University of Birmingham

School of Engineering
Department of Mechanical Engineering

Individual Engineering Project

FINAL REPORT

MEng

Surname	Perrett
First Name(s)	James
ID number	1272539
Supervisor's Name	Yongjing Wang
Project Title	Robotic disassembly sequence planning with uncertain interference

Table of Contents

Abstract

With rapid advancement in AI and robotics the potential for autonomous disassembly has been unlocked. To achieve full autonomy, robots must be able to make intelligent and informed decisions about stochastic, non-deterministic objectives. In this study, a disassembly sequence planning method is developed for products with uncertain interference conditions. The non-binary, remote considering, space interference matrices presented by Wang et al are adapted for modelling parts with uncertain interference [1]. This novel, fuzzy, dynamic modelling method is used in combination with an iterative re-planning strategy, to allow for on-line adaptation to failure. The re-planning strategy features several research inspired mechanisms, which together, generate sequences reflecting the stochastic training data. Three products of differing complexity are considered. Viable and creative disassembly sequence plans are generated, and when evaluated as an analytical tool, results indicate a strong adaptation to past experience.

Keywords

Remanufacturing, Disassembly Sequence Planning, Fuzzy, Uncertainty, Industry 4.0

1 Introduction

Remanufacturing is recognised as a key industry in reducing and overcoming this centuries environmental and economic challenges[2-11]. The expanding industry tackles high energy consumption; the over-harvesting of raw materials; and harmful environmental emissions - all with a low cost[12].

Remanufacturing starts with dismantling end-of-life (EoL) products; the condition of which is unknown[13]. Hence, most disassembly processes use human operators for their flexibility. Robotics can provide superior productivity and repeatability, yet, even in highly automated manufacturing scenario, it is still the human operators who are the decision makers and problem solvers. This contrast between *automated* and *autonomous* disassembly has encouraged research into disassembly sequence planning (DSP). DSP is a method which allows robots to design optimised disassembly sequences and make intelligent decisions on non-deterministic problems[14-18]. In 1984, Bourjault was one of the first the investigate DSP modelling by using man-machine precedence relationships to create an assembly *tree*, or liaison graph[19]. This work was built upon by De Fazio and Whitney in 1987, who applied it to generate assembly sequences by defining connective states with binary variables[20].

In line with recent advances in artificial intelligence, several nature inspired optimisation techniques have been applied for generating optimal disassembly sequences[21-27]. Agrawal and Nallamothu et al. employed genetic algorithms (GA's) to produce optimised DSP's from a geometric description of an assembly[28]. Tseng, Huang et al. then adapted conventional GA's to form a novel 'Flatworm' algorithm of superior quality[29]. Notably, this paper utilised the concept of a *penalty value matrix* to assess the cost of changing tooling and direction. Remarkably, it appears that within existing literature, most adaptive optimisation methods cannot be used for online decision due to their complexity and iterative nature[30].

To the authors' knowledge, only one paper permits online decision making. Laili, Tao et al. allowed for real-time optimisation by considering bees algorithm in combination with the two pointer-method[31]. In this study, if a component were to fail during operation, the strategy ignores previously removed components and reconfigures a new DSP for the remaining assembly. Although successfully integrating cost and reliability prioritisation with real-time decision making, this technique lacks a learning mechanism. Thus, a gap exists in predicting real time uncertainties in product condition alongside a re-planning strategy. This paper intends to address this issue.

Previous research has attempted to model uncertainty using fuzzy data sets[32-36]. However, these models only consider *serviceability* and *human factors* in estimating economic viability. No such research addresses the issue of EoL product condition in relation to DSP, and the consequent maximised environmental benefits of increased disassembly completion. Furthermore, most listed literature cannot anticipate or adapt to failure in real time. This type of flexibility is at the core of industry 4.0[37-39].

By utilising a fuzzy ranking scheme to assess predicted product condition, similar to methods used in [29], [33] and [35], this study will quantify a live dynamic uncertainty interference matrix. Which through Monte-Carlo forward uncertainty propagation and iterative estimation, a re-planning algorithm will learn to make informed online removal decisions based upon the success of previous actions.

