors of operation 10. An exclusive matrix E=[e_{jk}] is established here to depict the EOR relations among operations in the AOG, and e_{jk} is expressed as:

$$e_{jk} = \begin{cases} 1, & \text{if operation } k \text{ can be performed immediately} \\ 0, & \text{otherwise} \end{cases}$$

For example, matrix E of the ballpoint is shown in Table 2. It can be seen from the exclusive matrix that when $e_{12} = -1$, it indicates that operations 1 and 2 are exclusive and either of them can be performed but both cannot be performed simultaneously. Furthermore, matrix E is a symmetric matrix distinguishing from P. Notice that the EOR relation may occur between non-neighboring operations, e.g., operations 1 and 6, i.e., $e_{16} = -1$ and $e_{61} = -1$.

In the current problem, a feasible disassembly sequence not only satisfies the precedence relation but also deals with the EOR relation. The EOR relation is uncommon in other DGs and exclusive matrix E is firstly proposed in the current research. Note that an

Table 2
Matrix E of the ballpoint.

e_{jk}	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	−1	0	−1	0	−1	0	0	0	0	−1	0	0
2	−1	0	−1	0	0	0	0	0	−1	0	0	−1	0
3	0	−1	0	−1	0	−1	0	0	−1	0	−1	−1	0
4	−1	0	−1	0	0	−1	0	0	−1	0	−1	−1	0
5	0	0	0	0	0	−1	0	−1	−1	0	−1	−1	−1
6	−1	0	−1	−1	−1	0	−1	0	−1	0	0	−1	−1
7	0	0	0	0	0	−1	0	−1	0	0	−1	0	−1
8	0	0	0	0	−1	0	−1	0	−1	0	0	−1	0
9	0	−1	−1	−1	−1	−1	0	−1	0	0	−1	0	−1
10	0	0	0	0	0	0	0	0	0	0	0	0	0
11	−1	0	−1	−1	−1	0	−1	0	−1	0	0	−1	−1
12	0	−1	−1	−1	−1	−1	0	−1	0	0	−1	0	−1
13	0	0	0	0	−1	−1	−1	0	−1	0	−1	−1	0

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

operation may be immediate and EOR predecessors of one operation at the same time, e.g., operations 5 and 12 are both immediate and EOR predecessors of operation 7.

2.2. Mathematical model

Disassembly operations are sequenced for an efficient disassembly activity, and the recovery profit is obtained from subassemblies. To compute the profit conveniently, a transition matrix T is utilized here which is proposed by Lambert to depict the relationship between subassemblies and operations in the AOG (Lambert, 1999). Matrix T of the ballpoint is illustrated in Table 3. The cells of the matrix are expressed via t_{ij}, in which index i denotes subassembly i, and index j denotes disassembly operation j. t_{ij} is defined as follows:

$$t_{ij}=\begin{cases}1, & \text{if subassebmly } i \text{ is obtained by operation } j\\ -1, & \text{if subassebmly } i \text{ is obtained via operation } j\\ 0, & \text{otherwise}\end{cases}.$$

Table 3 provides transition matrix T of the ballpoint. From Table 3, we can see that parent (1) (ABCDEF) is separated via operation 2, hence $t_{12}=-1$. Since this operation generates two subordinates (3) (ABCDF) and (14) (E), $t_{32}=1$ and $t_{14,\ 2}=1$. In T, operation 0 refers to the initial disassembly operation which is employed to generate the assembly product such that $t_{10}=1$ in Table 3.

The proposed model is aimed to decide on an optimal sequence with the disassembly level for the SDDP. The notations, index set, parameters and decision variables used in the current study are illustrated below:

(1) i: The index of a subassembly, $i \in \{1, 2, \ldots, N\}$, where N is the number of all nodes in a given AOG;
(2) j, k: The index of an operation, j, $k \in \{0, 1, 2, \ldots, J\}$, where operation 0 represents the initial disassembly operation, and J is the number of operations;
(3) m, n: The index used to represent the position of disassembly operations in the disassembly sequence i.e., m, $n \in \{1, 2, \ldots, J\}$.
(4) A_j: Set of immediate predecessors of operation j;
(5) B_j: Set of EOR operations of operation j;
(6) C_1: Operation cost per unit time;
(7) C_2: Extra cost per unit time between two adjacent operations;
(8) R_i: Revenue of subassembly i;
(9) ct_j: Consumed time of performing operation j;
(10) et_{jk}: Extra time from operation j to operation k, which is sequence-dependent;
(11) t_{ij}: The value of cell (i, j) in transition matrix T, and it may enter -1, 0, or 1.

Table 3
Transition matrix T of the ballpoint.

t_{ij}	0	1	2	3	4	5	6	7	8	9	10	11	12	13
(1)	1	−1	−1	0	0	0	0	0	0	0	0	0	0	0
(2)	0	1	0	−1	0	0	0	0	0	0	0	0	−1	0
(3)	0	0	1	0	−1	0	0	0	0	0	0	−1	0	0
(4)	0	0	0	1	1	−1	0	0	0	0	0	0	0	−1
(5)	0	0	0	0	0	0	−1	0	0	0	0	1	0	0
(6)	0	0	0	0	0	1	0	−1	0	0	0	0	1	0
(7)	0	0	0	0	0	0	1	0	−1	0	0	0	0	1
(8)	0	0	0	0	0	0	0	0	0	−1	0	0	1	0
(9)	0	0	0	0	0	0	0	1	0	0	−1	1	0	1
(10)	0	0	0	0	0	1	0	0	1	1	0	0	0	0
(11)	0	0	0	0	0	0	0	1	1	0	0	0	0	0
(12)	0	0	0	0	0	0	0	0	0	0	1	0	0	0
(13)	0	0	0	0	0	0	0	0	0	0	1	0	0	0
(14)	0	0	1	1	0	0	0	0	0	1	0	0	0	0
(15)	0	1	0	0	1	0	1	0	0	0	0	0	0	0

Decision variables:

(1) x_j: $x_j=1$, if operation j is performed; otherwise, $x_j=0$;
(2) y_{jm}: $y_{jm}=1$, if operation j is the mth performing operation in the disassembly sequence; otherwise, $y_{jm}=0$.

We assume that the required data in the disassembly model is given at hand, such as c_j and R_i, and based on the AOG we formulate a selective disassembly optimization model for SDDP as follows.

$$\text{Max}\quad f=\sum_{i=1}^{J}\sum_{j=0}^{N}t_{ij}R_ix_j-\sum_{j=1}^{J}C_1ct_jx_j-\sum_{j=1}^{J}\sum_{k=1}^{J}\times\sum_{m=1}^{J-1}C_2et_{jk}y_{jm}y_{k,m+1} \tag{1}$$

$$\text{s.t.}\quad x_0=1 \tag{2}$$

$$x_j+x_k\le 1,\ \ j\in B_k,\ \ k=0,\ 1,\ 2,\ \ldots,\ J; \tag{3}$$

$$\sum_{j\in A_k}\sum_{m=1}^{n-1}y_{jm}\ge y_{kn},\ \ k=1,\ 2,\ldots,\ J; \tag{4}$$

$$x_j=\sum_{m=1}^{n-1}y_{jm},\ \ j=1,\ 2,\ldots,\ J; \tag{5}$$

$$\sum_{m=1}^{n-1}y_{jm}\le 1,\ \ m=1,\ 2,\ldots,\ J; \tag{6}$$

$$x_j,\ y_{jm}\in\{0,1\},\ \ j,k=1,\ 2,\ldots,\ J; \tag{7}$$

Eq. (1) depicts the objective function of the SDDP which aims at maximizing the recovered profit of an EOL product. The first term in Eq. (1) is the total revenue to separate a product, the second one is the disassembly cost of all operations, and the third one represents the extra cost between two adjacent operations. Constraint (2) denotes that the initial disassembly operation must be done. Constraint (3) is used to deal with the EOR relations between operations which implies that the exclusives operations are inhibited in a sequence. Constraint (4) guarantees that an operation can be performed after at least one of its predecessors has been completed. Constraint (5) shows the relation between decision variables. Constraint (6) ensures that at most one operation can be done at that time during the disassembly process. And constraint (7) represents that decision variables can take 0 or 1.

