Heuristic solution methods for the selective disassembly sequencing problem under sequence-dependent costs

Jully Jeunet, Federico Della Croce, Fabio Salassa

▶ **To cite this version:**

Jully Jeunet, Federico Della Croce, Fabio Salassa. Heuristic solution methods for the selective disassembly sequencing problem under sequence-dependent costs. Computers and Operations Research, 2021, 127, pp.105151. 10.1016/j.cor.2020.105151 . hal-03097669

HAL Id: hal-03097669
https://hal.science/hal-03097669

Submitted on 15 Dec 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Version of Record: https://www.sciencedirect.com/science/article/pii/S0305054820302689
Manuscript_8a6229ef0658173874971c6910f8a284

Heuristic solution methods for the selective disassembly sequencing problem under sequence-dependent costs

Jully Jeunet[1], Federico Della Croce[2,3], Fabio Salassa[2]

[1]CNRS, Université Paris Dauphine, PSL Research University, CNRS UMR[7243], Lamsade, place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16. Corresponding Author. jully.jeunet@dauphine.fr

[2]Dipartimento di Ingegneria Gestionale e della Produzione (DIGEP), Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino 10129, Italy

[3]CNR, IEIIT, Torino, Italy

Abstract

The first Waste Framework Directive issued by the European Union dates back to the seventies but was drastically amended in the last decade to reduce environmental impacts of waste by encouraging reuse, recycling and remanufacturing. Product recovery starts with disassembly which results in high labor costs. Disassembly supports environmentally conscious choices like replacement of defective parts to extend the life span of products, removal of suitable components for reuse and extraction of hazardous substances to decontaminate materials for reprocessing. Besides, selective disassembly also accommodates maintenance and repairs. Optimizing the cost of disassembly is crucial to make this process an economically viable option. Due to change tools and parts reorientation, disassembly costs are sequence-dependent. Therefore minimizing the disassembly cost involves the search for an adequate sequence of disassembly tasks. Consequently, this paper addresses the disassembly sequencing problem for selective and sequential disassembly under sequence-dependent costs. As optimal formulations fail to handle real-world cases, we develop a randomized greedy algorithm (needing a very few number of parameters to be set and proving to be robust with respect to their value) and a matheuristic to solve efficiently medium to large-sized instances.

Keywords: Disassembly; Product recovery; Maintenance; Greedy algorithms; Matheuristics

1 Introduction

Disassembly encompasses a wide variety of issues that mostly depend upon the decision level. On a strategic level, design for disassembly such as increasing modularity is meant for easier reuse of components and materials (see for instance Chiodo and Ijomah, 2014 or Soh et al., 2014). Disassembly for remanufacturing using strategic perspectives has been studied in Tian et al. (2017) and Priyono et al. (2015). Also the problem has been studied in multi station environments with line balancing requirements (Li et al., 2019) and when buffer allocation among stages is considered (Alfieri et al., 2020).

In disassembly planning problems, some recent works focus on joint optimization of collecting End-Of-Life products and disassembling them (Habibi et al., 2017) or aim at reducing unnecessary inventories generated by products disassembly (Godichaud and Amodeo, 2018).

In these planning models, the disassembly cost of products is a parameter since the disassembly sequencing problem is assumed to be already solved. Thus, adjacent and central to disassembly planning is disassembly sequencing where the main objective is to find a sequence of disassembly tasks so as to detach parts of interest (disassembly can be either selective or complete) while minimizing the dismantling cost or time, subject to precedence constraints

© 2020 published by Elsevier. This manuscript is made available under the CC BY NC user license
https://creativecommons.org/licenses/by-nc/4.0/

between components. When disassembly is performed manually as is the case in this paper, the disassembly cost is equal to the disassembly time multiplied by the labour cost. Thus as the cost is proportional to the time, these terms can interchangeably be used. The disassembly time of each disassembly operation depends on the sequence, due to tool selection and part reorientation. That is, the disassembly time of operation j, if performed after disassembling part i, not only includes the time to dismantle j but also the time to change tool if different from the one needed for i as well as the time to move the product so as to access the fasteners of j. With sequence-dependent costs (or time), the sequential disassembly sequencing problem (DSP) is known to be a NP-hard problem (Lambert and Gupta, 2008) and as such, exact methods have limited real-world applicability.

In this paper, we address the DSP with the aim of obtaining target components from a product for the purpose of maintenance, recycling or remanufacturing, where parts are sequentially and manually disassembled and disassembly times are deterministic and sequence-dependent. For this problem, an optimal solution method has been developed by Han et al. (2013) that can be applied to DSP with up to 50 parts. For large scale problems, these authors have proposed simple priority rules, therefore leaving room for the development of more efficient heuristic solution methods. We thus propose a Randomized Greedy Sequencing Algorithm and a Matheuristic approach. Our greedy algorithm generates feasible sequences by filling one by one the empty positions with components probabilistically selected according to their disassembly time. Random repetitions are performed and insertion of disassembly operations is considered for further improvements. Our greedy approach only needs a very few number of parameters to be set and proved to be robust with respect to their values. Matheuristics are based on a decomposition of the problem into sub-problems of limited size for which an optimal solution can be quickly obtained. So far, they had not yet been tested on the DSP, despite their ability to solve efficiently many combinatorial problems in production and operations management (e.g. Iassinovskaia et al., 2017; Della Croce et al., 2014(a) and 2014(b); Raa et al., 2013; Guerrero et al., 2013). As Matheuristics make use of mathematical programming models, they can provide near optimal solutions only to moderate-sized problems. The advantage of our Matheuristic for the DSP is its ability to find better solutions than the exact approach of Han et al. (2013) that often failed to find the optimal solution to medium-sized problems in a reasonable computation time. For large-sized problems with hundreds components to disassemble, Han et al. (2013) showed the superiority of the Nearest Neighbor algorithm over other simple priority rules. However, when compared with our Greedy algorithm and our Matheuristic, the Nearest Neighbor algorithm performed so poorly that we suggested an improved randomized version of this algorithm. We used a rigorous simulation framework to generate realistic instances for which we showed the superiority of our methods over that of Han et al. (2013). For many instances with product structures with 50 components, the exact approach of Han et al. (2013) most often failed to find the optimal solution, in which cases our heuristic solution methods beat by far the solution of the optimal method obtained after one hour of CPU. For large instances with several hundreds of parts, the Nearest Neighbor algorithm which was assessed as the best heuristic in Han et al. (2013) showed deviations to our Randomized Greedy Search Algorithm from 30 to 75% whereas these deviations did not exceed 10% on average for our Randomized version of the Nearest Neighbor algorithm.

In real-world problems, disassembly is generally selective because (*i*) remanufacturing processes require retrieval of cores only over non-remanufacturable parts; (*ii*) maintenance is implemented on specific usury parts; (*iii*) not all materials can be recycled or can only be recycled a limited number of times. The most frequent situation is that of a single worker sequentially dismantling a product to obtain one part at a time. Therefore, parallel execution of disassembly operations to remove several parts simultaneously has been seldom studied in the literature (see e.g. Ren et al., 2018; Zhang et al. 2014 or Edmunds et al., 2012). Besides, disassembly remains a labor intensive process because automation demands high investment costs and implies a loss of flexibility in accessing fasteners. Actually, human intervention will always be needed to choose proper tools for removing fasteners when damaged by corrosion for instance (Chang et al., 2017). As the labor cost is proportional to the disassembly time, minimizing the cost is equivalent to minimizing the total disassembly time (see Gonzalez and Adenso-Dìaz, 2006 or Wang and Johnson, 1995). In addition to time or cost minimization, other objectives are sometimes considered like removing components according to a priority list (Adenso-Dìaz et al., 2008) or maximizing the profit gained from recycling (Ren et al., 2018) or maximizing this profit and minimizing the energy consumption related to disassembly (Tian et al., 2019). More recently, the quality of disassembled components has also been included (Bentaha et al., 2020; Tian et al., 2018).

As previously mentioned, the cost of a disassembly operation is sequence-dependent and is usually assumed

to be deterministic in sequential disassembly settings. Several papers consider this dependence through geometric aspects like rotation or more generally disassembly directions to access fasteners (Tseng et al., 2018; Yeh et al., 2012; Go et al., 2012; Smith et al. 2012; Gonzalez and Adenso-Dìaz, 2006; Güngör and Gupta, 1997 and 2001). Solution methods developed in these papers are mainly metaheuristics. Yeh et al. (2012) developed a simplified swarm optimization with costs integrating learning effects; Go et al. (2012) adopted a Genetic Algorithm and Gonzalez and Adenso-Dìaz (2006) proposed a scatter search algorithm.

Other papers are based on disassembly cost or time estimates between each feasible pairs of operations (Luo et al., 2016; Han et al., 2013; Lambert and Gupta, 2008; Lambert, 2006 and 2007; Huang et al., 2000; Johnson and Wang, 1998). Luo et al. (2016) used an ant colony approach to solve small instances of the DSP, whereas Han et al. (2013) developed an optimal formulation similar to the modified two-commodity network flow model of Lambert (2006).

Besides, in parallel disassembly environment, some contributions deal with random disassembly costs or times, developing heuristic and exact approaches (Kim and Lee, 2018; Kim et al., 2018; Kim et al., 2017; Tian et al., 2012a, 2012b, 2013).

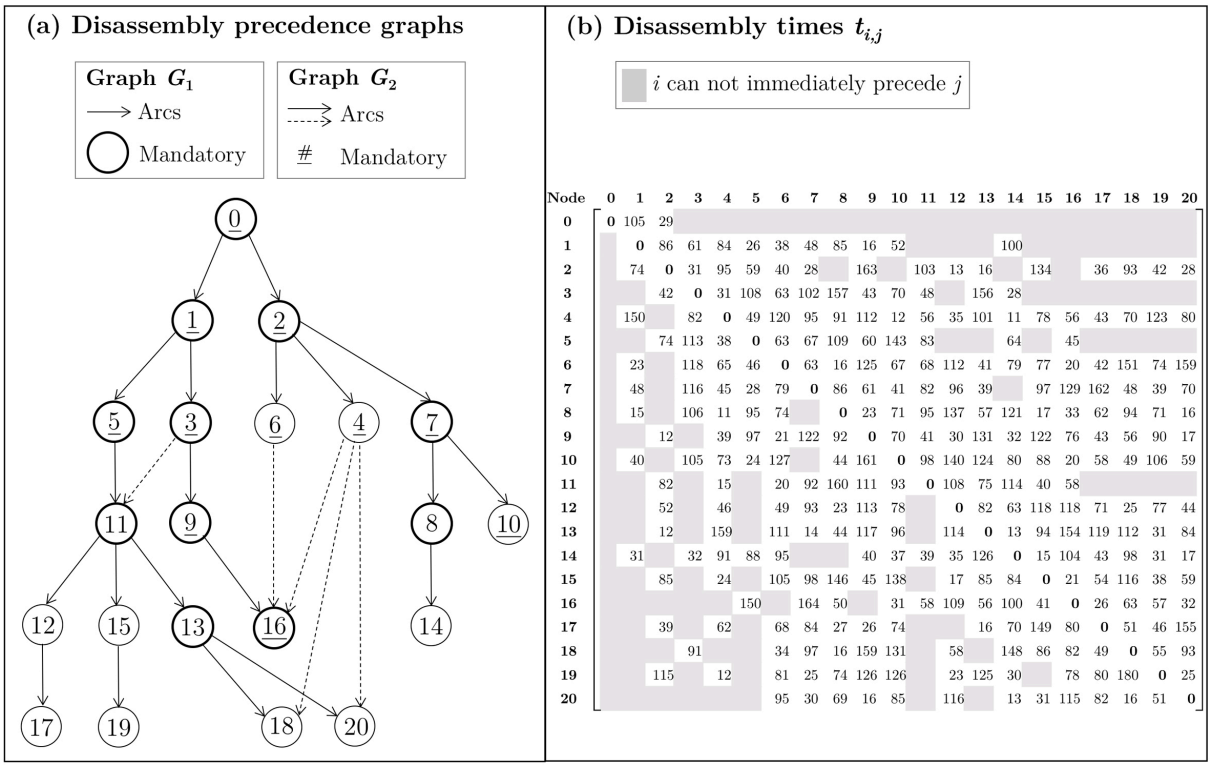
The remainder of the paper is organized as follows. The next section provides the problem description and the representation of precedence constraints. Section 3 gives the optimal formulation of the DSP as proposed by Han et al. (2013). The randomized greedy sequencing algorithm is detailed in Section 4 and Section 5 describes the matheuristic algorithm. The simulation framework and results analysis are presented in Section 6. Section 7 summarizes our findings and suggests directions for further research.