This paper is organised as follows: In Section 2, novel mathematical representations of physical assemblies in 2D space are defined. In section 3, the algorithms' dynamic re-planning process is dissected. In section 4, Iterative estimation and sequence ranking schemes are explained. In section

5, Several 3D case studies are examined to assess the algorithms' performance, and in section 6, concluding observations are made with respect to the wider context and future applications.

2 Fuzzy mathematical representation of physical assemblies

Traditionally, assemblies are defined in 3D space using an Assembly matrix, AND/OR graphs, a Space Interference matrix, and Contact matrices. The definitive tools use Boolean algebra to outline rules on disassembly operations, directions and orders. This paper proposes the use of two fuzzy dynamic matrices in combination with a traditional *Contact matrix* to define an assembly in 3D space. However, for the purpose of illustrating the proposed method a simple 2D assembly, shown in figure 1, will be considered in 4 directions.

The Contact matrix defines each components' contacting relations in each of the 4 or 6 directions. As figure 1 has 9 components and 4 removal directions (X+, X-, Y+, Y-), a 9x36 matrix is populated with 1's for contact (in the direction of motion, relative to another part) and 0's for non-contact. Table 2E presents a compressed Contact matrix or *Relation matrix*, which functions like an OR gate, outputting a value of 1 if the component shares contact with another part in any given direction. This matrix will be used to facilitate neighbouring penalties in the event of failure.

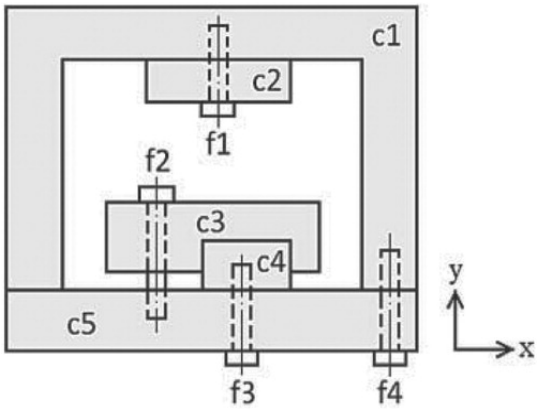

Figure 1. An example product (Jin et al. 2015 [39]).

Two new non-binary matrices are developed in this study, one a mirror of the other. Initially based on the fuzzification of Wang, Lan et als' space interference matrix which considers remote relations[1, 40]. The *Feasibility* matrix provides a value for each components' direction of movement relative to each other. Scoring from 0 (entirely free to move) to 100 (direct contact/impossible). The *Confidence* matrix evaluates 'how guaranteed is this this move?' from 0 (Absolutely confident) to 0.06 (Somewhat uncertain). Table 1 indicates the corresponding scoring systems. It is worth mentioning that the scores provided are subjective, based purely on intuition, and can be changed to the user's preference. Secondly, they are initial conditions only, and will be subject to change throughout the Algorithm.

Table 1. Text in red indicates scores used in Examples 2-3

Feasibility & Confidence scoring metric

Removal Relationship	Contact	Feasibility	Confidence
Free to move	0	0	0
Thread	0	0.3	0.04
Perpendicular	0	0.01	0.01
Parrallel	0	0.2	0.04 | 0.05
Remote	0	0.5 | 0.4	0.06 | 0.03
Direct/Impossible	1	100	0

These initial conditions still effect DSP generation and sequence ranking greatly, and have been assumed as follows: Perpendicular interference, i.e. Pulling a component directly away from another has the lowest feasibility and confidence score, both 0.01 respectively. In theory this move has the greatest probability of success for two contacting components even though rust and temperature induced adhesion remain a possibility. Parallel removal is considered slightly less feasible (0.2) and more uncertain (0.04) due to increased surface friction and prolonged contact time during removal. Similarly, threads are attributed 0.3 feasibility and 0.04 confidence respectively. An impossible move scores 100 feasibility and 0 confidence as we can be certain that two physical objects cannot move through each other. Finally, remote interference refers to a future contact in a given direction and is considered somewhat feasible (0.5) but very uncertain (0.06). It is important to remember that although counter-intuitive, scoring low in both Feasibility and Confidence indicates a high probability of removal.