3. The proposed algorithm

SDDP is similar to the asymmetric traveling salesman problem (TSP) due to its sequence-dependent characteristic (Lambert, 2006). However, TSP is NP-hard which means that when the SDDP problem size (i.e., the number of subassemblies/operations in a product) increases, the solution space is exponentially increased and it is difficult to find an optimum solution in a reasonable time. Furthermore, SDDP is complicated problem as compared to TSP as there are some additional constraints which are taken into

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

consideration during determining sequencing solution in DP, such as precedence and EOR relations. Thus, this work proposes an efficient metaheuristic approach (namely, IGA) to address the SDDP, and the proposed algorithm consists of six tasks as follows:

3.1. Solution encoding

The encoding form of a solution directly affects the efficiency of the proposed method, and it should enable to illustrate a solution effectively. In this paper an AOG is adopted to present all possible disassembly sequences of a product. The precedence and EOR relations exist in the graph especially for the EOR relation which is studied less in the previous work. Herein, the double-vector list structure is utilized to form a disassembly solution, and it is expressed by $v = \{v^1, v^2\}$ $v^1 = \{s_1, \ldots, s_j, \ldots, s_J\}$ represents a disassembly operation sequence in which each element, which is an integer from 1 to J, corresponds to one task, and J is the number of tasks in a product. $v^2 = \{q_1, \ldots, q_j, \ldots, q_J\}$ represents a binary vector which includes J binary values. If s_j in v^1 is performed, $q_j = 1$; otherwise, $q_j = 0$. For instance, $v^1 = \{2, 4, 3, 5, 8, 6, 7, 9, 1\}$ and $v^2 = \{1, 1, 0, 0, 1, 0, 0, 1, 0\}$ mean that operations 2 (s_1), 4 (s_2), 8 (s_5) and 9 (s_8) in v^1 are carried out since q_1, q_2, q_5 and q_8, are equal 1 in v^2.

3.2. Initialization of population

Initialization of population is key to create feasible solutions (chromosomes) of the optimization problem. Randomly initialization of solutions of the disassembly sequence for SDDP is employed in the proposed algorithm to generate diversified solutions in the search space. Here, it is essential to ensure the precedence relation and inhibit the appearance of the EOR operations for a feasible sequence. In other words, constraints (3) and (4) are required to be satisfied when initializing the population. Hence, a two-phase heuristic approach is developed in the current research to initialize individuals. It consists of two heuristic procedures, and its detailed steps are presented in the following.

Phase 1 Eliminate the EOR operations in v:

Differing from most of DGs, the EOR relation has to be considered in a disassembly AOG. Hence, we suggest a heuristic technique to deal with this issue, and Exclusive matrix E is applied here. The heuristic process is as follows:

Step 1: Start.

Step 2: Randomly generate v, i.e., v^1 and v^2, where v^1 is a disassembly operation sequence including the integer numbers from 1 to J, and v^2 is a binary array in which all elements are initialized to be 1.

Step 3: Set $j = 1$.

Step 4: If q_j is equal 0, go to Step 7. Otherwise, according to matrix E, identify the exclusive operations of the jth operation (s_j) from s_{j+1} to s_J in v^1, and store them into a set G.

Step 5: If G is empty, go to Step 7. Otherwise, go to Step 6.

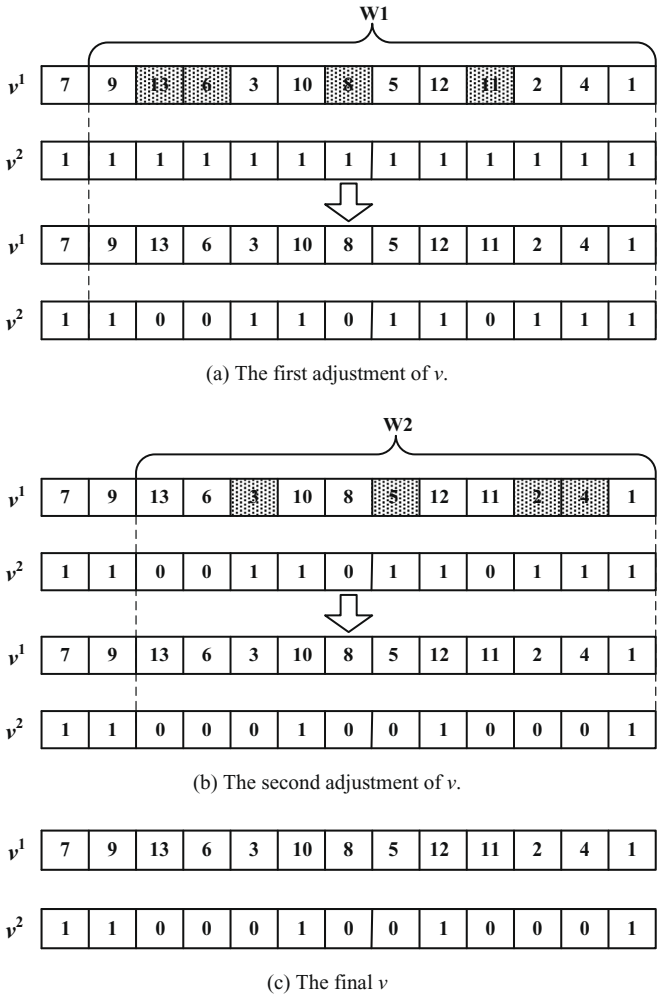
Step 6: If there are some exclusive operations in G whose corresponding elements in v^2 are 1, the elements are reset as 0. Otherwise, G becomes an empty set.

Step 7: $j = j + 1$.

Step 8: If j is equal J, go to Step 9. Otherwise, go to Step 4.

Step 9: Stop the procedure.

In Fig. 3, a random solution v of the ballpoint is used to explain the steps above. Fig. 3 (a) illustrates the first adjustment of v. It can be seen that all operations are carried out since all elements in v^2 are initialized as 1 (Step 2). Clearly it is not practicable for v with respect to the EOR operation. According to E, s_3, s_4, s_7 and s_{10} marked in shade are exclusive with s_1 (operation 7) in the subsequence W1, which are memorized into G (Step 4). The operations in



(a) The first adjustment of v.

(b) The second adjustment of v.

(c) The final v

Fig. 3. The adjustment for a random solution v of the ballpoint in **phase 1**.

set G are prohibited in sequence v, i.e., q_3, q_4, q_7 and q_{10} become 0 in v^2 (Step 6). Then, follow Steps 7 and 8, the next iteration starts from Step 4. Repeat Steps 4–8, the subsequence W2 in v is adjusted, see Fig. 3 (b). After the third iteration of Steps 4–8, we can obtain the resulting v without exclusive operations, see Fig. 3 (c).

Phase 2 Guarantee the precedence constraint:

The first phase focuses on dealing with the EOR relation in v, and a set of operations performed are therefore yielded. However, it is not ensured that v meets the prior relation. For example, in Fig. 3 (c), the final disassembly sequence is $7 \rightarrow 9 \rightarrow 10 \rightarrow 12 \rightarrow 1$ which is infeasible for the ballpoint. Note that the disassembly operations are performed form left to right side in v. However, the operations performed enable to form a feasible sequence through a random adjustment procedure. Moreover in matrix P, the priorities of all operations can be seen whereas it is relatively complicated to sequence operations only based on matrix P. Therefore, the number of immediate predecessors (NIPs) of each operation are created to easily choose some operations with the highest priority. It is well noted that the priorities among operations are dynamically varied as the disassembly operations are performed one by one. Further, NIPs of each operation are reduced as its immediate predecessors are removed.

The initial information of immediate predecessors for all operations corresponding to Fig. 3 are listed in Table 4 according to matrix P. The symbol "/" used in Table 4 represents the

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

Table 4
The initial information of immediate predecessors for operations.

No. of operations	1	2	3	4	5	6	7	8	9	10	11	12	13
Immediate predecessors	∅	/	/	/	/	/	12	/	12	7	/	1	/
NIPs	0	/	/	/	/	/	1	/	1	1	/	1	/

corresponding operation is not implemented in *v*, and "∅" denotes that the corresponding operation has no immediate predecessors at that time. Moreover, the NIPs for an operation is set to be −1 if it is removed. When operation 1 is disassembled at first, the updated data of Table 4 is presented in Table 5. It can be seen from Table 5, that the data of operations 1 and 12 is changed due to the alteration of their precedence relations. Further, an operation is able to be dismantled when its NIPs becomes 0. Hence, one operation is randomly selected to be disassembled from the operation(s) whose NIPs are 0 until no more operations left to be performed. Finally, a feasible disassembly sequence is produced for the product. The random procedure for initializing an individual is described as:

Step 1: Start.