2 Problem description

As in Lambert (2006), we adopt a disassembly precedence graph, G, to represent disassembly operations and their precedence relationships. Such a graph is a set of $n+1$ nodes where root node 0 corresponds to the start of disassembly and n is the total number of parts that can be obtained through disassembly operations whose precedence relationships are symbolized by arcs.

Figure 1(a) displays two graphs G_1 and G_2 each representing a product with $n = 20$ components. Graph G_1, including only arcs in solid lines, exhibits a complete divergent structure where disassembling one component never requires more than one direct predecessor to be dismantled before, immediately or not in the sequence. In addition to arcs in solid lines, G_2 involves dotted arcs and thus has a general structure where detaching one component can need several operations to be performed before. Letting $\Gamma^{-1}(j)$ be the set of direct predecessors of node j, in a complete divergent structure we have $\left|\Gamma^{-1}(j)\right| = 1$, $\forall j = 1..n$ whereas $\left|\Gamma^{-1}(j)\right| \geq 1$ in a general structure. For instance $\Gamma^{-1}(16) = \{9\}$ in G_1 and $\Gamma^{-1}(16) = \{4, 6, 9\}$ in G_2.

We introduce the notion of mandatory components to designate not only target components but also all the components that have to be disassembled before these targets to meet the precedence constraints. The set $\mathcal{M}$ of mandatory components is simply defined as $\mathcal{M} = \mathcal{T} \cup_{j \in \mathcal{T}} \hat{\Gamma}^{-1}(j)$, where $\mathcal{T}$ is the set of targets and $\hat{\Gamma}^{-1}(j)$ is the set of ancestors of j (direct and non direct predecessors) in the graph. Ancestor matrix $\hat{\Gamma}^{-1}$ is obtained from predecessor matrix Γ^{-1} by setting first $\hat{\Gamma}^{-1} := \Gamma^{-1}$ and then $\hat{\Gamma}^{-1} := \hat{\Gamma}^{-1} \vee \left(\Gamma^{-1}\right)^i$, $i = 1..n$. Set $\mathcal{M}$ is a straightforward feasible sequence to the disassembly problem and also provides the minimum disassembly rate of a product, $(|\mathcal{M}| - 1)/n$, which gives a better picture of the complexity of the problem than the sole number of target components does. Indeed, depending upon its location in the graph and on the structure of it, one target component may require from one to up to all components to be disassembled. In G_1 or G_2, if the target component is 5, the disassembly problem is quite easy to solve whereas candidate sequences are far more numerous if component 18 is a target. In Figure 1(a), bold circles represent mandatory parts in G_1 with $\mathcal{M}_1 = \{0, 1, 2, 3, 5, 7, 8, 9, 11, 13, 16\}$ and underlined numbers are mandatory components numbers in G_2 with $\mathcal{M}_2 = \{0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 16\}$ leading to a minimum disassembly rate of 50% and a minimum of three targets in both cases ($\mathcal{T}_1 = \{8, 13, 16\}$ in G_1 and $\mathcal{T}_2 = \{5, 10, 16\}$ in G_2). The recycle rate thus varies from 3/20 to 0.5 and more if all components in the optimal sequence can be recycled. As disassembly times are sequence-dependent, it can be optimal to disassemble one or more non mandatory parts between two mandatory ones. A matrix of disassembly times $t_{i,j}$ is provided in Figure 1(b), where $t_{i,j}$ is the disassembly time in seconds when i immediately precedes j in the sequence. With these



Node	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	**0**	105	29																		
1		**0**	86	61	84	26	38	48	85	16	52				100						
2		74	**0**	31	95	59	40	28		163		103	13	16		134		36	93	42	28
3			42	**0**	31	108	63	102	157	43	70	48		156	28						
4		150		82	**0**	49	120	95	91	112	12	56	35	101	11	78	56	43	70	123	80
5			74	113	38	**0**	63	67	109	60	143	83			64		45				
6		23		118	65	46	**0**	63	16	125	67	68	112	41	79	77	20	42	151	74	159
7		48		116	45	28	79	**0**	86	61	41	82	96	39		97	129	162	48	39	70
8		15		106	11	95	74		**0**	23	71	95	137	57	121	17	33	62	94	71	16
9			12		39	97	21	122	92	**0**	70	41	30	131	32	122	76	43	56	90	17
10		40		105	73	24	127		44	161	**0**	98	140	124	80	88	20	58	49	106	59
11			82		15		20	92	160	111	93	**0**	108	75	114	40	58				
12			52		46		49	93	23	113	78		**0**	82	63	118	118	71	25	77	44
13			12		159		111	14	44	117	96		114	**0**	13	94	154	119	112	31	84
14		31		32	91	88	95			40	37	39	35	126	**0**	15	104	43	98	31	17
15			85		24		105	98	146	45	138		17	85	84	**0**	21	54	116	38	59
16						150		164	50		31	58	109	56	100	41	**0**	26	63	57	32
17			39		62		68	84	27	26	74			16	70	149	80	**0**	51	46	155
18				91			34	97	16	159	131		58		148	86	82	49	**0**	55	93
19			115		12		81	25	74	126	126		23	125	30		78	80	180	**0**	25
20							95	30	69	16	85		116		13	31	115	82	16	51	**0**

Figure 1: Disassembly precedence graphs and disassembly times

times, optimal sequences for G_1 and G_2 both include non mandatory component 6. They are obtained using the formulation of the disassembly sequencing problem we present in the next section.

3 Mathematical programming formulation of the disassembly sequencing problem

We adopt the formulation of Han et al. (2013) for it proved to be at least as efficient on several instances as two other formulations of lower complexity that we developed and in which variables express the position of components in the disassembly sequence, with either a linear or a quadratic objective function. These alternate formulations are given in Appendix A. Han et al. (2013) make use of the variables defined in Table 1 in which we also summarize the notations for the parameters.

Variables	
$\boldsymbol{x_{i,j}}$	Binary variable equal to one if i immediately precedes j in the sequence, and zero otherwise
$\boldsymbol{y_j}$	Binary variable equal to one if j is disassembled, and zero otherwise
$\boldsymbol{z_{i,j}}$	Binary variable equal to one if i must be disassembled before j in the sequence and zero otherwise
Parameters	
$\boldsymbol{\Gamma(j)}$	Set of direct successors of j in the precedence disassembly graph
$\boldsymbol{\Gamma^{-1}(j)}$	Set of direct predecessors of j in the precedence graph
$\boldsymbol{\hat{\Gamma}^{-1}(j)}$	Set of ancestors of j in the precedence graph (direct and non direct predecessors)
$\boldsymbol{\mathcal{T}}$	Set of target components
$\boldsymbol{\mathcal{M}}$	Set of all mandatory components to be disassembled which includes set $\mathcal{T}$ and all their ancestors in the graph. We have $\mathcal{M} = \mathcal{T} \cup_{j \in \mathcal{T}} \hat{\Gamma}^{-1}(j)$
$\boldsymbol{\bar{\mathcal{M}}}$	Set of non mandatory components, $\bar{\mathcal{M}} = \{0..n\} \setminus \mathcal{M}$
$\boldsymbol{t_{i,j}}$	Disassembly time (in sec.) of operation j when i immediately precedes j in the sequence

Table 1: Notations

The objective is to minimize the total disassembly time

$$\min \sum_{i=0..n} \sum_{j=0..n} t_{i,j} \cdot x_{i,j}, \tag{1}$$

subject to the following constraints.

Input and output flow conservations are written as

$$\sum_{i=0..n} x_{i,j} = y_j, \quad \forall j = 1..n. \tag{2}$$

$$\sum_{j=0..n} x_{i,j} \leq y_i, \quad \forall i = 0..n, \quad j = 0..n. \tag{3}$$

Mandatory components must be disassembled

$$y_j = 1, \quad \forall j \in \mathcal{M}. \tag{4}$$

If i is a direct predecessor of j in the graph and if j is disassembled ($y_j = 1$) then i must be disassembled as well ($y_i = 1$)

$$y_j \leq y_i, \quad \forall j = 1..n, \quad i \in \Gamma^{-1}(j). \tag{5}$$

If i is a direct predecessor of j in the graph then j cannot be disassembled before i in the sequence

$$z_{j,i} = 0, \quad \forall j = 1..n, \quad i \in \Gamma^{-1}(j). \tag{6}$$

If j is disassembled before i, the opposite is false and conversely

$$z_{j,i} + z_{i,j} = 1, \quad \forall i = 1..n, \quad j = 1..n, \quad i \neq j. \tag{7}$$

Constraints (7) are redundant with (6) but increase the tightness of the formulation.

If, in the sequence, component i is disassembled before j and j before k thus i is disassembled before k

$$z_{i,j} + z_{j,k} - z_{i,k} \leq 1, \quad \forall i = 0..n, \quad j = 1..n, \quad k = 1..n, \quad i \neq j, \quad k \neq j. \tag{8}$$

If i is not disassembled before j ($z_{i,j} = 0$) thus i cannot be disassembled immediately before j ($x_{i,j} = 0$)

$$z_{i,j} - x_{i,j} \geq 0, \quad \forall i = 0..n, \quad j = 0..n. \tag{9}$$

Let us note that constraints (9) allow for $z_{i,j} = 1$ and $x_{i,j} = 0$ meaning that $z_{i,j} = 1$ if i must precede j in the sequence even if j is not sequenced right after i ($x_{i,j} = 0$).

Finally a disassembly operation cannot be performed on one component itself

$$x_{i,i} = 0, \quad \forall i = 0..n. \tag{10}$$

4 Randomized Greedy Sequencing Algorithm (RGSA)

At the core of our RGSA is a sequencing procedure that generates a feasible sequence by filling one by one the empty positions with mandatory components probabilistically selected according to their best cumulative disassembly time. The cumulative time of any component is defined as its disassembly time from its immediate predecessor in the sequence, to which is added that of its quickest successor operation. A pass of the sequencing procedure consists in repeating this procedure on a same set of mandatory components until no improvement of the sequence is found during the last N repetitions. Over the whole repetitions, we record the maximum time loss associated with each non mandatory operation and the position in the sequence at which this maximum occurs. In this way we get a proxy of opportunity time losses as a result of the exclusion of non mandatory operations from the sequence. Based on these losses, non mandatory components are then considered one by one for possible insertion in the sequence in subsequent passes. To describe our RGSA we adopt the notations and definitions in Table 2. Obviously, root node 0 is sequenced in the first position (Pos = 1) of the sequence. The pseudocode of the whole algorithm is given in Table 3 where Table 3(a) provides the main steps of the RGSA and Table 3(b) details the sequencing procedure.