For the purposes of this experiment a few scoring intricacies must be detailed:

1. Removing a bulky component from a thread is categorised as impossible.
2. Removing a washer or spacer from a bolt is classified as *parallel* interference, and vice versa, whereas, removing a nut from a bolt is assigned *thread* interference.
3. If a component shares both parallel and perpendicular interference, such as C4 and C3 in Figure 1, the greater score is assumed.
4. Finally, if a part has multiple points of contact, as C1 and C5 share two, for simplicity this is modelled as a single contact.

Using Example 1 (Fig.1), the feasibility score for C1 in the X+ direction can be calculated as the sum of C1's feasibility when moved in the X+ direction relative to each component. The following matrix indicates this compression:

$$F_{source} = \begin{bmatrix} 0 & 0 & 0 & 0 & F_{C1x+}^{C2} & F_{C1x-}^{C2} & F_{C1y+}^{C2} & F_{C1y-}^{C2} & \cdots & F_{C1x+}^{f4} & F_{C1x-}^{f4} & F_{C1y+}^{f4} & F_{C1y-}^{f4} \\ F_{C2x+}^{C1} & F_{C2x-}^{C1} & F_{C2y+}^{C1} & F_{C2y-}^{C1} & 0 & 0 & 0 & 0 & \cdots & F_{C2x+}^{f4} & F_{C2x-}^{f4} & F_{C2y+}^{f4} & F_{C2y-}^{f4} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ F_{f4x+}^{C1} & F_{f4x-}^{C1} & F_{f4y+}^{C1} & F_{f4y-}^{C1} & F_{f4x+}^{C2} & F_{f4x-}^{C2} & F_{f4y+}^{C2} & F_{f4y-}^{C2} & \cdots & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$F_{C1X+} = \sum 0 + F_{C1x+}^{C2} + F_{C1x+}^{C3} + F_{C1x+}^{C4} + F_{C1x+}^{C5} + F_{C1x+}^{f1} + F_{C1x+}^{f2} + F_{C1x+}^{f3} + F_{C1x+}^{f4} \qquad \textbf{(1)}$$

$$F_{C1X+} = \sum (0 + 0.5 + 0.5 + 0.5 + 0.2 + 100 + 0.5 + 0.5 + 100) = 202.7$$

Where F_{source} is the uncompressed Feasibility matrix (shown in Table 2B), F_{C1x+}^{C2} represents the feasibility interference between C1 and C2 when moving C1 in the X+ direction, and F_{C1X+} is the feasibility score moving of C1 in the X+ direction at a given moment.

The same summation method is used to compress the Confidence matrix. The full *uncompressed* and more digestible *compressed* matrices can be seen below in Tables 2(A-F). Likewise, the Python code created for the compression function is available in Appendix 6: *Pseudocode/Python Appendix*.

Tables 2(A-F) Uncompressed C, F & U, and corresponding Compressed R, F & U matrices

A) Contact Matrix

	C1.X	C1-X	C1Y	C1-Y	C2X	C2-X	C2Y	C2-Y	C3X	C3-X	C3Y	C3-Y	C4X	C4-X	C4Y	C4-Y	C5X	C5-X	C5Y	C5-Y	f1X	f1-X	f1Y	f1-Y	f2X	f2-X	f2Y	f2-Y	f3X	f3-X	f3Y	f3-Y	f4X	f4-X	f4Y	f4-Y
C1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
C2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
C3	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
C4	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
C5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
f1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f2	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f3	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f4	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