Step 2: Extract the operations which are performed in *v* after **Phase** 1, and aggregate them into a new array *v'*, such as *v'*: 7 → 9 → 10 → 12 → 1 in Fig. 4 provided from Fig. 3 (c).

Step 3: In term of matrix *P*, initialize NIPs.

Step 4: If there are no more operations left, go to Step 6. Otherwise, based on NIPs, choose one operation with the highest priority as the current performing operation, and go to Step 5.

Step 5: Update matrix *P* and NIPs; Go back to Step 4.

Step 6: Stop the procedure, and a feasible disassembly sequence *v''* is derived, such as *v''* in Fig. 4.

Hereto, the initialization of a random *v* can be completed through **Phases 1** and **2**.

3.3. Assessment of fitness value

Once the feasible solutions of SDDP are generated via the two-phase method, it is necessary to evaluate a solution *v''*, and decode it into a disassembly sequence for a product. However, *v''* is a complete disassembly sequence, which indicates that all parts are separated from a product and the highest disassembly level (HDL) is reached here. For instance, in Figs. 4 and 1 → 12 → 7 → 10 → 9 represents a complete disassembly process and its disassembly level (DL) is 5, which is equal to its HDL. DL is not more than HDL in a partial/selective disassembly. Thus, DL, which is a random integer number between 1 and HDL, is used to decide on a partial sequence based on *v''*. Still in case of *v''* in Fig. 4, if DL = 3, the partial disassembly sequence is 1 → 12 → 7. Then, according to Eq. (1) in the model, the profit value is easily calculated. We utilize the profit of each solution to evaluate its fitness value, and the greater the profit, the better the disassembly solution.

3.4. Selection operation

The selection is employed to choose chromosomes for reproduction, and the roulette wheel selection is well noted and adopted in this stage which is a fitness-based strategy. The selection probability sp_h of the *h*th chromosome in the current population is

Table 5
The updated information of immediate predecessors for operations.

No. of operations	1	2	3	4	5	6	7	8	9	10	11	12	13
Immediate predecessors	∅	/	/	/	/	/	12	/	12	7	/	∅	/
NIPs	−1	/	/	/	/	/	1	/	1	1	/	0	/

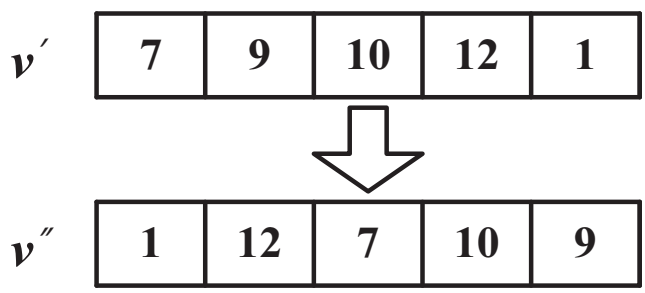


Fig. 4. The random adjustment from *v'* to *v''* in **Phase 2**.

formulated as:

$$sp_h = \frac{f_h}{\sum_{h=1}^{popsize} f_h}, \tag{8}$$

where f_h and *PopSize* represent the fitness value of chromosome *h* and the population size, respectively. In this method of selection, the individuals with good fitness values have more "survival" opportunities to be the offspring, which contributes to keeping the better individuals.

3.5. Crossover operation

The crossover is the crucial procedure to change the search solution space in GA, and its implementation efficiency can directly determine the global search ability in the proposed method. In this work the crossover is executed to not only obtain new chromosomes by exchanging genes carried in the parents but also ensure the feasibility of the offspring.

In order to keep the integrality of gene recombination, the initial double-vector link structure *v* is applied to the crossover process, and precisely, only v^1 is crossed as v^2 is initialized as a constant vector. The current individual' v^1 is deemed as one parent, and another one is randomly obtained from the rest individuals of the current population. Herein, we adopt a double-point crossover operation for two parents, and subsequently two offspring chromosomes are created. Furthermore, both offspring are dealt with **Phases** 1 and 2 similar to the initialization to guarantee the feasibility, while the better one is chosen as the new individual through a simple greedy selection.

3.6. Mutation operation

Mostly, the chromosomes are easily trapped in the local optimal solutions, and the mutation facilitates seeking the neighboring solutions and enhancing the diversity of the population. The mutation operation generally leads to a negation of the binary gene locus or slight perturbation in the chromosome with probability P_m.

For current research problem, however, the mutation may not be effective for the initial *v* since: (1) each gene in v^2 must be initialized as 1 and cannot be negated; and (2) it is relatively possible to generate the same disassembly sequence *v'* after the slight perturbation of v^1 with respect to the EOR relation, e.g., in Fig. 3 (a), if a swap occurs between two genes in v^1, *v'* is also likely to be 7 → 9 → 10 → 12 → 1, and the probability is more than 0.59 by a rough calculation. Furthermore, it is difficult for *v'* to inherit and keep the correct genetic information because *v'* need be further adjusted by **Phase** 2, which means that it is little of significance to mutate *v'*. Therefore, a two-point swap method is applied to the mutation of *v''*. Likewise, it is necessary to ensure the feasibility of *v''* when inducing gene mutation. Fortunately, only using **Phase** 2 of

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

the two-phase heuristic enables to produce a feasible sequence for a mutated *v*".

3.7. Main algorithm

Hereto, the main steps of IGA have been depicted, and it is implemented in Fig. 5. Notice that IGA adopts the maximum iteration count *MaxIter* as a termination criterion, i.e., when it is reached, output the solution and terminate the algorithm.

4. Experimental results and analysis

This section describes the computational experiments and results to evaluate the performance of IGA. The experiments are performed in MATLAB 7.14 and runs on an Intel(R) Core (TM) i5 CPU (3.20GHz/8.00G RAM) PC with a Windows 7 operating system. The performance of IGA is tested by different scales of instances, including a large-scale real case. For the required data in our experiments but not collected, we will randomly create this for our problem. First, a series of experiments are executed to find the best value combination of parameters (crossover probability P_c and mutation probability P_m) by running IGA, in which three combinations of parameters are investigated using different size problems. Then, the proposed algorithm is compared with two exact approaches, where one is the default exact algorithm from CPLEX 12.6.3 based on the proposed model, and another is an iterative method proposed by Lambert (2007). Finally, we adopt a large-scale real case to demonstrate the practical value of our proposed method.

4.1. Tuning of algorithm parameters

Some parameters are generally predetermined for IGA, such as the crossover and mutation probabilities P_c and P_m. In order to find the best combination of parameters, a series of experiments are attempted to run IGA using three different disassembly products with different size. Product 1 is taken from a simple case in the literature (De Mello and Sanderson, 1990), and cell phone 1 used by Behdad et al. is selected as product 2 (Behdad et al., 2010), and product 3 is an assumed product proposed by Koc et al. (2009). Each combination of parameters, which mainly involves P_c and P_m, is tested ten times for each instance, and the average f value f_a of IGA is shown in the last column of Table 6.

In Table 6, according to the problem size, *MaxIter* and *PopSize* are set to be smaller values for observing the experimental results conveniently, since the same optimal/near-optimal solutions are easily sought with different combination of P_c and P_m when the greater *MaxIter* and *PopSize* are defined. It can be seen from Table 6 that for each instance there is no statistically significant difference in terms of f_a. However, the combination of $P_c = 0.8$ and $P_m = 0.2$ overall outperforms the other two combinations although the difference is very small between them, which can be clearly demonstrated by Fig. A4 in **Appendix**. Hence, $P_c = 0.8$ and $P_m = 0.2$ are employed in the following experiments.

Procedure: The improved GA

Input: SDDP data set, GA parameters

Output: a near-optimal solution

```
Start
Initialize
    Initialize the GA parameters: the maximum iteration count MaxIter, the size of population PopSize, the crossover and
    mutation probabilities Pc and Pm;
    Initialize population through the two-phase heuristic in section 3.2;
Evaluate each chromosome from the population according to section 3.3;
While (the termination condition has been not reached)
    Selection
        Perform the roulette wheel selection for the current population and determine PopSize individuals;
    While (not reach the PopSize)
        Crossover
        If (r < Pc) then // r is a random number between 0 and 1
            Execute the crossover operation in section 3.5;
        End if;
        Mutation
        If (r < Pm) then
            Execute the mutation operation in section 3.6;
        End if;
    End while;
Obtain new population and evaluate fitness;
Update the current best chromosome;
End While;
Output a near-optimal solution;
End;
```

Fig. 5. The pseudocode of IGA.