To illustrate, let us consider the example in Figure 1. With graph G_1, **RGSA Initialization** consists in setting $S^{\text{best}} := \mathcal{M}_1 = \{0, 1, 2, 3, 5, 7, 8, 9, 11, 13, 16\}$ which is the straightforward feasible sequence that we provided in Section 2. The corresponding disassembly time is $\text{Obj}(S^{\text{best}}) = t_{0,1} + \cdots + t_{13,16} = 776$. A first application of the **Sequencing procedure** is implemented on set $\mathcal{M}$ without insertion ($i^* = -1$). In position Pos = 1, we have $S_1 = \{0\}$, $s = 0$.

In position Pos = 2, the set K of candidates is $K = \Gamma(0) = \{1, 2\}$, with $M = \{1, 2\}$ since both operations are mandatory. In sub-procedure **Select mandatory compo** we first compute t_m for $m \in \{1, 2\}$. Using Eq. (11) and Eq. (13) for $m = 1$, we get $\Gamma^F(1) = \Gamma(1) = \{3, 5\}$ and $F_1 = K \setminus \{1\} \cup \Gamma^F(1) = \{2, 3, 5\}$, meaning that from operation 1, component 2 or 3 or 5 can be disassembled. From Eq. (14), we obtain the cumulative disassembly time of operation 1, $t_1 = t_{0,1} + \min\{t_{1,2}, t_{1,3}, t_{1,5}\} = 131$. This time t_1 reflects the minimum time we could obtain if operation 1 was performed between 0 and its quickest following operation, namely 5. Applying the same reasoning on candidate $m = 2$, we get $t_2 = t_{0,2} + \min\{t_{2,1}, t_{2,4}, t_{2,6}, t_{2,7}\} = 57$.

In each position Pos $\geq$ 2 to be filled in the sequence we have

$\boldsymbol{S_{\text{Pos}-1}}$	Set of previously sequenced operations, with $S_{\text{Pos}-1} = \{s_1, s_2, \ldots, s_{\text{Pos}-1}\}$ and $s_1 = 0$
$\boldsymbol{s}$	Last sequenced component in $S_{\text{Pos}-1}$ ($s \equiv s_{\text{Pos}-1}$)
$\boldsymbol{\Gamma^F(k)}$	Set of immediate successors of component k *in the graph* that can be dismantled if k is disassembled. These feasible successors are such that all their direct predecessors in the graph (except k) are already detached. We have $$\Gamma^F(k) = \left\{j \in \Gamma(k) \mid \Gamma^{-1}(j) \setminus \{k\} \subset S_{\text{Pos}-1}\right\} \quad (11)$$ In a complete divergent graph, we always have $\Gamma^F(k) = \Gamma(k)$.
$\boldsymbol{K}$	Set of all candidate components that can be detached in position Pos with $$K := K \setminus \{s\} \cup \Gamma^F(s) \quad (12)$$ In position 2, we have $K := \{0\} \setminus \{0\} \cup \Gamma^F(0) = \Gamma(0)$
$\boldsymbol{F_k}$	Set of feasible successors *in the sequence* of candidate $k \in K$ if k is disassembled, with $$F_k = K \setminus \{k\} \cup \Gamma^F(k) \quad (13)$$
$\boldsymbol{t_k}$	Cumulative disassembly time if k is sequenced after s and before its successor *in the graph* with minimum time. We have $$t_k = t_{s,k} + \arg\min_{j \in F_k} \left\{t_{k,j}\right\} \quad (14)$$
$\boldsymbol{M}$	Set of mandatory components in K with $M = K \cap \mathcal{M}$
$\boldsymbol{\bar{M}}$	Set of non mandatory components in K with $\bar{M} = K \setminus M$ and obviously we have $K = M \cup \bar{M}$
$\boldsymbol{t^M, t^{\bar{M}}}$	Minimum of cumulative times over mandatory candidates and non mandatory ones, respectively ($t^M = \min_{m \in M}\{t_m\}$ and $t^{\bar{M}} = \min_{\bar{m} \in \bar{M}}\{t_{\bar{m}}\}$)
$\boldsymbol{p_m}$	Probability of mandatory candidate m to be sequenced in position Pos $$p_m = \left(t^M/t_m\right) / \sum_{j \in M}\left(t^M/t_j\right) \quad (15)$$

Score and insertion position

$\boldsymbol{\Delta_{\bar{m}}}$	Score of non mandatory candidate $\bar{m}$ based upon the maximum time loss associated with its non inclusion in the sequence
$\boldsymbol{\delta_{\bar{m}}}$	Position in the sequence at which the maximum $\Delta_{\bar{m}}$ occurs
$\boldsymbol{\delta^{\text{best}}_{\bar{m}}}$	Saving of position $\delta_{\bar{m}}$ each time we obtain an improved sequence S^{best}
$\boldsymbol{i^*}$	Index of the non mandatory component considered for possible insertion in set $\mathcal{M}$

Table 2: Notations and definitions for the heuristic

(a) RGSA

RGSA Initialization

$\mathcal{M} = \mathcal{T} \cup_{j \in \mathcal{T}} \hat{\Gamma}^{-1}(j)$
Compute $|\mathcal{M}|$
$S^{\text{best}} := \mathcal{M}$
$\text{Obj}(S^{\text{best}}) = \sum_{(i,j) \in \mathcal{M}} t_{i,j}$
$\Delta_j = 0, \forall j = 1..n$
$\delta_j^{\text{best}} = -1, \forall j = 1..n$
$i^* = -1$

Multi-pass sequencing

do

  Initialization
  $\delta_j = -1, \forall j = 1..n$
  NoImprovement=0
  ImproveInsert=0

  do

    Sequencing procedure
    if $\text{Obj}(S) < \text{Obj}(S^{\text{best}})$
      $\text{Obj}(S^{\text{best}}) = \text{Obj}(S)$
      $S^{\text{best}} := S$
      $\delta_j^{\text{best}} := \delta_j, \forall j = 1..n$
      NoImprovement=0
      ImproveInsert+ = 1
    else
      NoImprovement+=1
  while NoImprovement $< N$

  RGSA updates
  if $i^* > 0$
    if ImproveInsert > 0
      $\mathcal{M} := \mathcal{M} \cup \{i^*\}$
      $\bar{\mathcal{M}} := \bar{\mathcal{M}} \setminus \{i^*\}$
    else
      $\Delta_{i^*} := -1$

  Compute $\Delta = \max_{\bar{m} \in \bar{\mathcal{M}}} \{\Delta_{\bar{m}}\}$
  if $\Delta > 0$
    Determine $i^* = \arg\max_{\bar{m} \in \bar{\mathcal{M}}} \{\Delta_{\bar{m}}\}$

while $\Delta > 0$

(b) Sequencing procedure

Initialize sequence (Pos = 1)

$s = 0$
$S_1 = \{0\}$
$\text{Obj}(S_1) = 0$

for Pos = 2.. $|\mathcal{M}| - 1$

  Update sets
  Determine K using Eq. (12)

  if $i^* > 0$ **and** $\delta_{i^*}^{\text{best}} > 0$
    if $\text{Pos} = \delta_{i^*}^{\text{best}}$
      $\mathcal{M} := \mathcal{M} \cup \{i^*\}$
    else
      $\mathcal{M} := \mathcal{M} \setminus \{i^*\}$
    Update $|\mathcal{M}|$

  Select mandatory compo
  for all candidates $m \in M$, $M = K \cap \mathcal{M}$
    Compute cumulative time t_m
      Determine set $\Gamma^F(m)$ with Eq. (11)
      Determine set F_m with Eq. (13)
      Compute t_m with Eq. (14)

  Compute $t^M = \min_{m \in M} \{t_m\}$

  for all candidates $m \in M$
    Compute p_m with Eq. (15)

  Draw $r = \pi + u$, where $u \sim U[0, 1 - \pi]$ and
      π is a parameter depending on n

  Select m^* accordingly
  if $m^* = i^*$
    $|\mathcal{M}| + = 1$

  Update score and insertion position
  for all candidates $\bar{m} \in \bar{M}$, $\bar{M} = K \setminus M$
    Compute cumulative time $t_{\bar{m}}$

  Compute $t^{\bar{M}} = \min_{\bar{m} \in \bar{M}} \{t_{\bar{m}}\}$
  if $t^{\bar{M}} < t^M$
    Determine $\bar{m}^* = \arg\min_{\bar{m} \in \bar{M}} \{t_{\bar{m}}\})$
    if $\Delta_{\bar{m}^*} < 1 - t^{\bar{M}}/t^M$ and $\Delta_{\bar{m}^*} \neq -1$
      Update
      $\Delta_{\bar{m}^*} := 1 - t^{\bar{M}}/t^M$
      $\delta_{\bar{m}^*} = \text{Pos}$

  Update sequence
  $S_{\text{Pos}} = S_{\text{Pos}-1} \cup \{m^*\}$
  $\text{Obj}(S_{\text{Pos}}) := \text{Obj}(S_{\text{Pos}-1}) + t_{s,m^*}$
  $s := m^*$

Last position (Pos = $|\mathcal{M}|$**)**
  Fill it with the remaining mandatory component

Table 3: Pseudocode of the RGSA and the sequencing procedure

The minimum $t^M = \min\{t_1, t_2\} = 57$ leads to a probability of selecting operation 1 equal to $p_1 = (57/131)/((57/131)+(131/131)) \backsimeq 0.31$ and consequently $p_2 = 0.69$. To give the algorithm a better chance to pick the best option, the random draw is set to a minimum value π. Preliminary experiments showed that a better performance is obtained when π is increased with n since with more operations, the selective pressure must increase to limit the choice to a few number of good operations. If for instance $\pi = 0.60$ and $r = 0.70$ then $m^* = 2$ is selected. The sub-procedure UPDATE SCORE AND INSERTION POSITION is skipped in Pos $= 2$ since set $\bar{M}$ of non mandatory operations is empty. UPDATE SEQUENCE leads to $S_2 = \{0, 2\}$, Obj$(S_2) = t_{0,2} = 29$, $s = 2$.