B) Feasibility Matrix

	C1.X	C1-X	C1Y	C1-Y	C2X	C2-X	C2Y	C2-Y	C3X	C3-X	C3Y	C3-Y	C4X	C4-X	C4Y	C4-Y	C5X	C5-X	C5Y	C5-Y	f1X	f1-X	f1Y	f1-Y	f2X	f2-X	f2Y	f2-Y	f3X	f3-X	f3Y	f3-Y	f4X	f4-X	f4Y	f4-Y
C1	0	0	0	0	0.5	0.5	0.01	100	0.5	0.5	0	0.5	0.5	0.5	0	0.5	0.2	0.2	0.01	100	100	100	100	100	0.5	0.5	0	0.5	0.5	0.5	0	0.5	100	100	100	100
C2	0.5	0.5	100	0.01	0	0	0	0	0	0	0	0.5	0	0	0	0.5	0	0	0	0.5	100	100	100	100	0	0	0	0.5	0	0	0	0.5	0	0	0	0
C3	0.5	0.5	100	0	0	0	0.5	0	0	0	0	0	100	100	0.2	100	0	0	0.5	0	0	0	0.5	0	100	100	100	100	0	0	0	0.5	0.5	0	0	0
C4	0.5	0.5	0.5	0	0	0	0.5	0	100	100	100	0.2	0	0	0	0	0.2	0.2	0.01	100	0	0	0.5	0	0	0.5	0	0	100	100	100	0.5	0	0	0	0
C5	0.2	0.2	100	0.01	0	0	0.5	0	0	0	0.5	0	0.2	0.2	100	0.01	0	0	0	0	0	0	0.5	0	100	100	100	100	100	100	100	100	100	100	100	100
f1	100	100	100	0.3	100	100	100	0.3	0	0	0	0.5	0	0	0	0.5	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0
f2	0.5	0.5	0.5	0	0	0	0	0	100	100	0.3	100	0.5	0	0	0	100	100	0.3	100	0	0	0	0	0	0	0	0	0.5	0	0	0	0.5	0	0	0
f3	0.5	0.5	0.5	0	0	0	0.5	0	0	0	0.5	0	100	100	100	0.3	100	100	100	0.3	0	0	0.5	0	0	0.5	0	0	0	0	0	0	0.5	0	0	0
f4	100	100	100	0.3	0	0	0	0	0	0	0	0	0	0	0	0	100	100	100	0.3	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0

C) Confidence Matrix

	C1.X	C1-X	C1Y	C1-Y	C2X	C2-X	C2Y	C2-Y	C3X	C3-X	C3Y	C3-Y	C4X	C4-X	C4Y	C4-Y	C5X	C5-X	C5Y	C5-Y	f1X	f1-X	f1Y	f1-Y	f2X	f2-X	f2Y	f2-Y	f3X	f3-X	f3Y	f3-Y	f4X	f4-X	f4Y	f4-Y
C1	0.00	0.00	0.00	0.00	0.06	0.06	0.01	0.00	0.06	0.06	0.00	0.06	0.06	0.06	0.00	0.06	0.04	0.04	0.01	0.00	0.00	0.00	0.00	0.00	0.06	0.06	0.00	0.00	0.06	0.06	0.00	0.00	0.00	0.00	0.00	0.00
C2	0.06	0.06	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00
C3	0.06	0.06	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.06	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C4	0.06	0.06	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C5	0.04	0.04	0.00	0.01	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.04	0.04	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f1	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f2	0.06	0.06	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f3	0.06	0.06	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f4	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

D) Relation matrix

	C1	C2	C3	C4	C5	f1	f2	f3	f4
C1	0	1	0	0	0	1	0	0	0
C2	1	0	0	0	0	1	0	0	0
C3	0	0	0	1	0	0	0	0	0
C4	0	0	1	0	0	0	0	0	0
C5	0	0	0	0	0	0	0	0	0
f1	1	1	0	0	0	0	0	0	0
f2	0	0	0	0	0	0	0	0	0
f3	0	0	0	0	0	0	0	0	0
f4	0	0	0	0	0	0	0	0	0

E) Feasibility matrix

	x+	x-	y+	y-
C1	202.7	202.7	200.02	402
C2	100.5	100.5	200	102.51
C3	201	200.5	201.2	201
C4	201.2	201.2	201.51	200.2
C5	300.4	300.4	501.5	300.02
f1	200	200	200	2.6
f2	202	200.5	1.6	200
f3	201	201	202	0.6
f4	200	202	200	0.6

F) Confidence matrix

	x+	x-	y+	y-
C1	0.34	0.34	0.02	0.24
C2	0.06	0.06	0	0.31
C3	0.12	0.06	0.2	0.12
C4	0.16	0.16	0.23	0.04
C5	0.08	0.08	0.18	0.02
f1	0	0	0	0.32
f2	0.24	0.06	0.2	0
f3	0.12	0.12	0.24	0.08
f4	0	0.24	0	0.08

Remember that the Relation matrix functions like an OR gate, whereas, the Feasibility and Confidence matrices total a components' directional relationships. The initial scores were devised to allow the program to determine between feasible and impossible moves, i.e. if a component were to score over 100 in the feasibility matrix, it can be considered unfeasible or interlocked at that moment. Defining these extremities will allow sufficient room for the program to influence DSP's in the future, while avoiding impossible recommendations.