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

Table 6
Influence of *Pc* and *Pm* in the proposed approach for three products.

No. of products	Number of subassemblies	Number of operations	*MaxIter*	*PopSize*	P_c	P_m	f_a
1	12	15	10	10	0.6	0.05	8.462
					0.7	0.10	8.428
					0.8	0.20	8.562
2	18	10	4	2	0.6	0.05	1.869
					0.7	0.10	1.880
					0.8	0.20	1.880
3	14	23	20	20	0.6	0.05	11.816
					0.7	0.10	11.917
					0.8	0.20	11.917

4.2. Comparisons of IGA and the exact method from CPLEX 12.6.3

Metaheuristics usually return a 'good enough' solution while mathematical programming generates the exact optimum solution. It is of significance to validate whether the proposed metaheuristics can yield the optimal solution as the exact solution. The solution gap between IGA and the exact algorithm can effectively demonstrate the performance of IGA. Therefore, according to the mathematical model in section II-B, the default exact method of CPLEX 12.6.3 is used to search exact solutions of the problems. Meanwhile, IGA is executed 20 times (sample size) each in order to have sufficient statistical data for comparison since the algorithm has probabilistic and randomized features. Apart from the above three products, a radio set (product 4) (Lambert, 1999), which includes 29 subassemblies and 30 operations, is also considered, and its AOG is given in Fig. A1 of the Appendix.

Table 7 provides the solution results of the CPLEX solver for products 1, 2, 3 and 4, respectively, and the best result f_{best} and f_a of running 20 times of IGA are also shown with different *MaxIter* and *PopSize* in each instance. In Table 7, β is the percentage of obtaining the optimum solution in 20 trials, and Gap are calculated by Gap $= |f^* - f_a|/f^* \times 100\%$, where f^* is the f value of the optimal solution obtained from the CPLEX software. "-" represents that the optimum cannot be found within an acceptable CPU time which is set to 12 h.

From Table 7, we can conclude that:

1) The proposed IGA successfully solves the SDDP, and enables to seek the optimum solution similar to the solutions obtained from CPLEX for each considered problem. Furthermore, it is more possible to find the optimal solution with the increase of *MaxIter* and *PopSize*, which can be clearly shown by the values of β for each instance. Moreover, f_a is greatly approaching f^* when *MaxIter* and *PopSize* reach a certain degree, which is demonstrated in Fig. A5. of **Appendix.** Especially for product 3, the average f value is equal f^* when *MaxIter* = 40 and *PopSize* = 20.
2) In products 1, 2 and 3, the CPLEX solver can obtain the optimum solution within an acceptable time. However, the required CPU time is unmanageable for product 4, specifically, the exact algorithm has not terminated in 12 h. It reveals the poor efficiency of the exact method for handling some larger-scale instances, precisely, the used time grows exponentially as the problem size increases. Fortunately, for product 4 the 'good enough' solutions are rapidly produced by IGA. Furthermore, the proposed approach also addresses the smaller size problems efficiently according to the average CPU time of products 1, 2, and 3. Thus, in terms of the computation speed, IGA outperforms the CPLEX method greatly.
3) The performance of IGA is impacted by the size of *MaxIter* and *PopSize*, i.e., the larger the values of *MaxIter* and *PopSize*, the better performance of IGSA. Moreover, the running time may moderately expand when *MaxIter* and *PopSize* are greater, while the resultant solutions may be more satisfactory, which can be validated by the values of f_a and Gap for each instance.

4.3. Comparisons with an iterative method

The comparison results show the feasibility and effectiveness of

Table 7
Results comparison between the exact method and IGA.

No. of Products	CPLEX		IGA						
	f^*	CPU time (s)	*MaxIter*	*PopSize*	f_{best}	f_a	β	Gap	Average CPU time (s)
1	8.64	1.92	4	4	8.64	7.376	15%	14.63%	0.004
			5	5	8.64	8.384	30%	1.97%	0.007
			10	5	8.64	8.479	50%	1.86%	0.013
			10	10	8.64	8.601	90%	0.45%	0.023
2	1.963	1.38	2	2	1.963	1.826	60%	6.98%	0.003
			4	2	1.963	1.871	80%	4.67%	0.005
			4	4	1.963	1.940	95%	1.17%	0.009
3	12.22	13.22	10	10	12.22	11.241	35%	8.01%	0.037
			20	10	12.22	11.571	65%	5.31%	0.070
			10	20	12.22	11.470	55%	6.14%	0.073
			40	20	12.22	12.220	100%	0.00%	0.271
4	–	–	20	10	14.8824	14.7573	25%	0.84%	0.102
			20	20	14.8824	14.7997	35%	0.56%	0.201
			50	20	14.8824	14.8436	50%	0.26%	0.481
			50	50	14.8824	14.8748	90%	0.05%	1.1662

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

IGA in the previous section. To further demonstrate the performance of the proposed approach, it is also compared to an iterative algorithm reported in the literature (Lambert, 2007). The iterative procedure is developed to deal with a relaxed binary integer linear programming for SDDP. Although repetitive calculation of the binary linear programming problem is also required, a considerable alleviation in CPU time can be observed. In contrast with the previous work, this clearly shortens the solving time of SDDP and quickly converges to the exact optimum especially for the products with increased complexity through the series of intermediate solutions that are generated by the iterative process. The detailed steps of the iterative method can be referred in the literature (Lambert, 2007).

Four scenarios are considered here to investigate the performance of the iterative method and IGA. The first SDDP scenario is taking the ballpoint in Fig. 2 as an example. The second SDDP scenario is the radio set, i.e., product 4, that is applied in Table 7. The third and fourth scenarios are the reduced structures modified from an imaginary product with 31 operations and 63 subassemblies including components and parts (Lambert, 2007), and both AOGs are depicted in Figs. A2 and A3 of the Appendix, respectively.

The computational results of the iterative approach and IGA for four same scenarios are given in Table 8 and Table 9, respectively. The iterative procedure is run in the CPLEX solver rather than the XA solver of the original paper. Table 8 reveals the iterative processes of four scenarios, which includes the number of iterations, the objective value f^*, the quasi-solutions/optimum solutions and the used time. Furthermore, the infeasible subsequences are listed for each quasi-solution. The symbol ● acts as a separator between the sequential and cyclic operations of the quasi-solution. Not only cycles, such as 5 → 7→ 5 in the second iteration of scenario 3, must be avoided, but other erroneous subsequences must also be inhibited, such as 7 → 5→ 3 in the 1st iteration of scenario 3. A cycle can be considered a special case of an erroneous subsequence. For IGA, *MaxIter* and *PopSize* are determined in term of the problem size, where two requirements are needed to be considered: (1) adequately converge to the optimum solution; and (2) use the CPU time as short as possible. Likewise, the proposed algorithm is executed 20 times each.

From Table 8, we can see that the optimum solution is found only after one iteration in the first SDDP scenario, and for scenarios

Table 8
Results of the iterative method by Lambert (2007) for four scenarios.