In position Pos $= 3$, the set K of candidates is updated using Eq. (12) so we get $K := \{1, 2\} \setminus \{2\} \cup \Gamma^F(2)$ with $\Gamma^F(2) = \Gamma(2) = \{4, 6, 7\}$, that is $K = \{1, 4, 6, 7\}$. The set of mandatory components is $M = \{1, 7\}$ on which we apply sub-procedure SELECT MANDATORY COMPO. We obtain $F_1 = \{1, 4, 6, 7\} \setminus \{1\} \cup \{3, 5\} = \{3, 4, 5, 6, 7\}$, thus $t_1 = t_{2,1} + \min\{t_{1,3}, t_{1,4}, t_{1,5}, t_{1,6}, t_{1,7}\} = 100$. The set of feasible successors of component 7 is $F_7 = \{1, 4, 6, 8, 10\}$, thus $t_2 = 69$. We therefore have $t^M = 69$ so $p_1 \backsimeq 0.41$ and $p_7 = 0.59$ leading to the selection of $m^* = 7$ (since $\pi = 0.60$). As the set of non mandatory candidates is not empty, $\bar{M} = \{4, 6\}$, sub-procedure UPDATE SCORE AND INSERTION POSITION is implemented. We have $F_4 = \{1, 6, 7\}$, $t_4 = 190$ and $F_6 = \{1, 4, 7\}$ so $t_6 = 63$. Time t_6 provides the minimum $t^{\bar{M}} = 63$, as it is lower than $t^M = 69$. This suggests that it might be better to insert non mandatory operation $\bar{m}^* = 6$ instead of $m^* = 7$ in Pos $= 3$. To reflect this potential loss of opportunity, a score $\Delta_6 = 1 - 63/69 \backsimeq 0.087$ is assigned to operation 6 and $\delta_6 = 3$ is recorded as the corresponding insertion position. UPDATE SEQUENCE gives $S_3 = \{0, 2, 7\}$, Obj$(S_3) = 57$, $s = 7$.

In position Pos $= 4$, the set of candidates is $K := \{1, 4, 6, 7\} \setminus \{7\} \cup \Gamma^F(7)$, that is $K = \{1, 4, 6, 8, 10\}$. The sequencing procedure proceeds by placing operation 1 in position 4 and sub-procedure UPDATE SCORE AND INSERTION POSITION provides a score $\Delta_4 = 1 - 57/74 \backsimeq 0.23$ for operation 4 and $\delta_4 = 4$ is recorded.

Let us note that for graph G_2, in position Pos $= 3$ the set of mandatory candidates is $M = \{1, 4, 6, 7\}$ instead of $\{1, 7\}$ for G_1. Focusing solely on component 4, we have $\Gamma(4) = \{16, 18, 20\}$ but $\Gamma^F(4)$ is empty since none of the operations $\{16, 18, 20\}$ can be performed even if component 4 was disassembled. Indeed, dismantling component 16 requires not only operation 4 to be performed before but also operations 6 and 9 that are not sequenced yet (the set $\Gamma^{-1}(16) \setminus \{4\} = \{6, 9\}$ is not included in $S_2 = \{0, 2\}$); and the same holds for components 18 and 20.

The sequencing procedure stops when all mandatory components in $\mathcal{M}$ are included in the sequence, the last one being added outside of the loop **for Pos** $= 2..|\mathcal{M}| - 1$ to avoid wrong updates of scores and positions for non mandatory operations. The first complete sequence we obtain for G_1 is $S = \{0, 2, 7, 1, 5, 11, 13, 3, 9, 8, 16\}$ with a total disassembly time Obj $(S) = 478 <$ Obj$(S^{\text{best}}) = 776$ s. Following the instructions in RGSA right after the sequencing procedure (see Table Table 3(a)), we set $S^{\text{best}} := S$ and we save the insertion positions of non mandatory components in δ_j^{best}. At the end of the first pass, the best sequence remains unchanged but scores and insertion positions have been updated several times. In RGSA UPDATES we determine the overall best score Δ which is attributed to component $i^* = 4$, $\Delta_4 = 0.719$, with an insertion position $\delta_4^{\text{best}} = 6$. Thus, another pass of the sequencing procedure is performed with component 4 being considered for possible insertion. Sub-procedure UPDATE SETS in the SEQUENCING PROCEDURE specifies that component $i^* = 4$ will possibly be inserted in position $\delta_4^{\text{best}} = 6$ and nowhere else, reminding its selection is probabilistic. At the end of the pass, following RGSA UPDATES, if the best sequence has improved, operation 4 is added to the set of mandatory components $\mathcal{M}$, be this operation included or not in S^{best} . Otherwise $i^* = 4$ is definitely excluded from further inclusion in $\mathcal{M}$ by setting its score Δ_4 to an arbitrary negative value ($\Delta_4 = -1$). The RGSA stops when all non mandatory components with positive scores have been assessed in this way, that is after 6 passes in our example where operations $4, 20, 6$ and 18 are examined successively. Note that operation 20 is finally inserted between operations 9 and 8 during the third pass, which leads to an improvement of the objective function (Obj$(S^{\text{best}}) = 472$). Another pass with part 20 is performed but with a negative insertion position so as to prolong the search. After the third pass, we do not record any further improvement of the sequence.

It should be noted that a candidate i^* for insertion in position $\delta_{i^*}^{\text{best}} > 0$ can contribute to reach a better solution as it allows for a shuffling of selection probabilities, be i^* inserted or not in S^{best} . Positions δ_j are reset to -1 at the beginning of each pass whereas scores Δ_j keep their maximum value. Thus, it is possible that a component with the best score so far has a negative insertion position in which case the algorithm amounts to perform another pass of sequencing on an unchanged set of mandatory components. Further improvements can be reached for the

search is simply prolonged over N and if so, the component will finally be included in $\mathcal{M}$ and excluded from $\bar{\mathcal{M}}$ in order to create diversity. Otherwise the component will definitely stay in set $\bar{\mathcal{M}}$.

Positive insertion positions δ_j^{best} correspond to positions where the maximum scores were reached in any sequence, be this sequence the best so far or not. Thus, position δ_j^{best} can be negative if the best score Δ_j recorded over all previous passes is not reached in any sequence of the current pass. This approach has proved to perform better than other options we tested, such as using always a positive insertion position corresponding to the best score, or recording the best scores and corresponding positions for the best local sequence over one pass. Besides, we also unsuccessfully tried to constraint the sequencing procedure to possibly insert a component from its best position, when positive, to all subsequent positions.

5 Matheuristic (MH)

Starting from an initial sequence S, the matheuristic considers each pair of adjacent operations (a, b) in S and determines a subset L of components to be optimized, with a as a root node, $a > 0$. This list L is first made of the direct successors of all components already sequenced before b to which we add $\{a, b\}$. Second, the list involves all the direct successors of the components previously determined, that is descendants of candidates. Once L is optimized (L^*) we replace in S the subsequence in L^* from a to b. In this way some operations can be inserted between a and $b$and to update S, we delete duplicates after b. If the updated sequence is improved, we record it. Another pair (a, b) is then selected in the updated S, by setting $a := b$ and $b = s_{\text{Pos }(a)+1}$. Optimization of subproblems stops when Pos(b) reaches the value of $|S| - 1$.

To allow for a simple writing of the pseudocode of our matheuristic, we generalize the definition of the set of feasible successors $\Gamma^F(k)$ given in Eq. (11). Set K keeps the same definition, thus including all candidate components to be sequenced at any iteration. Let us now define $\mathcal{S}$ as the set involving not only K but also the previously sequenced components at any iteration. Set $\Gamma^F(k)$ is rewritten as

$$\Gamma^F(k) = \left\{ j \in \Gamma(k)\,;\, j \notin K \mid \Gamma^{-1}(j) \setminus \{k\} \subset \mathcal{S} \right\}. \tag{16}$$

Let us note that in our RGSA, in Eq. (16) condition $j \notin K$ is never activated, and $\mathcal{S} = S_{\text{Pos}-1}$. Table 4 displays the pseudocode of our matheuristic.

To illustrate, let us consider graph G_2 in Figure 1(a). In step **MH Initialization**, we choose $S^{\text{initial}} = S^{\text{RGSA}} = \{0, 2, 6, 1, 7, 10, 5, 4, 3, 9, 16\}$, so we set $S^{\text{best}} = S^{\text{RGSA}}$; $S = S^{\text{best}}$ and Obj (S^{best}) = Obj $(S^{\text{RGSA}}) = 444$. We have $a = 2$; $b = 6$, with Pos$(a) = 2$; Pos$(b) = 3$.

At the first iteration of MH, we set $K = S_3 = \{0, 2, 6\}$ and $\mathcal{S} = S_3 = \{0, 2, 6\}$. In sub-procedure Determine new feasible succ. we use Eq. (16) to obtain $\Gamma^F(0) = \{1\}$ since successor 2 of node 0 is already included in K and predecessor of 1 is 0 which is already sequenced. We get $\Gamma^F(2) = \{4, 7\}$ as $\Gamma(2) = \{4, 6, 7\}$ but $\{6\} \in K$ and for $j = 4, 7$ we have $\Gamma^{-1}(j) \setminus \{2\} = \{2\} \setminus \{2\} = \varnothing$ and the empty set is necessarily included in $\mathcal{S}$. Set $\Gamma^F(6) = \varnothing$ since 6 has 16 as a successor in the graph but predecessors of 16 are 4, 6 and 9 and operations 4 and 9 are not in set $\mathcal{S}$.

We now set $K = \bigcup_{k \in K} \Gamma^F(k) \cup \{a, b\} = \Gamma^F(0) \cup \Gamma^F(2) \cup \Gamma^F(6) \cup \{2, 6\}$, so we have $K = \{1, 2, 4, 6, 7\}$. We thus have $\mathcal{S} = S_{\text{Pos}(a)-1} \cup K = \{0, 1, 2, 4, 6, 7\}$. Set K contains all components that will be re-sequenced and set $\mathcal{S}$ includes all operations sequenced so far and those that will be re-sequenced in the current iteration. Again, in sub-procedure Determine new feasible desc. we use Eq. (16) in the same way as before to obtain $\bigcup_{k \in \{1,2,4,6,7\}} \Gamma^F(k) = \{3, 5\} \cup \varnothing \cup \varnothing \cup \varnothing \cup \{8, 10\}$. Set L of components to be optimally resequenced is therefore $L = K \bigcup_{k \in \{1,2,4,6,7\}} \Gamma^F(k)$ that is $L = \{1, 2, 3, 4, 5, 6, 7, 8, 10\}$ with 10 being the only one non mandatory component. To apply the formulation of Han et al. (2013) on this subproblem, we take $a = 2$ as the root node and we extract from matrices Γ^{-1} and $(t_{i,j})_{i,j}$ lines and rows $l_i \in L$. Calling for CPLEX, we get $L^* = \{2, 7, 6, 1, 3, 4, 10, 5\}$ which leads to $S = \{0, 2, 7, 6, 1, 7, 10, 5, 4, 3, 9, 16\}$ with Obj $(S) = 474$, so S^{best} is not updated.