2.1 Monte-Carlo propagation of uncertainty

The decision on which component to remove is based on the *Summation* matrix. This matrix is the sum of feasibility, the mean of the 10000 times the normal distribution of confidence, and two times the standard deviation. This calculation applies a tolerance to the initial feasibility scores. Two times the standard deviation is used to simulate a 95% confidence interval.

$$S = \begin{bmatrix} F_{C1X+} & F_{C1X-} & F_{C1Y+} & F_{C1Y-} \\ F_{C2X+} & F_{C2X-} & F_{C2Y+} & F_{C2Y-} \\ \vdots & \vdots & \vdots & \vdots \\ F_{f4X+} & F_{f4X-} & F_{f4Y+} & F_{f4Y-} \end{bmatrix} + \begin{bmatrix} \mu_{C1X+} & \mu_{C1X-} & \mu_{C1Y+} & \mu_{C1Y-} \\ \mu_{C2X+} & \mu_{C2X-} & \mu_{C2Y+} & \mu_{C2Y-} \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{f4X+} & \mu_{f4X-} & \mu_{f4Y+} & \mu_{f4Y+} \end{bmatrix} + 2 \begin{bmatrix} \sigma_{C1X+} & \sigma_{C1X-} & \sigma_{C1Y+} & \sigma_{C1Y-} \\ \sigma_{C2X+} & \sigma_{C2X-} & \sigma_{C2Y+} & \sigma_{C2Y-} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{f4X+} & \sigma_{f4X-} & \sigma_{f4Y+} & \sigma_{f4Y-} \end{bmatrix} \quad \textbf{(2)}$$

Where S is the Summation matrix and F_{C1X+}, μ_{C1X+} & σ_{C1X+} represent the directional feasibility, the mean of the normal distribution of directional confidence and the standard deviation respectively. This summation is conducted before each removal decision and as the confidence matrix is updated every iteration, this uncertainty propagation can prove decisive. Tables 3[a-c] illustrate the first iteration of cycle one, in which F4 is chosen for removal in the Y- direction.

Tables 3(A-C). The compressed uncertainty (Confidence) propagating matrices & the total Summation matrix

A) Mean of normal distribution

	x+	x-	y+	y-
C1	0.0038	0.0079	0.0006	-0.0048
C2	-0.0028	0.001	0.0014	0.0021
C3	0.003	-0.0013	-0.0009	-0.002
C4	-0.0058	0.0005	0.005	-0.0006
C5	0.002	0.0018	0.0002	0.0006
f1	0	0	0	0.0029
f2	0.0022	0.001	-0.0012	0
f3	0.0022	0.0036	0.0019	0.0001
f4	0	-0.0028	0	-0.001

B) 2x Standard deviation

	x+	x-	y+	y-
C1	0.69	0.68	0.04	0.47
C2	0.12	0.12	0.24	0.38
C3	0.24	0.12	0.4	0.24
C4	0.32	0.31	0.45	0.08
C5	0.16	0.16	0.36	0.04
f1	0	0	0	0.64
f2	0.47	0.12	0.4	0
f3	0.24	0.24	0.48	0.16
f4	0	0.49	0	0.16

C) Summation matrix

	x+	x-	y+	y-
C1	203.39	203.39	200.06	402.47
C2	100.62	100.62	201.24	101.89
C3	201.25	200.61	201.59	201.24
C4	201.51	201.51	201.97	200.28
C5	300.56	300.56	501.86	300.06
f1	200	200	200	3.24
f2	202.47	200.62	2	200
f3	201.24	201.25	202.48	0.76
f4	200	202.49	200	0.76