No. of scenarios	No. of iterations	f^*	Sequence	Infeasible subsequences	CPU time (s)	Accumulated CPU time (s)
1	1	4.78	0 → 1→ 12	No	0.11	0.11
2	1	14.9324	0 → 2→ 10 ● 4 → 15→ 23 → 27→ 29 → 30→ 4	4 → 15→ 23 → 27→ 29 → 30→ 4	2.68	7.99
	2	14.9324	0 → 2→ 10 → 4→ 15 ● 23 → 27→29 → 30→ 23	23 → 27→29 → 30→ 23	2.70	
	3	14.8824	0 → 2→ 10 → 30→ 4 → 15→ 23 → 27→ 29	No	2.61	
3	1	50.0	0 → 1→ 7 → 5→ 3 → 6	7 → 5→ 3	0.12	0.41
	2	48.5	0 → 1→ 2 → 3→ 6 ● 5 → 7→ 5	5 → 7→ 5	0.14	
	3	47.5	0 → 1→ 2 → 3→ 6 → 7→ 5	No	0.15	
4	1	57.0	0 → 1→ 2 → 6→ 10 → 4→ 9 → 3 ● 5 → 7→ 5	5 → 7→ 5	0.35	8.16
	2	57.0	0 → 1→ 2 → 7→ 5 ● 3 → 6→ 10 → 4 → 9 → 3	6 → 10→ 4 → 9→ 3	0.42	
	3	56.5	0 → 1→ 2 → 6→ 10 → 4→ 7 → 5→ 9 → 3	2 → 6→ 10 → 4, 10 → 4→ 7 → 5 and 7 → 5→ 9 → 3	0.32	
	4	56.5	0 → 1→ 2 → 7→ 5 → 9 ● 3 → 6→ 10 → 4→ 3	6 → 10→ 4 → 3	0.46	
	5	56.5	0 → 1→ 2 → 7→ 5 → 3→ 6 → 10→ 4 → 9	7 → 5 → 3	0.40	
	6	56.0	0 → 1→ 2 → 6→ 10 → 7→ 5 ● 4 → 9→ 3 → 4	10 → 7→ 5 and 9 → 3→ 4	0.42	
	7	56.0	0 → 1→ 2 → 7 ● 3 → 6→ 10 → 4 → 9 → 5→ 3	10 → 4 → 9 → 5	0.40	
	8	55.5	0 → 1→ 2 → 7→ 5 ● 3 → 6→ 9 → 3 ● 4 → 10→ 4	6 → 9→ 3 and 4 → 10→ 4	0.41	
	9	55.5	0 → 1→ 2 → 6→ 9 → 5→ 7 → 10→ 4 → 3	7 → 10→ 4 → 3	0.45	
	10	55.0	0 → 1→ 2 → 6→ 9 → 5→ 3 ● 4 → 7→ 10 → 4	6 → 9→ 5 → 3 and 4 → 7→ 10 → 4	0.48	
	11	55.0	0 → 1→ 2 → 6→ 10 → 7 ● 3 → 4→ 3 ● 5 → 9→ 5	3 → 4→ 3 and 5 → 9→ 5	0.45	
	12	55.0	0 → 1→ 2 → 6→ 10 → 7 ● 3 → 4→9 → 5→ 3	3 → 4→9 → 5→ 3	0.39	
	13	54.5	0 → 1→ 2 → 7→ 5 → 4→ 9 → 3→ 6 → 10	7 → 5 → 4 → 9→ 3	0.43	
	14	54.5	0 → 1→ 2 → 7→ 5 ● 3 → 6→9 → 10→ 4 → 3	9 → 10→ 4	0.48	
	15	54.5	0 → 1→ 2 → 7→ 5 ● 3 → 4→ 9 → 3 ● 6 → 10→ 6	6 → 10→ 6	0.42	
	16	54.0	0 → 1→ 2 → 7→ 5 → 10→ 4 → 9→ 3 → 6	7 → 5→ 10 → 4 → 9 → 3	0.49	
	17	54.0	0 → 1→ 2 → 7 ● 3 → 6→ 10 → 4→ 5 → 9→ 3	10 → 4→ 5	0.48	
	18	53.5	0 → 1→ 2 → 3→ 6 → 10→ 4 → 9→ 7 → 5	10 → 4→ 9 → 7→ 5	0.45	
	19	53.5	0 → 1→ 2 → 4→ 9 → 5→ 3 → 6→ 10 → 7	No	0.46	

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

Table 9
Results of IGA for four scenarios.

No. of scenarios	*MaxIter*	*PopSize*	f_{best}	f_a	Standard deviation of f	Average CPU time (s)	Standard deviation of CPU time
1	10	10	4.7800	4.7800	0.0000	0.0332	0.0034
2	50	50	14.8824	14.8824	0.0000	1.1871	0.0092
3	40	40	47.5000	47.2750	0.5495	0.6069	0.0153
4	100	50	53.5000	51.6000	0.6407	5.8624	0.0549

2 and 3, three iterations are proceeded to seek the solution. Especially for the product 4 of scenario 2, it is almost impossible to reveal the resultant solution within a reasonable CPU time through the default CPLEX method, see Table 7. However, the iterative approach enables to address this effectively, and a short CPU time (7.99 s) is consumed here. For the fourth scenarios, although there are 23 subassemblies and 11 operations as shown in Fig. A3, the iterative procedure is relatively complicated which consists of 19 iterations. In terms of its accumulated CPU time (8.16 s), its performance seems to be satisfactory. Nevertheless, it is stressed that all the calculations have been carried out with manual search for identifying infeasible subsequences. According to our experience, for the fourth scenario at least 10 s is used each identification sometimes more than 30 s which means that it may exceed 200 s in the whole iteration if and only if each identification has no human errors. In addition, the default CPLEX method unexpectedly converges to the exact optimal solution within 20 s, which seems to be more efficient due to the lack of manual search.

Fortunately, the proposed algorithm performs excellently with respect to the computation speed. As shown in Table 9, its average CPU times obviously outperform the CPU times of scenarios 1, 2 and 4 in Table 8. This appears different compared with others for scenario 3, while we have to keep in mind that the used time of manual search of erroneous subsequences is not considered in the iteration.

The performance of the objective value f is further compared between IGA and the iterative method. From Table 9, by observing f_{best} we can conclude that the proposed approach is able to successfully find the best solution similar to the exact optimum solution (see f^* in Table 8) in each scenario. Especially for scenarios 1 and 2, the optimal one is determined each trial of IGA, which is demonstrated in Fig. 6. Fig. 6 also demonstrates that IGA can seek the optimum of the third scenario in most cases. Relatively, it performs worst in scenario 4, and finds the optimal solution once, i.e., the fourth test. But as seen in Fig. 6, the rest results of scenario 4 is approaching the optimum, and Gap is less than 3.6% between f_a and f^*. Thus, the best solutions found by IGA could be accepted as near optimal solutions in this work. Meanwhile, the standard deviation values of f and CPU time are very small in Table 9, which indicates that the algorithm performance is stable.

Furthermore, we attempt to execute the iterative method for solving the original imaginary product with 31 operations and 63 subassemblies proposed by Lambert (2007). However, the first iteration is not completed after 12 h, which indicates that the CPU time is unmanageable for this case we investigated. Fortunately, the proposed approach can quickly generate the near optimal solution. In fact, the iterative method has an outstanding performance for solving some large-scale instances when there are moderate parallel operations (Lambert, 2007), which can be also demonstrated in Table 8. Specifically, the size of the second scenario is relatively large with 30 operations and 29 subassemblies, and only three iterations are implemented using a reasonable CPU time (7.99 s). But with respect to the fourth scenario, its calculation is relatively labored which focuses on the human identification for erroneous subsequences.

The convergence of the iteration process is satisfactory, and the

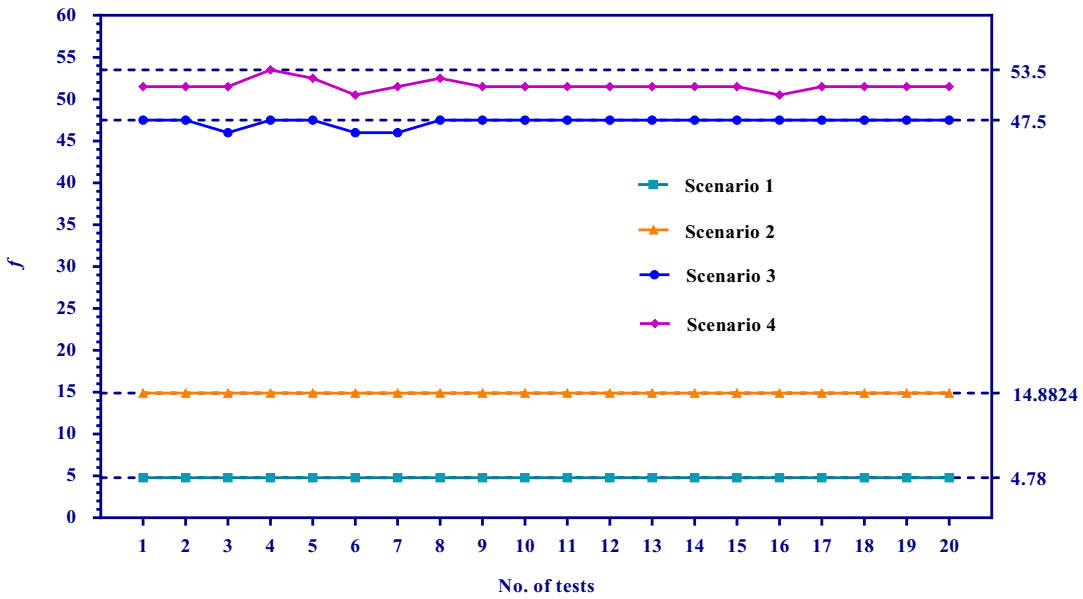


Fig. 6. f values of IGA for four scenarios.