(a) MH

MH Initialization
$S^{\text{best}} := S^{\text{initial}}$
$\text{Obj}(S^{\text{best}}) = \text{Obj}(S^{\text{initial}})$
$S := S^{\text{best}}$
$a = s_2$, $\text{Pos}(a) = 2$
$b = s_3$, $\text{Pos}(b) = 3$

Iterate optimization of subsequences
do
  Set
  $K = S_{\text{Pos}(b)}$
  $\mathcal{S} = S_{\text{Pos}(b)}$
  Determine new feasible succ.
  for all $k \in K$
    Determine $\Gamma^F(k)$ with Eq. (16)

  Set
  $K = \bigcup_{k \in K} \Gamma^F(k) \cup \{a, b\}$
  $\mathcal{S} = S_{\text{Pos}(a)-1} \cup K$
  Determine new feasible desc.
  for all $k \in K$
    Determine $\Gamma^F(k)$ with Eq. (16)

  Determine subset L to optimize
  $L = K \bigcup_{k \in K} \Gamma^F(k)$

(b) MH...continued

  if $|L| > 2$
    Optimize L
    Set a as the root node
    Extract parameters of the sub-problem
    Call CPLEX to get $L^* = \{a, l_2^*, \ldots, b, \ldots, l_{|L|}^*\}$
    Consider $L^*_{\text{Pos }(b)} = \{a, l_2^*, \ldots, b\}$
    if $\left|L^*_{\textbf{Pos }(b)}\right| > 2$
      Update S
      In S, replace $\{a, b\}$ with $L^*_{\text{Pos }(b)}$
      Delete in S after b all $j \in L^*_{\text{Pos }(b)}$

      Compute $|S|$

      Compute $\text{Obj}(S)$
      if $\text{Obj}(S) < \text{Obj}(S^{\text{best}})$
        $S^{\text{best}} := S$
        $\text{Obj}(S^{\text{best}}) = \text{Obj}(S)$

      Update (a, b)
      $a := b$
      Determine $\text{Pos}(a)$ in S
      $b := s_{\text{Pos}(a)+1}$
      $\text{Pos}(b) := \text{Pos}(a) + 1$

while $\text{Pos}(b) < |S|$

Table 4: Pseudocode of our matheuristic MH

At the second iteration, we have $S = \{0, 2, 7, 6, 1, 10, 5, 4, 3, 9, 16\}$. We set $a = 6$ with Pos $(a) = 4$ so $b = s_5 = 1$. We optimize $L = \{6, 1, 3, 4, 5, 8, 9, 10, 11, 14\}$ and we obtain $L^* = \{6, 1, 3, 9, 4, 10, 5\}$ so S is left unchanged.

At the third iteration, $S = \{0, 2, 7, 6, 1, 10, 5, 4, 3, 9, 16\}$, with $a = 1$ and $b = 10$. The subsequence that we optimize is $L = \{1, 3, 4, 5, 8, 9, 10, 11, 14\}$ and we obtain $L^* = \{1, 3, 9, 4, 10, 5\}$. Thus $S = \{0, 2, 7, 6, 1, 3, 9, 4, 10, 5, 16\}$ and Obj $(S) = 383 < 444$, so we set $S^{\text{best}} = S$.

The next iterations do not lead to any additional improvement. Thus, MH finally provides Obj $(S^{\text{best}}) = 383$ whereas the optimal total disassembly time is 334. However MH reaches the optimal solution for G_1.

It should be mentioned that in our MH, we do not control the size of the subset of operations to be optimized. In the simulation experiments we present in the next section, the maximum proportions of these operations compared to mandatory components n with $n \in \{50, 100, 200, 300, 400, 500\}$ were respectively $\{2.80, 0.60, 0.35, 0.28, 0.125, 0.10\}$. For instance, in one instance with $n = 50$ and 10 mandatory components, the maximum size of the subset was equal to $2.8 \cdot 10 = 28$ components.

6 Simulation experiments

We first describe the instances and parameters setting in Subsection 6.1. The experimental design involved 2 phases. In the first phase, our incentive is to test the efficiency of our heuristics against the optimal solution provided by the MILP formulation of Han et al. (2013), for disassembly graphs with 50 components.

We decided not to include the Branch and Fathoming algorithm (B&F) that Han et al. (2013) developed to solve instances with a maximum number of 50 components, due to its performance. For the most complex DSP the authors solved to optimality, the B&F was not always able to find the optimal solution whereas applying CPLEX to their MILP formulation provided the optimum in several seconds only on average. Our incentive is not to provide an alternative to the exact approach when CPLEX can reach the optimal solution in a reasonable computation

time. Preliminary simulations showed that such "easy-to-solve" instances are related to specific combinations of parameters, like a high product structure complexity, a low number of items to disassemble or disassembly graphs with less than 50 components. In practice however, disassembly graphs have a low product structure complexity and recycling rates should be as high as possible. Thus, our simulation experiment encompasses more realistic instances for which CPLEX actually often failed to find an optimal solution, therefore emphasizing the need for heuristic solution methods.

Like in Han et al. (2013), our second phase considers large-sized instances (100 to 500 components) that cannot be solved to optimality. For large instances, these authors used exclusively several priority rules amongst which the Nearest Neighbor algorithm (NN) proved to be the most efficient. Therefore, along with our Randomized Greedy Sequencing Algorithm RGSA and our Matheuristic MH, we also included NN. Due to the very poor performance of NN, we tested a randomized version of that algorithm (RNN), in which the selection probability p_m of each mandatory component m was set equal to $\exp\left(|\mathcal{M}| \cdot \min_{j \in M}\left\{t_{s,j}\right\} / t_{s,m}\right)$, with $p_{\bar{m}} = 0$ (non mandatory operations) and then normalized. RNN was repeated for an execution time equal to that of RGSA and the best solution was recorded. As RNN provided much better results than NN, we chose to include it as well in the first phase.

Results of the 2 phases are presented and discussed in Subsection 6.2.

6.1 Instances generation and parameters setting

Number of components. For the sake of comparison with the optimal formulation of Han et al. (2013) a first set of experiments was conducted with a number of components $n = 50$ which is the maximum number that CPLEX could handle. As in Han et al. (2013), larger problems from 100 to 500 components were then considered.

Disassembly precedence graphs. We set the lowest level of each graph to the reasonable value of $\lfloor (n+1)^{0.5+0.01 \cdot \text{Rep}} \rceil$ where Rep is the seed used for replications with Rep = 1..5. To generate disassembly graphs, we used the product structure complexity index $\mathcal{C}$ of Kimms (1997), with $\mathcal{C} = 0$ for complete divergent graphs in which the number of arcs equals the number of components. We have $\mathcal{C} = 1$ when each component is connected to all components at lower levels. In practice disassembly graphs tend to have complexity indices close to zero because minimizing the number of connections is one of the strategies of design for disassembly. We therefore considered $\mathcal{C} = 0$, $\mathcal{C} = 0.05$ and $\mathcal{C} = 0.031$, this positive value corresponds to the structure complexity of an electronic calculator used as a real case in Han et al. (2013). Our graph generator randomly allocates components at each level, arcs are then created one by one at random until all nodes are connected with the desired structure complexity.

Recycling rate / minimum disassembly rate. The European directive on electrical and electronic equipment wastes (WEEE) set for year 2015 a minimum recycling rate of 50% for most categories of WEEE such as small household appliances, electrical and electronic tools or medical devices. We thus considered a minimum disassembly rate $\rho = 0.50$ in all instances, leading to a number of mandatory components to be disassembled $|\mathcal{M}| = \lfloor \rho \cdot n \rceil$. More ρ values were examined for $n = 50$ components as we set $\rho \in \{0.20, 0.35, 0.50, 0.65, 0.80\}$.

The set $\mathcal{M}$ of mandatory components was iteratively filled at random based upon their number of ancestors. At each iteration, the desired number of ancestors of a mandatory operation is a uniformly distributed random fraction of the maximum possible number of ancestors.

Disassembly time. Following Gonzalez and Adenso-Dìaz (2006), the disassembly time t_{ij} is defined as $t_{ij} = t_j \cdot \delta_1 \cdot \delta_2$, where t_j is the disassembly time of component j and θ_1 and θ_2 are correction factors to account for a possible speed reduction occurring respectively when moving/rotating component j is necessary and when tool changing is required, with $\theta_1 \in \{1.00, 1.15\}$ and $\theta_2 \in \{1.00, 1.10\}$ picked at random with equal probabilities. Based on case studies (Go et al., 2012 and Luo et al., 2016), disassembly times t_j in seconds were uniformly drawn at random in the range $\{10, \ldots, 100\}$ with probability 0.85 and in the range $\{101, \ldots, 150\}$ with complementary probability.

Parameters of the RGSA. The number of repetitions of the sequencing procedure with no improvement was set to $N = 300$ for all instances, whatever the number of components. Pilot studies showed that the best minimum probability π for selecting a mandatory component was equal to 0.60 for 50 components, 0.70 for 100 components, 0.80 for 200 and 300 components, and 0.85 for 400 and 500 components.

6.2 Simulation results

6.2.1 Medium-sized instances

In this first set of experiments with $n = 50$, we generated disassembly precedence graphs with a complexity $\mathcal{C} \in \{0, 0.031, 0.05\}$ and we considered five values of the minimum disassembly rate $\rho \in \{0.20, 0.35, 0.50, 0.65, 0.80\}$. Performing 5 replications for each combination of parameters led to 75 problems to solve. The MILP of Han et al. (2013) was coded in C and linked with the CPLEX callable optimization library version 12.5. We run CPLEX with a time limit of 3600 s, RGSA, RNN (with same execution time as RGSA), and the Matheuristic with RGSA and RNN as initial solutions. We also applied the Nearest Neighbor algorithm (NN) but we finally excluded it from the results due to its quite poor performance (total disassembly times were about 64% higher than those of CPLEX on average). Detailed results on the 75 problems are provided in Table 8, Appendix B. Table 5 displays the average Gap/deviations and execution times over replications, for each combination of parameters values, as well as averages over all cases.

		CPLEX			**RGSA**			**RNN**			**MH-RGSA**			**MH-RNN**		
Rate	Complex.	Obj.	Gap	CPU	Obj.	Dev.	CPU	Obj.	Dev.	CPU	Obj.	Dev.	CPU	Obj.	Dev.	CPU
0.20	0	441.80	27.69	3311.84	538.40	23.74	0.01	527.80	21.53	0.01	431.00	1.49	13.59	411.60	-3.78	11.16
	0.031	357.80	4.93	2320.53	498.60	39.80	0.01	439.80	20.80	0.01	444.80	25.17	13.04	380.00	5.32	9.31
	0.05	312.20	0.00	413.70	411.80	31.93	0.01	401.00	28.32	0.01	350.60	12.25	1.94	332.40	6.19	2.70
0.35	0	748.40	44.24	3617.39	668.40	-7.34	0.06	669.80	-6.36	0.06	580.60	-18.60	39.48	577.60	-18.89	31.39
	0.031	599.80	26.97	3246.57	601.80	3.40	0.04	608.60	4.17	0.04	566.40	-3.32	9.48	559.20	-4.91	9.97
	0.05	543.20	16.75	3559.73	648.20	18.92	0.05	641.60	18.86	0.05	570.80	5.51	13.04	566.60	4.02	9.46
0.50	0	1095.20	52.41	3616.45	750.20	-27.57	0.14	830.40	-18.53	0.14	698.40	-30.92	38.05	751.20	-25.16	34.69
	0.031	812.00	36.72	3644.16	744.40	-6.93	0.07	767.40	-3.94	0.07	710.80	-10.94	22.55	716.80	-10.59	19.49
	0.05	896.40	37.07	3610.01	765.60	-11.29	0.07	841.40	-2.58	0.07	754.80	-12.61	19.60	788.20	-8.06	11.36
0.65	0	1287.80	56.32	3614.94	818.80	-34.88	0.15	884.80	-29.88	0.15	818.80	-34.88	68.33	834.60	-33.45	73.44
	0.031	1164.60	50.71	3621.88	826.80	-28.88	0.09	886.60	-23.86	0.09	804.60	-30.79	32.73	858.80	-26.20	37.98
	0.05	980.00	37.27	3617.95	894.40	-8.40	0.08	1008.20	3.05	0.08	883.60	-9.58	14.75	949.40	-2.81	10.57
0.8	0	1801.80	64.60	3614.80	975.80	-45.24	0.18	1026.00	-42.42	0.18	956.60	-46.34	72.43	968.00	-45.60	158.35
	0.031	1281.80	47.55	3612.45	996.80	-22.21	0.13	1111.00	-13.27	0.13	977.40	-23.75	53.43	1044.40	-18.64	52.77
	0.05	1263.00	42.97	3620.88	1046.00	-15.41	0.13	1149.80	-7.26	0.13	1037.60	-16.08	40.89	1085.00	-11.76	31.73
	Overall	905.72	36.41	3269.55	745.73	-6.03	0.08	786.28	-3.43	0.08	705.79	-12.89	30.22	721.59	-12.95	33.62