3 On-line Disassembly Re-planning Algorithm

Re-planning means that upon failure the algorithm must adapt and learn. Thus, a new component must be suggested for removal and relevant matrices updated accordingly. This is achieved through a number of techniques including a penalty value matrix, a recursion stopper, and constant confidence manipulation. The control flow chart shown in Figure 2 briefly describes the *iterative* re-planning method, plus the full *cycle* output and system feedback. A full description follows:

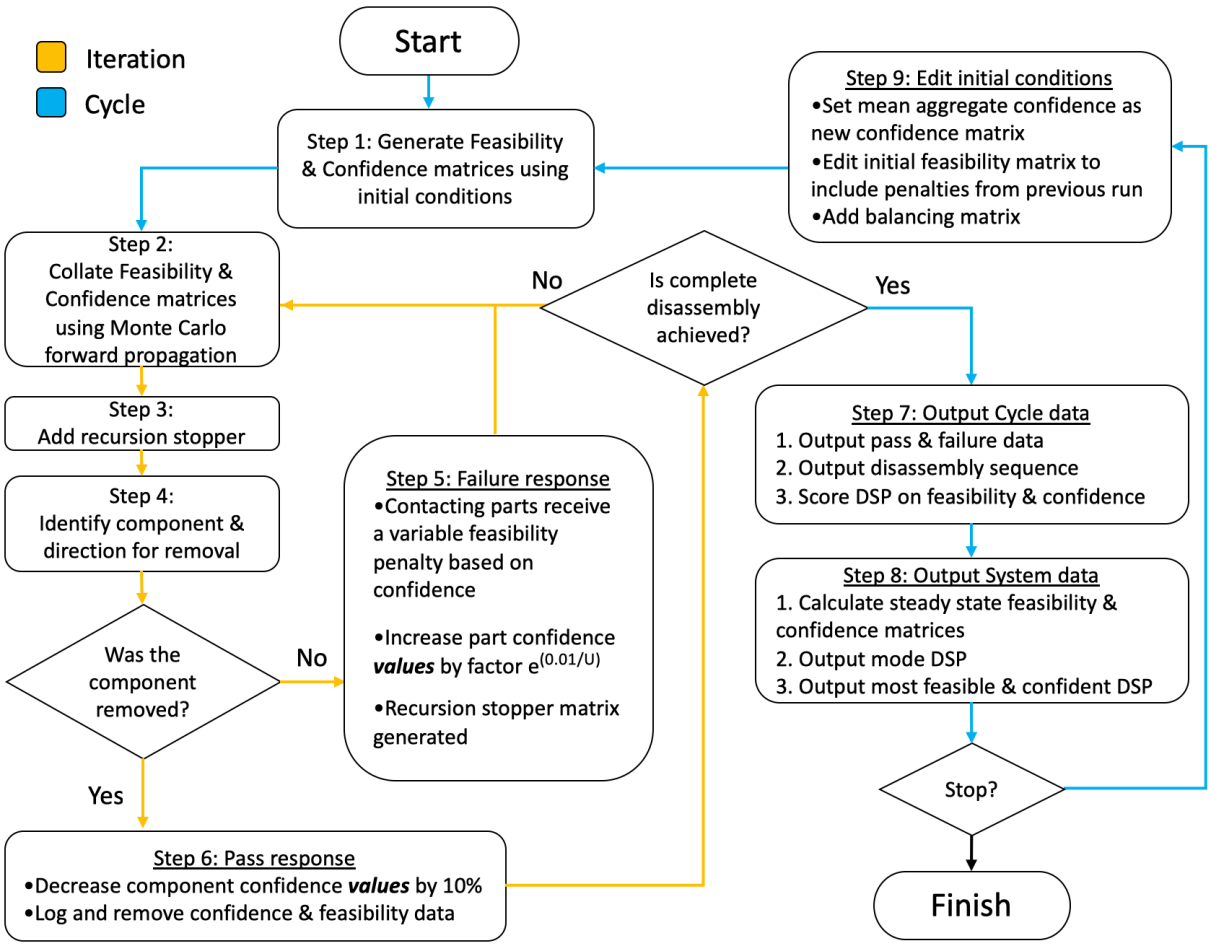

Figure 2. The Algorithm flow chart

The complete Algorithm for 3D assemblies can be seen in Appendix 7: *Full code of Example 3*.