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

Table 10
Results of IGA for solving the HG5-20 triaxial five speed mechanical transmission.

No. of scenarios	*MaxIter*	*PopSize*	f_{best}	f_a	Standard deviation of f	Average CPU time (s)
1	50	50	81.0151	78.0280	2.2888	1.2857
2	100	50	81.4410	78.8311	2.1146	2.4179
3	100	100	81.4410	80.0601	1.1302	5.2504
4	200	100	81.4528	80.6529	0.4537	10.1788

required CPU time appears relatively small as compared to the default CPLEX algorithm based on the proposal model. However, the manual search for infeasible subsequences not only easily leads to manual errors but also takes extra time that is related to the complexity of products. Further, the iterative procedure is actually an exact approach based on a binary integer programming, and the CPU time may be unmanageable for some complex scenarios since the CPU time tends to increase exponentially with the problem size. Fortunately, the proposed method is not limited to this case while enables to obtain the 'good enough' solutions, i.e., near optimum solutions. Therefore, for SDDP, the CPLEX method better suits for dealing with small sized instances, and the iterative approach prefers the medium/large-scale ones with modest parallel operations, and IGA are applicable to a wide range of cases.

4.4. Case study

This subsection employs our proposed method to address a large-scale real instance, i.e., a HG5-20 triaxial five speed mechanical transmission. As shown in Fig. 6A of **Appendix**, the transmission consists of 12 main parts, and its AND/OR graph with 39 operations and 34 subassemblies is given in Fig. 7A. Let $P_c = 0.8$ and $P_m = 0.2$, and *MaxIter* and *PopSize* are tuned here. Each parameter combination is tested 20 runs using IGA and Table 10 presents the numerical results. From the table, it can be observed that f_{best}, f_a, and the standard deviation are better with the increase of *MaxIter* and *PopSize*. Especially when *MaxIter* = 200 and *PopSize* = 100, IGA appears highly stable with less than 0.5 standard deviation. On the other hand, the computational cost is very low in each scenario, e.g., the biggest CPU time is approximately 10 s as seen in the last column of Table 10. Further, the best solution found by IGA is compared with the traditional solutions determined by the operators' experience. We find that our solution can shorten approximately 11% disassembly time while the recovery profit is almost increased by 7%. This demonstrates that the proposed approach can efficiently improve the disassembly solution in practice.

5. Conclusions

The development of sustainable manufacturing and circular economy has led to increasing attention on remanufacturing. Compared with extracting virgin materials for manufacturing products, remanufactured products represent a more environmentally preferred approach as it conserves the energy and materials consumed during component manufacturing. Disassembly planning is the first step of remanufacturing industries whose result has a significant impact on the economic benefits of sustainable manufacturing. Current research investigates a sequence-dependent disassembly planning optimization issue that aims at finding the order of disassembling subassemblies in an EOL product with the maximal recovery profit. A graphical representation based on AOG is adopted in this work and a disassembly model is developed using AOG. In the AOG, precedence and exclusive OR matrices are simultaneously formed to describe precedence and EOR relations among components or operations, and the proposed model is formulated using the two matrices. The model can not only depict the SDDP, but be directly applied to the mathematical programming. It indicates that an exact method enables to be carried out for this problem, which is of significance to demonstrate the effectiveness of the proposed metaheuristics as well as proposed mathematical model. In addition, a two-phase heuristic is proposed and utilized to quickly initialize feasible solutions, which successfully deals with the complex constraints from the AOG. Finally, an efficient metaheuristic based on GA called IGA is presented to solve the SDDP.

The performance of the proposed IGA is measured through a number of experiments against the performance of two famous exact methods. The comparison results indicate that the proposed IGA outperforms the exact methods considered in terms of the computational efficiency, while the satisfactory solutions are effectively yielded for different sizes of problems. Finally, a real-world case is applied to testify our approach, and the numerical results demonstrate that IGA can efficiently compute the real instance while the solutions obtained by IGA are superior to the existing solutions determined by the operators' experience. However, the robustness of our solutions has not been tested, which is associated with the uncertainty of the EOL products. Specifically, the revenues of subassemblies in EOL products are highly related to their qualities when reprocessing or recycling them, whereas the subassembly quality cannot be certain for decision makers (operators). Furthermore, the quality status of each subassembly might change during the disassembly process under different disassembly resources, e.g., the processing workshop and disassembly tools. To provide a more practical method for selective disassembly planning, the uncertainty of the quality of EOL products could be studied in the future.

Acknowledgements

This research is supported by the Funds for the Key Research and Development Program of Guangdong Province (Grant No. 2019B090921001), Guangzhou Leading Innovation Team Project (Grant No. 201909010006), National Natural Science Foundation of China (Grant No. 51575211, 51875251), and the U.S. National Science Foundation (Grant No. 1512217). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Natural Science Foundation of China and the U.S. National Science Foundation. We are grateful to the editor and anonymous referees for their constructive comments to improve this paper.

Appendix

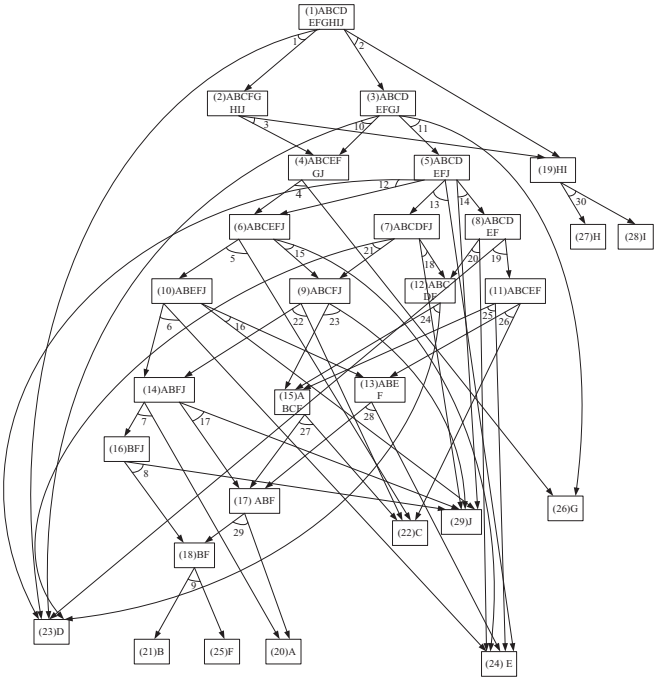


Fig. A1. The AND/OR graph of the radio set (product 4) in scenario 2.

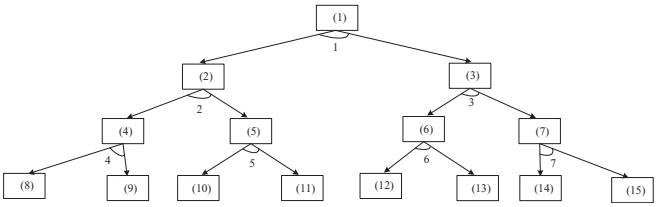


Fig. A2. The AND/OR graph of the product in scenario 3.

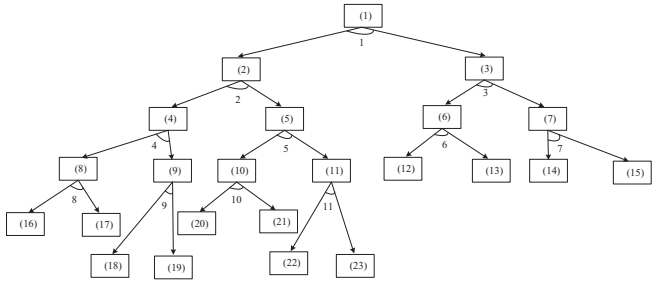


Fig. A3. The AND/OR graph of the product in scenario 4.