Table 5: 50 components - average results over replications

There is a clear inverse relationship between the performance of CPLEX and that of the heuristics. CPLEX reached the optimal solution only in 12 cases over 75, all of them being obtained with low disassembly rates ($\rho \leq 0.35$). CPLEX gaps to optimality increase with ρ for the search space is augmented accordingly. However an improvement of the CPLEX solutions is always observed as the complexity index increases because more constraints of type (5) and (6) are activated, which tightens the formulation. For the lowest ρ value, $\rho = 0.20$, the best heuristic is MH-RNN with an average deviation of 2.58% from CPLEX and a quite low average CPU time of 7.72 s versus 2015.36 s for CPLEX. With $\rho = 0.35$, the best heuristic is again MH-RNN with an average deviation of -6.59% and a CPU time of 16.94 s, whereas CPLEX required on average 3474.56 s. When $\rho \geq 0.50$, all heuristics outperform CPLEX. MH-RGSA becomes the best heuristic possibly because the inclusion of non mandatory components in the sequence ameliorates the performance when the disassembly rate is increased. Let us note that in all cases, MH takes longer to produce a solution when $\mathcal{C} = 0$ than it does for instances with $\mathcal{C} > 0$ in which subsets of components to be optimized are always smaller (some operations have more than one predecessor to be dismantled). Likewise, the performance of MH is greater when $\mathcal{C} = 0$.

6.2.2 Large instances (from 100 to 500 components)

For large problems with $n \in \{100, 200, 300, 400, 500\}$, we fixed $\rho = 0.5$ and $\mathcal{C} = 0.031$ throughout. Performing 5 replications for each $n-$value, we got 25 problems on which we applied RGSA, RNN and the matheuristic with RGSA as an initial solution. The rationale for this choice comes from the significantly better performance of RGSA for large problems, with an average improvement of 6.25% over RNN. For our instances with 50 components, the average improvement of RGSA over RNN was only 1.13% which made it worth to try both heuristics as initial solutions for MH. From $n \geq 300$, MH-RGSA was no longer used since some subsequences to optimize included more than 50 components. We thus applied MH\D, a version of the matheuristic without considering descendants of candidates. In the pseudocode given in Table 4, MH\D simply skips sub-procedure DETERMINE NEW FEASIBLE DESC. RGSA was used as a benchmark since it represents the best compromise between solution quality and execution time. Detailed results for the 25 problems are given in Table 9 in Appendix B.

Table 6 displays for each heuristic and each $n-$value the average over all replications of the deviations of solutions to RGSA as well as the average CPU. Note that the CPU for RNN is not reported since the running time of RNN was set equal to that of RGSA. The execution time of NN was negligible ($2.14 \cdot 10^{-4}$ s on average) so we did not report it. Below the deviations we also indicated in brackets the number of times RGSA was outperformed over the 5 replications. And below the CPU for MH, we give the time limit we had to set for CPLEX to avoid out of memory issues. For MH, in column "Neighb.", we provide an indicator of the neighborhood size explored by CPLEX, defined as the average of maximum number of components to be optimally sequenced, expressed in percentage of the number of mandatory components. For instance, with $n = 100$ and MH-RGSA, the average maximum size of components optimally sequenced in each replication is equal to 25.80, which leads to a neighborhood size of 51.60% (25.80/50).

	RGSA		**NN**(*)	**RNN**	**MH-RGSA**			**MH\D-RGSA**		
n	Obj.	CPU s	Dev. %	Dev. %	Dev. %	CPU s	Neighb. %	Dev. %	CPU s	Neighb. %
100	1269.2	0.27	75.32	9.10	-3.02	134.84	51.60	-0.93	11.63	31.60
			(0)	(0)	(2)	(no)		(1)	(no)	
200	2339.8	0.87	52.67	9.40	-7.98	4130.42	32.60	-1.43	44.86	18.40
			(0)	(1)	(4)	(no)		(1)	(no)	
300	3078.8	1.76	44.70	8.81	-0.52	1578.87	23.60	0	35.81	13.20
			(0)	(0)	(1)	(120 s)		(0)	(no)	
400	4111.2	3.33	31.92	3.42				-0.22	68.66	11.40
			(0)	(1)				(1)	(240 s)	
500	4921.0	4.70	31.70	3.90				-0.42	86.41	9.28
			(0)	(1)				(1)	(240 s)	

(*) Han et al. (2013)

Table 6: Results for large instances

For $n < 300$, MH-RGSA is able to provide substantial improvements over RGSA but at the price of dramatic execution time increases. With a much more reasonable CPU time, MH\D-RGSA can outperform RGSA but in a limited number of cases (one case over five for each n, except for $n = 300$). It should be noted that when n increases, the performance of MH degrades for the explored neighborhood relative to the problem size decreases.

Finally, let us note that an additional pass of the MH did not change the results.

7 Conclusion

For real-world instances, the disassembly sequencing problem has been primarily solved using metaheuristics which require fine tuning to obtain good quality solutions. To the best of our knowledge this paper is the first that develops a randomized greedy heuristic and a matheuristic that are easy to implement and capable to efficiently solve medium to large scale instances. For medium-sized problems, our randomized greedy sequencing algorithm (RGSA) outperformed CPLEX whenever it was unable to reach the optimal solution, that is as soon as the minimum

disassembly rate was greater than 50%. Starting from the RGSA solution, our matheuristic offered further significant improvements. From 300 components, improvements of the RGSA solution by the matheuristic MH-RGSA become less significant and execution times are pretty high but the use of MH\D is always worthwhile since it might ameliorate the solution in a quick execution time.

Possible enhancements of our solution methods include a refinement of the selection process of non mandatory components in RGSA, especially for low disassembly rates as well as exploration of other neighborhoods in the matheuristic, like picking pairs of non adjacent operations, eventually at random, between which the optimal subsequence is inserted.

Acknowledgments

Jully Jeunet is grateful for her time as a visiting researcher at DIGEP, Politecnico di Torino, where this research has been conducted.

A Alternate formulations to the DSP

In these formulations, we define $u_{j,k}$ as the binary variable that takes a value of one if operation j occupies position k in the sequence. Variables $z_{i,j}$ no longer exist and other variables keep the same definition as in Table 1.

In the linear formulation, the objective to minimize is the same as in Eq. (1) subject to the following constraints.

Constraints to link variables $x_{i,j}$ with variables $u_{j,k}$. The variable $x_{i,j}$ takes a value of 1 if i immediately precedes j in the sequence which means that i occupies position k ($u_{i,k} = 1$) and j occupies position $k+1$ ($u_{j,k+1} = 1$). In all other cases $x_{i,j}$ must equal zero: i occupies k ($u_{i,k} = 1$) and j does not occupy $k+1$ ($u_{j,k+1} = 0$); i does not occupy k ($u_{i,k} = 0$) and j occupies $k+1$ ($u_{j,k+1} = 1$); i and j do not occupy position k and $k+1$, respectively ($u_{i,k} = u_{j,k+1} = 0$). Obviously we have $x_{i,j} = u_{i,k} \cdot u_{j,k+1}$ which can be linearized as follows

$$u_{i,k} + u_{j,k+1} - x_{i,j} < 2, \quad \forall i = 0..n, \quad j = 0..n, \quad k = 1..n. \tag{17}$$

As $x_{i,j}$ is in the objective function to be minimized, $x_{i,j}$ will take a value of zero when $u_{i,k} + u_{j,k+1} = 0$ or 1.

Disassembly and position. If component j is not disassembled ($y_j = 0$) then it does not occupy any position in the sequence. Conversely, if j is disassembled ($y_j = 1$) then j occupies one and only one position in the sequence. This is written as

$$\sum_{k=1..n+1} u_{j,k} = y_j, \quad \forall j = 0..n. \tag{18}$$

It should be noted that we necessarily have $y_0 = 1$ and node 0 occupies position 1 in the sequence thus we have $u_{0,1} = 1$ and $u_{0,k} = 0, \ \forall k = 2..n+1$ and $u_{j,1} = 0, \ \forall j = 1..n$.

Adjacency constraints between occupied positions. Since disassembly is selective, all disassembled components must occupy adjacent positions in the sequence. This means that if position k is such that $\sum_{j=1..n} u_{j,k} = 0$ thus all subsequent positions must not be occupied. This is written as

$$\sum_{j=0..n} u_{j,k} - \sum_{j=0..n} u_{j,k+1} \geq 0, \quad \forall k = 1..n. \tag{19}$$

Precedence constraints. If j is disassembled ($y_j = 1$) and if j occupies position k in the sequence ($u_{j,k} = 1$) thus all its direct predecessors $i \in \Gamma^{-1}(j)$ must have been disassembled before, which is written as

$$\left|\Gamma^{-1}(j)\right| \cdot u_{j,k} - \sum_{i \in \Gamma^{-1}(j)} \sum_{l=1..k-1} u_{i,l} \leq 0, \quad \forall j = 1..n, \quad k = 2..n+1, \tag{20}$$

In the quadratic formulation, the objective is written as

$$\sum_{i=0..n} \sum_{j=0..n} \sum_{k=1..n+1} c_{i,j} \cdot u_{i,k} \cdot u_{j,k+1}, \tag{21}$$

subject to the same constraints as before except constraints (17) that link variables $x_{i,j}$ and $u_{j,k}$ since variables $x_{i,j}$ no longer exist in this formulation.

Table 7 gives the number of variables and constraints for each formulation. Although our linear formulation has less constraints than that of Han et al. (2013), preliminary testing showed that both formulations were comparable in terms of execution time so we kept the existing published formulation. Even though the quadratic formulation saves a significant number of constraints, it led to very large execution times compared with the linear formulations.

Formulation	**#Variables**	**#Constraints**
Han et al. (2013)	$2(n+1)^2$	n^3+5n^2+8n+5
Our linear	$2(n+1)^2$	n^3+3n^2+3n+1
Our quadratic	$(n+1)^2+n+1$	n^2+2n+1

Table 7: Number of variables and constraints for the three formulations

B Tables of detailed results

Table 8 provides the results for each of the 40 instances with 50 components. There are 10 problems per minimum disassembly rates $\rho \in \{0.20, 0.35, 0.50, 0.65\}$. For each ρ value, 5 replications are performed with the two complexity indices $\mathcal{C} \in \{0, 0.031\}$. For CPLEX 3600 s, we give the objective value which is the total disassembly time of the sequence, the gap to best bound in % and the CPU in seconds. For the other methods, we provide the deviation (Dev.) in % of the solution to that of CPLEX 3600s as well as the execution time in seconds.