Steps 1-4 (Section 2): The compressed Feasibility, Confidence and Relation matrices are generated. As previously stated, feasibility, the mean of the normal distribution of confidence and two times standard deviation is summed and compressed. If a new component is not selected for removal a recursion stopper is added (Section 3.2). The component with lowest summation score is chosen for removal.

Step 5 - Failure Response (Section 3): Upon failure, all contacting parts receive a variable feasibility penalty based on the uncompressed confidence matrix and a fixed value, *penalty cost.* The components' confidence *values* are then increased by a factor $e^{(0.01/U)}$ – increasing uncertainty. Lastly, if the penalty is less than the difference to prevent component re-suggestion, a recursion stopper matrix is generated and stored to provide a temporary penalty next iteration.

Step 6 - Pass Response (Section 3): Upon successful removal of a given component, respective confidence *values are* decreased by 10%. The components feasibility and confidence data are stripped

from the source data and stored in corresponding *Aggregate* matrices. Likewise, the components feasibility, confidence and their product at that moment are kept as *scores* for later evaluation of sequences.

Step 7 – Cycle Data (Section 4): The cycle completes and the Algorithm outputs pass/failure data as well as the achieved DSP. Total scores are calculated for the completed sequence.

Step 8 – System Data (Section 4): The system outputs mean aggregate feasibility and confidence. The steady state matrices display the effect of all past cycles. A dictionary of sequences is created to find the mode DSP. The scores are then used to locate the *most feasible*, *most certain* and *best* overall DSP.

Step 9 – Edit Initial Conditions (Section 4): The Algorithm runs until the number of cycles equals the *run_size* criterion. Else initial confidence is set to equal mean aggregate confidence; the previous cycles penalties are applied to the stored Feasibility matrix; and a balancer is subtracted to allow for iterative estimation, stopping the new Feasibility matrix from converging. The new Feasibility matrix is then stored, ready for the next cycle.

3.1 Penalty Matrix

It is reasonable to assume that upon failure a part will grow in uncertainty and become less feasible; the same is somewhat true for neighbouring components. This being the case, variable penalties are applied to all contacting parts which hold uncertainty (or confidence values). The coordinates of these parts are taken from the relation matrix.

The penalties are applied in the form of a *penalty matrix* which is determined as the product of *contacting* parts *uncompressed confidence values* and a *fixed penalty cost*. Where, confidence values can be seen like weights in a neural network - the dynamic properties of which will be described in section 3.4. This penalty matrix is then applied to the *uncompressed feasibility matrix* before the next iterations compression, summation and removal decision. The python code can be seen in Figure 3.

```
[j] = np.where(r[x] == 1) #j is a list of row co-ordinates where 1's exist in Relation matrix
count_components = 0 #counts contacting components
coords = [] #co-ordinates of contacting relations
for z in j:
        count_components += 1
        coords.append([int(x),z])
        coords.append([z,int(x)])

penalty_matrix = np.zeros((run,rise)) #run-rise is the row-columns of source data
for [a,b] in coords:
        penalty_matrix[a,(b*d_s):(b*d_s+d_s)] += u[a,(b*d_s):(b*d_s+d_s)]
        #duplicating contacting parts confidence data (in all directions)
        penalty_matrix = penalty_matrix*penalty_cost

comp_p_matrix = compress(penalty_matrix)
f += penalty_matrix #penalty matrix applied to uncompressed feasibility
agg_penalty += penalty_matrix #storing all penalties to add to initial F in next cycle
```

Figure 3. Python code for penalty matrix application

In the example given below (Tables 4. A-C), the penalty cost is 50, although, this should be tweaked for differing product and run complexity. Here, $C1$ is chosen for removal in the $Y+$ direction and fails. As $C1$, $C2$ and $C5$ are the only contacting components they all receive penalties.

The penalties are greater in the X axis as opposed to the Y due to the greater initial uncertainty associated with remote and parallel interference (0.06, 0.04) than perpendicular (0.01). As C1 is contacting two components its penalties are equivalent to the total shared penalties. It is important to note that this example is taken from iteration 5 in cycle 1, and that throughout the program relevant confidence matrices will undergo significant change, thus, determining the weight of these penalties.