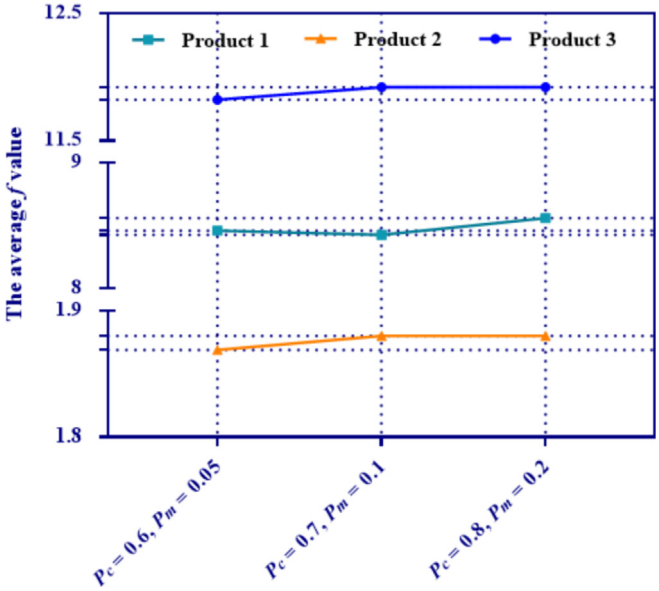


Fig. A4. f performance comparison with different P_c and P_m for three products.

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

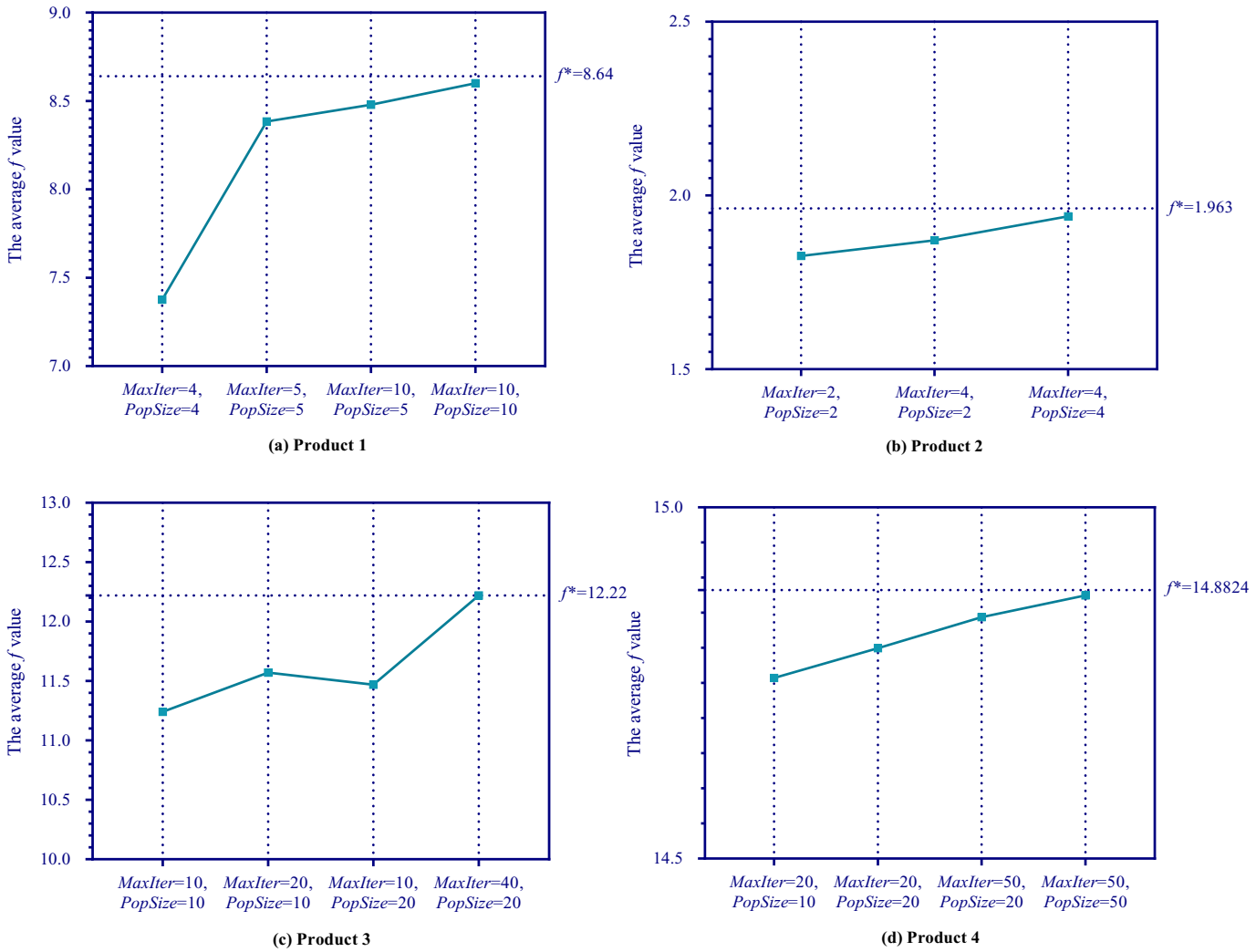


Fig. 5A. The average *f* values of IGA for products 1, 2, 3 and 4 with different *MaxIter* and *PopSize*.

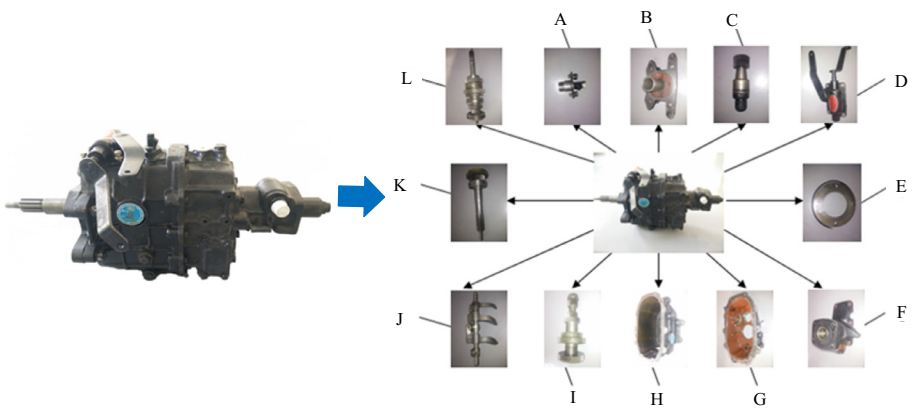


Fig. 6A. The HG5-20 triaxial five speed mechanical transmission and its main parts.