			CPLEX			RGSA			RNN			MH-RGSA			MH-RNN		
Rate	Complex.	Rep.	Obj	Gap	CPU	Obj	Dev.	CPU	Obj	Dev.	CPU	Obj	Dev.	CPU	Obj	Dev.	CPU
0.2	0	1	335	0.00	2056.44	480	43.28	0.01	449	34.03	0.01	462	37.91	5.91	395	17.91	12.24
		2	575	47.26	3623.16	647	12.52	0.01	582	1.22	0.01	432	-24.87	3.00	456	-20.70	6.25
		3	338	18.84	3621.96	414	22.49	0.01	424	25.44	0.01	361	6.80	23.98	362	7.10	13.12
		4	453	38.07	3632.44	579	27.81	0.01	580	28.04	0.01	436	-3.75	28.61	437	-3.53	16.17
		5	508	34.28	3625.18	572	12.60	0.01	604	18.90	0.01	464	-8.66	6.43	408	-19.69	8.04
	0.031	1	417	0.00	1451.22	613	47.00	0.01	552	32.37	0.01	579	38.85	22.20	528	26.62	12.03
		2	437	24.65	3608.96	575	31.58	0.01	619	41.65	0.01	453	3.66	12.07	437	0.00	6.08
		3	273	0.00	3036.21	377	38.10	0.01	302	10.62	0.01	358	31.14	9.33	273	0.00	2.91
		4	289	0.00	16.40	430	48.79	0.01	317	9.69	0.01	367	26.99	3.93	289	0.00	3.22
		5	373	0.00	3489.88	498	33.51	0.01	409	9.65	0.01	467	25.20	17.66	373	0.00	22.29
	0.05	1	354	0.00	37.85	422	19.21	0.02	402	13.56	0.02	366	3.39	1.70	378	6.78	3.51
		2	317	0.00	43.39	449	41.64	0.01	455	43.53	0.01	436	37.54	0.31	387	22.08	0.57
		3	333	0.00	1656.19	469	40.84	0.01	460	38.14	0.01	371	11.41	2.79	340	2.10	5.48
		4	258	0.00	21.80	329	27.52	0.01	306	18.60	0.01	281	8.91	3.76	258	0.00	3.20
		5	299	0.00	309.25	390	30.43	0.01	382	27.76	0.01	299	0.00	1.12	299	0.00	0.73
0.35	0	1	502	18.64	3605.58	574	14.34	0.08	635	26.49	0.08	574	14.34	38.89	587	16.93	29.12
		2	827	50.86	3609.12	773	-6.53	0.04	679	-17.90	0.04	650	-21.40	28.89	585	-29.26	18.56
		3	977	60.31	3617.29	604	-38.18	0.05	643	-34.19	0.05	527	-46.06	46.25	564	-42.27	51.24
		4	667	43.55	3626.50	643	-3.60	0.06	643	-3.60	0.06	520	-22.04	29.61	520	-22.04	28.42
		5	769	47.86	3628.47	748	-2.73	0.06	749	-2.60	0.06	632	-17.82	53.75	632	-17.82	29.60
	0.031	1	647	33.67	3610.12	634	-2.01	0.02	607	-6.18	0.02	603	-6.80	5.69	607	-6.18	0.66
		2	756	44.97	3611.64	555	-26.59	0.03	602	-20.37	0.03	555	-26.59	6.47	561	-25.79	11.03
		3	606	40.70	3611.18	601	-0.83	0.04	628	3.63	0.04	581	-4.13	16.78	581	-4.13	7.87
		4	492	0.00	1782.35	670	36.18	0.05	665	35.16	0.05	587	19.31	7.34	541	9.96	14.56
		5	498	15.52	3617.58	549	10.24	0.05	541	8.63	0.05	506	1.61	11.11	506	1.61	15.72
	0.05	1	622	19.13	3611.43	836	34.41	0.04	687	10.45	0.04	630	1.29	25.51	630	1.29	24.60
		2	598	23.67	3604.99	658	10.03	0.04	675	12.88	0.04	615	2.84	9.03	675	12.88	7.79
		3	461	24.65	3613.24	530	14.97	0.07	600	30.15	0.07	526	14.10	6.83	450	-2.39	7.25
		4	517	0.00	3364.53	718	38.88	0.05	698	35.01	0.05	615	18.96	16.39	596	15.28	5.00
		5	518	16.31	3604.45	499	-3.67	0.05	548	5.79	0.05	468	-9.65	7.46	482	-6.95	2.64
0.5	0	1	832	40.37	3616.18	645	-22.48	0.12	757	-9.01	0.12	645	-22.48	39.34	703	-15.50	38.92
		2	1073	56.18	3613.26	803	-25.16	0.13	842	-21.53	0.13	771	-28.15	37.60	737	-31.31	30.49
		3	900	52.86	3612.08	568	-36.89	0.16	731	-18.78	0.16	568	-36.89	56.79	731	-18.78	24.50
		4	792	40.31	3626.72	744	-6.06	0.17	817	3.16	0.17	721	-8.96	40.06	759	-4.17	34.61
		5	1879	72.34	3613.99	991	-47.26	0.13	1005	-46.51	0.13	787	-58.12	16.47	826	-56.04	44.94
	0.031	1	883	38.60	3615.71	721	-18.35	0.03	748	-15.29	0.03	721	-18.35	15.18	748	-15.29	6.66
		2	972	50.10	3605.65	771	-20.68	0.07	886	-8.85	0.07	741	-23.77	21.98	804	-17.28	36.29
		3	665	33.31	3615.02	633	-4.81	0.08	666	0.15	0.08	633	-4.81	17.94	658	-1.05	6.22
		4	657	21.54	3728.59	727	10.65	0.09	776	18.11	0.09	693	5.48	40.97	643	-2.13	22.12
		5	883	40.05	3655.84	870	-1.47	0.08	761	-13.82	0.08	766	-13.25	16.68	731	-17.21	26.16
	0.05	1	656	9.60	3607.24	704	7.32	0.06	794	21.04	0.06	693	5.64	5.22	763	16.31	4.43
		2	1013	47.94	3605.82	815	-19.55	0.05	974	-3.85	0.05	813	-19.74	3.58	830	-18.07	18.68
		3	1187	62.10	3620.38	705	-40.61	0.07	786	-33.78	0.07	705	-40.61	5.06	712	-40.02	7.84
		4	863	34.62	3612.88	911	5.56	0.08	853	-1.16	0.08	870	0.81	68.70	853	-1.16	9.28
		5	763	31.07	3603.75	693	-9.17	0.10	800	4.85	0.10	693	-9.17	15.42	783	2.62	16.59
0.65	0	1	941	40.88	3619.93	777	-17.43	0.11	817	-13.18	0.11	777	-17.43	39.55	817	-13.18	75.08
		2	1496	64.51	3608.99	881	-41.11	0.14	1029	-31.22	0.14	881	-41.11	27.99	833	-44.32	39.80
		3	1107	54.34	3606.09	743	-32.88	0.16	791	-28.55	0.16	743	-32.88	125.99	745	-32.70	93.84
		4	1478	63.98	3610.61	838	-43.30	0.18	886	-40.05	0.18	838	-43.30	117.86	877	-40.66	112.93
		5	1417	57.90	3629.08	855	-39.66	0.14	901	-36.41	0.14	855	-39.66	30.27	901	-36.41	45.54
	0.031	1	1196	48.29	3616.12	875	-26.84	0.09	924	-22.74	0.09	851	-28.85	26.06	851	-28.85	10.82
		2	1247	54.82	3622.92	818	-34.40	0.09	907	-27.27	0.09	792	-36.49	19.90	907	-27.27	33.41
		3	1119	52.38	3621.83	819	-26.81	0.11	844	-24.58	0.11	791	-29.31	23.42	832	-25.65	30.14
		4	1066	48.63	3627.59	781	-26.74	0.09	805	-24.48	0.09	764	-28.33	74.20	805	-24.48	27.19
		5	1195	49.44	3620.94	841	-29.62	0.06	953	-20.25	0.06	825	-30.96	20.05	899	-24.77	88.32
	0.05	1	971	29.87	3606.05	886	-8.75	0.10	1007	3.71	0.10	886	-8.75	12.87	913	-5.97	3.27
		2	1064	41.26	3625.64	855	-19.64	0.07	1046	-1.69	0.07	855	-19.64	4.10	982	-7.71	5.04
		3	963	44.79	3621.46	828	-14.02	0.08	852	-11.53	0.08	828	-14.02	17.08	852	-11.53	12.67
		4	998	38.53	3622.38	970	-2.81	0.07	1104	10.62	0.07	962	-3.61	21.10	968	-3.01	6.78
		5	904	31.89	3614.24	933	3.21	0.09	1032	14.16	0.09	887	-1.88	18.58	1032	14.16	25.07
0.8	0	1	2186	70.48	3625.05	945	-56.77	0.16	1022	-53.25	0.16	945	-56.77	58.20	910	-58.37	144.11
		2	1720	63.34	3616.05	1027	-40.29	0.14	1151	-33.08	0.14	978	-43.14	77.77	1044	-39.30	61.20
		3	1566	62.50	3609.58	875	-44.13	0.18	936	-40.23	0.18	875	-44.13	71.16	865	-44.76	398.27
		4	1791	65.90	3619.27	982	-45.17	0.20	903	-49.58	0.20	935	-47.79	111.78	903	-49.58	106.14
		5	1746	60.80	3604.04	1050	-39.86	0.21	1118	-35.97	0.21	1050	-39.86	43.22	1118	-35.97	82.04
	0.031	1	1312	45.75	3612.20	1043	-20.50	0.15	1122	-14.48	0.15	1043	-20.50	48.37	1122	-14.48	23.38
		2	1355	52.91	3619.10	1020	-24.72	0.11	1148	-15.28	0.11	1020	-24.72	30.06	1148	-15.28	57.82
		3	1248	50.67	3621.51	930	-25.48	0.17	1009	-19.15	0.17	930	-25.48	49.34	985	-21.07	39.06
		4	1227	44.36	3605.22	964	-21.43	0.12	1143	-6.85	0.12	957	-22.00	90.50	932	-24.04	40.24
		5	1267	44.08	3604.20	1027	-18.94	0.08	1133	-10.58	0.08	937	-26.05	48.86	1035	-18.31	103.33
	0.05	1	1271	40.20	3611.49	1008	-20.69	0.13	1057	-16.84	0.13	968	-23.84	51.82	958	-24.63	29.49
		2	1541	53.20	3608.43	1079	-29.98	0.19	1247	-19.08	0.19	1079	-29.98	30.47	1033	-32.97	6.87
		3	1348	52.93	3632.61	978	-27.45	0.13	1076	-20.18	0.13	978	-27.45	60.52	1065	-20.99	65.90
		4	1057	34.81	3627.02	1100	4.07	0.08	1145	8.33	0.08	1098	3.88	48.68	1145	8.33	36.24
		5	1098	33.73	3624.85	1065	-3.01	0.14	1224	11.48	0.14	1065	-3.01	12.99	1224	11.48	20.16

Table 8: Detailed results for instances with 50 components

Table 9 displays the results for the 25 large instances with $\rho = 0.50$ and $\mathcal{C} = 0.031$ throughout and $n \in \{100, 200, 300, 400, 500\}$. The RGSA is used as a benchmark and for the matheuristics we provide in column "Size" the maximum number of components that was optimally sequenced.

		RGSA		**NN** (Han et al.)		**RNN**		**MH-RGSA**			**MH\D-RGSA**		
n	Rep	Obj	CPU	Dev.	CPU	Dev.	CPU	Dev.	CPU	Size	Dev.	CPU	Size
100	1	1407	0.34	77.68	0.00	12.22	0.34	0.00	72.69	24	0.00	1.85	17
	2	1114	0.17	43.36	0.00	10.59	0.17	0.00	78.59	25	0.00	17.08	16
	3	1001	0.38	89.81	0.00	8.39	0.38	0.00	178.30	24	0.00	14.87	15
	4	1436	0.23	64.97	0.00	3.13	0.23	-14.07	128.24	26	-4.67	5.65	15
	5	1388	0.22	100.79	0.00	11.17	0.22	-1.01	216.38	30	0.00	18.69	16
200	1	2350	1.12	39.96	0.00	6.47	1.12	-6.98	3119.51	35	0.00	21.65	19
	2	2078	1.03	52.50	0.00	17.18	1.03	0.00	4616.25	35	0.00	44.28	22
	3	2068	0.78	46.23	0.00	6.62	0.78	-5.46	4620.41	30	0.00	13.20	18
	4	2905	0.59	66.06	0.00	-0.41	0.59	-17.31	2459.83	30	-7.13	109.06	16
	5	2298	0.85	58.62	0.00	17.15	0.85	-10.14	5836.08	33	0.00	36.10	17
300	1	3113	2.02	34.24	0.00	5.14	2.02	0.00	1860.56	35	0.00	48.60	18
	2	3106	1.82	66.55	0.00	16.74	1.82	0.00	672.52	42	0.00	45.29	21
	3	3014	1.45	44.13	0.00	9.52	1.45	-2.59	2820.28	33	0.00	21.04	19
	4	3164	2.00	54.08	0.00	8.85	2.00	0.00	1493.69	34	0.00	34.00	20
	5	2997	1.48	24.49	0.00	3.80	1.48	0.00	1047.31	33	0.00	30.13	21
400	1	4161	3.81	20.81	0.00	-1.92	3.81				0.00	74.88	25
	2	4205	2.83	49.23	0.00	11.03	2.83				-1.09	123.34	25
	3	3946	2.79	29.40	0.00	3.40	2.79				0.00	37.86	19
	4	4163	3.19	34.30	0.00	0.58	3.19				0.00	52.62	24
	5	4081	4.04	25.85	0.00	4.02	4.04				0.00	54.62	21
500	1	4951	5.27	31.04	0.00	3.49	5.27				0.00	85.96	23
	2	4848	4.68	26.57	0.00	6.64	4.68				0.00	67.13	24
	3	4851	3.05	30.39	0.00	-1.44	3.05				-2.08	156.53	25
	4	4980	5.81	37.29	0.00	5.56	5.81				0.00	69.03	21
	5	4975	4.66	33.23	0.00	5.27	4.66				0.00	53.38	23

Table 9: Detailed results for large-sized instances with $\rho = 0.50$ and $\mathcal{C} = 0.031$

References

Adenso-Dìaz, B., Garcìa-Carbajal, S., & Gupta, S.M. 2008. A path-relinking approach for a bi-criteria disassembly sequencing problem. *Computers and Operations Research*, **35**, 3989–3997.

Alfieri, A., Matta, A., & Pastore, E. 2020. The time buffer approximated Buffer Allocation Problem: A row-column generation approach. *Computers and Operations Research*, **115**, 104835.

Bentaha, M.-L., Voisin, A., & Marangé, P. 2020. A decision tool for disassembly process planning under end-of-life product quality. *International Journal of Production Economics*, **219**, 386–401.

Chang, M.M.L., Ong, S.K., & Nee, A.Y.C. 2017. Approaches and challenges in product disassembly planning for sustainability. *Procedia CIRP*, **60**, 506–511.

Chiodoa, J.D., & Ijomahc, W.L. 2014. Use of active disassembly technology to improve remanufacturing productivity: automotive application. *International Journal of Computer Integrated Manufacturing*, **27**, 361–371.

Della Croce, F., Salassa, F., & T'kindt, V. 2014. A hybrid heuristic approach for single machine scheduling with release times. *Computers and Operations Research*, **45**, 7–11.

Edmunds, R., Kobayashi, M., & Higashi, M. 2012. Using constraint-satisfaction to optimise disassembly sequences generated from AND/OR information. *International Journal of Production Research*, **50**, 4105–4126.

Go, T.F., Wahab, D.A., Rahman, M.N. Ab., Ramli, R., & Hussain, A. 2012. Genetically optimised disassembly sequence for automotive component reuse. *Expert Systems with Applications*, **39**, 5409–5417.

Godichaud, M., & Amodeo, L. 2018. Collection-disassembly problem in reverse supply chain. *International Journal of Production Economics*, **199**, 16–25.

Gonzalez, B., & Adenso-Dìaz, B. 2006. A scatter search approach to the optimum disassembly sequence problem. *Computers and Operations Research*, **33**, 1776–1793.

Guerrero, W.J., Prodhon, C., Velasco, N., & Amaya, C.A. 2013. Hybrid heuristic for the inventory location-routing problem with deterministic demand. *International Journal of Production Economics*, **146**, 359–370.

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

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

Habibi, M.K.K., Battaïa, O., Cung, V.-D., & Dolgui, A. 2017. Collection-disassembly problem in reverse supply chain. *International Journal of Production Economics*, **183**, 334–344.

Han, H.-J., Yu, J.-M., & Lee, D.-H. 2013. Mathematical model and solution algorithms for selective disassembly sequencing with multiple target components and sequence-dependent setups. *International Journal of Production Research*, **51**, 4997–5010.

Huang, H.-H., Wang, M.H., & Johnson, M.R. 2000. Disassembly sequence generation using a neural network approach. *Computers and Operations Research*, **19**, 73–82.

Iassinovskaia, G., Limbourg, S., & Riane, F. 2017. The inventory-routing problem of returnable transport items with time windows and simultaneous pickup and delivery in closed-loop supply chains. *International Journal of Production Economics*, **183**, 570–582.

Johnson, M.R., & Wang, M.H. 1998. Economical evaluation of disassembly operations for recycling, remanufacturing and reuse. *International Journal of Production Research*, **36**, 3227–3252.

Kim, H.-W., & Lee, D.-H. 2017. An optimal algorithm for selective disassembly sequencing with sequence-dependent set-ups in parallel disassembly environment. *International Journal of Production Research*, **55**, 7317–7333.

Kim, H.-W., & Lee, D.-H. 2018. A sample average approximation algorithm for selective disassembly sequencing with abnormal disassembly operations and random operation times. *The International Journal of Advanced Manufacturing Technology*, **96**, 1341–1354.

Kim, H.-W., Park, C., & Lee, D.-H. 2018. Selective disassembly sequencing with random operation times in parallel disassembly environment. *International Journal of Production Research*, **56**, 7243–7257.

Kimms, A. 1997. *Multi-Level Lot-Sizing and Scheduling*. Production and Logistics. Physica-Verlag Heidelberg.

Lambert, A.J.D. 2006. Exact methods in optimum disassembly sequence search for problems subject to sequence dependent costs. *Omega*, **34**, 538–549.

Lambert, A.J.D. 2007. Optimizing disassembly processes subjected to sequence-dependent cost. *Computers and Operations Research*, **34**, 536–551.

Lambert, A.J.D., & Gupta, S.M. 2008. Methods for optimum and near optimum disassembly sequencing. *International Journal of Production Research*, **46**, 2845–2865.

Li, Jinlin, Chen, Xiaohong, Zhu, Zhanguo, Yang, Caijun, & Chu, Chengbin. 2019. A branch, bound, and remember algorithm for the simple disassembly line balancing problem. *Computers and Operations Research*, **105**, 47 – 57.

Luo, Y., Peng, Q., & Gu, P. 2016. Integrated multi-layer representation and ant colony search for product selective disassembly planning. *Computers in Industry*, **75**, 13–26.

Official Journal of the European Union. 2012. Directive 2012/19/EU of the European Parliament and of the Council of 4 July 2012 on Waste Electrical and Electronic Equipment (WEEE).

Priyono, A., Ijomah, W.L., & Bititci, U.S. 2015. Strategic operations framework for disassembly in remanufacturing. *Journal of Remanufacturing*, **5**, 1–16.

Raa, B., Dullaert, W., & Aghezzaf, E.-H. 2013. A matheuristic for aggregate production-distribution planning with mould sharing. *International Journal of Production Economics*, **145**, 29–37.

Ren, Y., Zhang, C., F.Zhaoc, Xiao, H., & G.Tian. 2018. An asynchronous parallel disassembly planning based on genetic algorithm. *European Journal of Operational Research*, **269**, 647–660.

Smith, S., Smith, G., & Chen, W.-H. 2012. Disassembly sequence structure graphs: An optimal approach for multiple-target selective disassembly sequence planning. *Advanced Engineering Informatics*, **26**, 306–316.

Soh, S.L., Ong, S.K., & Nee, A.Y.C. 2014. Design for disassembly for remanufacturing: Methodology and technology. *Procedia CIRP*, **15**, 407–412.

Tian, G., Zhou, M., & Chu, J. 2012a. A chance constrained programming approach to determine the optimal disassembly sequence. *IEEE Transactions on Automation Science and Engineerings*, **10**, 1004–1013.

Tian, G., Liu, Y., Tian, Q., & Chu, J. 2012b. Evaluation model and algorithm of product disassembly process with stochastic feature. *Clean Technologies and Environmental Policy*, **14**, 345–356.

Tian, G., Zhou, M., Chu, J., & Liu, Y. 2012c. Probability evaluation models of product disassembly cost subject to random removal time and different removal labor cost. *IEEE Transactions on Automation Science and Engineerings*, **9**, 288–295.

Tian, G., Zhou, M., & Li, P. 2018. Disassembly Sequence Planning Considering Fuzzy Component Quality and Varying Operational Cost. *IEEE Transactions on Automation Science and Engineering*, **15**(2), 748–760.

Tian, G., Ren, Y., Feng, Y., Zhou, M., Zhang, H., & Tan, J. 2019. Modeling and Planning for Dual-Objective Selective Disassembly Using and/or Graph and Discrete Artificial Bee Colony. *IEEE Transactions on Industrial Informatics*, **15**(4), 2456–2468.

Tian, Guangdong, Zhang, Honghao, Feng, Yixiong, Jia, Hongfei, Zhang, Chaoyong, Jiang, Zhigang, Li, Zhiwu, & Li, Peigen. 2017. Operation patterns analysis of automotive components remanufacturing industry development in China. *Journal of Cleaner Production*, **164**, 1363 – 1375.

Tseng, H.-E., Chang, C.-C., Lee, S.-C., & Huang, Y.-M. 2018. A block-based genetic algorithm for disassembly sequence planning. *Expert Systems with Applications*, **96**, 492–505.

Wang, M.H., & Johnson, M.R. 1995. Design for disassembly and recyclability: A concurrent engineering approach. *Concurrent Engineering*, **3**, 131–134.

Yeh, W.-C. 2012. Simplified swarm optimization in disassembly sequencing problems with learning effects. *Computers and Operations Research*, **39**, 2168–2177.

Zhang, X.F., Yu, G., Hu, Z.Y., Pei, C.H., & Ma, G.Q. 2014. Parallel disassembly sequence planning for complex products based on fuzzy-rough sets. *International Journal of Advanced Manufacturing Technology*, **72**, 231–239.