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

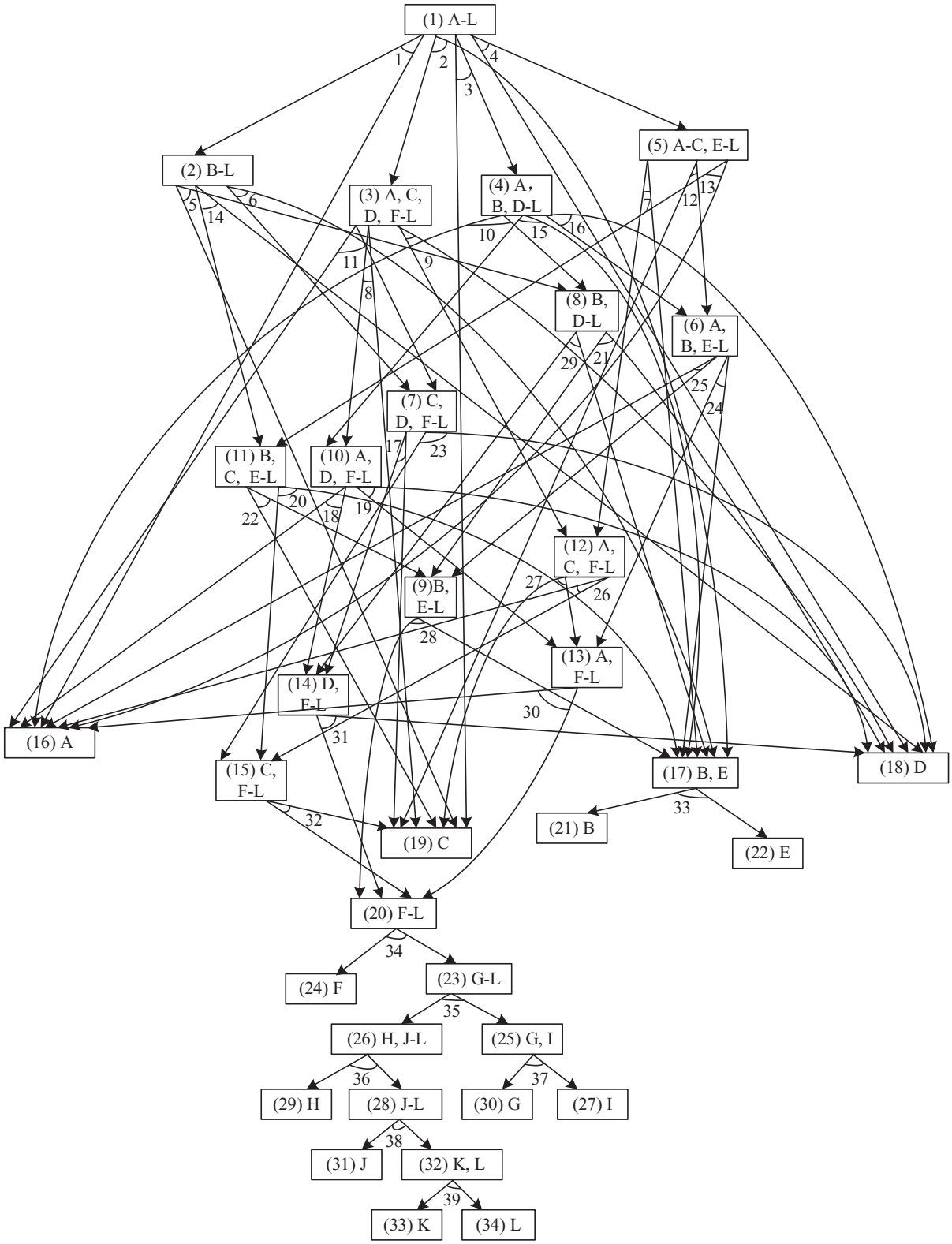


Fig. 7A. The AND/OR graph of the HG5-20 triaxial five speed mechanical transmission. Symbol '-' is used to simplify the composition of subassemblies in the AND/OR graph. For example, A-L denotes the alphabetical order from A to L, i.e., A, B, C, D, E, F, G, H, I, J, K, and L.

References

Behdad, S., Kwak, M., Kim, H., Thurston, D., 2010. Simultaneous selective disassembly and end-of-life decision making for multiple products that share disassembly operations. J. Mech. Des. 132, 041002.

Colledani, M., Battaïa, O., 2016. A decision support system to manage the quality of End-of-Life products in disassembly systems. CIRP Ann. - Manuf. Technol. 65, 41–44.

De Mello, L.H., Sanderson, A.C., 1990. AND/OR graph representation of assembly

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644

plans. IEEE Trans. Robot. Autom. 6, 188–199.
Edmunds, R., Kobayashi, M., Higashi, M., 2011. Generating optimal disassembly process plans from AND/OR relationships using a hierarchical genetic algorithm. In: Proceedings of the 21st CIRP Design Conference, pp. 16–23.
Guo, X., Liu, S., Zhou, M., Tian, G., 2016. Disassembly sequence optimization for large-scale products with multiresource constraints using scatter search and Petri nets. IEEE Trans. Cybern. 46, 2435–2446.
Guo, X., Liu, S., Zhou, M., Tian, G., 2018. Dual-objective program and scatter search for the optimization of disassembly sequences subject to multiresource constraints. IEEE Trans. Autom. Sci. Eng. 15, 1091–1103.
Harivardhini, S., Krishna, K.M., Chakrabarti, A., 2017. An Integrated Framework for supporting decision making during early design stages on end-of-life disassembly. J. Clean. Prod. 168, 558–574.
Johnson, M., Wang, M.H., 1998. Economical evaluation of disassembly operations for recycling, remanufacturing and reuse. Int. J. Prod. Res. 36, 3227–3252.
Kang, J.G., Lee, D.H., Xirouchakis, P., Persson, J.G., 2001. Parallel disassembly sequencing with sequence-dependent operation times. CIRP Ann. - Manuf. Technol. 50, 343–346.
Kheder, M., Trigui, M., Aifaoui, N., 2015. Disassembly sequence planning based on a genetic algorithm. Proc. Inst. Mech. Eng. C J. Mech. Eng. Sci. 229, 2281–2290.
Koc, A., Sabuncuoglu, I., Erel, E., 2009. Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph. IIE Trans. 41, 866–881.
Lambert, A., 2006. Exact methods in optimum disassembly sequence search for problems subject to sequence dependent costs. Omega 34, 538–549.
Lambert, A., 2007. Optimizing disassembly processes subjected to sequence-dependent cost. Comput. Oper. Res. 34, 536–551.
Lambert, A., Gupta, S.M., 2008. Methods for optimum and near optimum disassembly sequencing. Int. J. Prod. Res. 46, 2845–2865.
Lambert, A.J.D., 1999. Linear programming in disassembly/clustering sequence generation. Comput. Ind. Eng. 36, 723–738.
Lu, Q., Ren, Y., Jin, H., Meng, L., Li, L., Zhang, C., Sutherland, J.W., 2020. A hybrid metaheuristic algorithm for a profit-oriented and energy-efficient disassembly sequencing problem. Robot. Comput. Integr. Manuf. 61, 101828.
Meng, L., Zhang, C., Shao, X., Ren, Y., 2019a. MILP models for energy-aware flexible job shop scheduling problem. J. Clean. Prod. 210, 710–723.
Meng, L., Zhang, C., Shao, X., Ren, Y., Ren, C., 2019b. Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. Int. J. Prod. Res. 57 (4), 1119–1145.
Ondemir, O., Gupta, S.M., 2014. A multi-criteria decision making model for advanced repair-to-order and disassembly-to-order system. Eur. J. Oper. Res. 233, 408–419.
Percoco, G., Diella, M., 2013. Preliminary evaluation of artificial bee colony algorithm when applied to multi objective partial disassembly planning. Res. J. Appl. Sci. Eng. Technol. 6, 3234–3243.
Ren, Y., Tian, G., Zhao, F., Yu, D., Zhang, C., 2017a. Selective cooperative disassembly planning based on multi-objective discrete artificial bee colony algorithm. Eng. Appl. Artif. Intell. 64, 415–431.
Ren, Y., Yu, D., Zhang, C., Tian, G., Meng, L., Zhou, X., 2017b. An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem. Int. J. Prod. Res. 55 (24), 7302–7316.
Ren, Y., Zhang, C., Zhao, F., Tian, G., Lin, W., Meng, L., Li, H., 2018a. Disassembly line balancing problem using interdependent weights-based multi-criteria decision making and 2-Optimal algorithm. J. Clean. Prod. 174, 1475–1486.
Ren, Y., Zhang, C., Zhao, F., Xiao, H., Tian, G., 2018b. An asynchronous parallel disassembly planning based on genetic algorithm. Eur. J. Oper. Res. 269, 647–660.
Ren, Y., Zhang, C., Zhao, F., Triebe, M.J., Meng, L., 2018c. An MCDM-based multi-objective general variable neighborhood search approach for disassembly line balancing problem. IEEE Trans. Syst. Man Cybern.: Systems (99), 1–14.
Sanchez, B., Haas, C., 2018. A novel selective disassembly sequence planning method for adaptive reuse of buildings. J. Clean. Prod. 183, 998–1010.
Smith, S., Hsu, L.-Y., Smith, G.C., 2016. Partial disassembly sequence planning based on cost-benefit analysis. J. Clean. Prod. 139, 729–739.
Smith, S., Hung, P.-Y., 2015. A novel selective parallel disassembly planning method for green design. J. Eng. Des. 26, 283–301.
Tseng, H.-E., Chang, C.-C., Lee, S.-C., Huang, Y.-M., 2018. A Block-based genetic algorithm for disassembly sequence planning. Expert Syst. Appl. 96, 492–505.

Please cite this article as: Ren, Y et al., An efficient metaheuristics for a sequence-dependent disassembly planning, Journal of Cleaner Production, https://doi.org/10.1016/j.jclepro.2019.118644