Tables 4(A-C). Iteration 5, Cycle 1, Failure of C1: Summation matrix(A), Penalty matrix(B) & Recursion stopper(C)

Summation after [f4 & f1] removed

	x+	x-	y+	y-
C1	3.4	3.4	0.1	202.5
C2	0.6	0.7	100.0	3.4
C3	200.6	200.6	201.8	201.2
C4	200.9	201.5	203.2	200.3
C5	200.6	200.6	401.2	200.1
f1	0.0	0.0	0.0	0.0
f2	201.9	200.6	3.6	200.0
f3	200.6	201.3	201.9	4.5
f4	0.0	0.0	0.0	0.0

Compressed Penalty

	x+	x-	y+	y-
C1	2.0	2.0	0.4	0.0
C2	1.2	1.2	0.0	0.2
C3	0.0	0.0	0.0	0.0
C4	0.0	0.0	0.0	0.0
C5	0.8	0.8	0.0	0.2
f1	0.0	0.0	0.0	0.0
f2	0.0	0.0	0.0	0.0
f3	0.0	0.0	0.0	0.0
f4	0.0	0.0	0.0	0.0

Recursion Stopper

	x+	x-	y+	y-
C1	500	500	500	500
C2	0.0	0.0	0.0	0.0
C3	0.0	0.0	0.0	0.0
C4	0.0	0.0	0.0	0.0
C5	0.0	0.0	0.0	0.0
f1	0.0	0.0	0.0	0.0
f2	0.0	0.0	0.0	0.0
f3	0.0	0.0	0.0	0.0
f4	0.0	0.0	0.0	0.0

3.2 Recursion Stopper

As part of the re-planning strategy, it is paramount the program offers a new component for removal. If the penalty is not great enough to achieve this, a recursion stopper is used. This temporary penalty resets to zero each iteration.

In the example given above, C1, C2, f2 and f3 can be considered the only feasible removal options. The program attempts to move C1 in the Y+ direction (0.1) and fails. The respective penalties are applied to C1, C2 and C5, yet they alone will not inhibit the program from re-suggesting C1. As such, the recursion stopper applies an arbitrary value of 500 to the entire row of C1, temporarily increasing C1's next iteration summation score, and consequently, C2 in the X+ will be selected for removal. This temporary stopper is applied to the whole row to prevent a rare instance in which the penalties result in the selection of the same component but in a different direction.

This does mean that a repetitive loop can occur between two parts, but only if no other parts are *feasible*, and in this case, the assembly can be considered interlocked. The Pseudocode for the recursion stopper can be found in Appendix 6: *Pseudocode/Python Appendix*.

3.3 Training data

For the purpose of this experiment training data has been managed as a set of probability ranges or *Chance of Failure* (CoF). This will allow us to verify if the output is coherent with the input. Posing the question, are the steady state feasibility and confidence matrices rational after 10000+ iterations? Secondly, preparing the data in this manner allows easy adaptation for testing, rather than creating a finite list of instructions for each experiment. Lastly, using stochastic data will generate a degree of noise which would be expected in situ. These CoF ranges will be illustrated in the upcoming examples. The relevant pseudocode for the application of stochastic training data can be seen in Figure 4.

```
component = ["C1","C2","C3","C4","C5","f1","f2","f3","f4"]
prob_array_min = [0.08, 0.08, 0.08, 0.08, 0.12, 0.08, 0.12, 0.12, 0.08]
prob_array_max = [0.12, 0.12, 0.12, 0.12, 0.20, 0.12, 0.20, 0.20, 0.12]

prob = choose a number between prob_array_min & prob_array_max
generate a random number between prob_array_min & prob_array_max
decision = rand_dec > prob

if decision is False: #Fail
if decision is True: #Pass
```

Figure 4. Pseudocode for application of stochastic training data

3.4 Dynamic Uncertainty

The crucial element in adapting to uncertainties is how the confidence is manipulated at each decision. The following pass/fail configuration will determine the stability and plasticity of our system. For this study we plan to iterate thousands of times over, and that being the case, a relatively stable manipulation mechanism has been designed. The graph below presents the percentage change in confidence for both passes and failures.

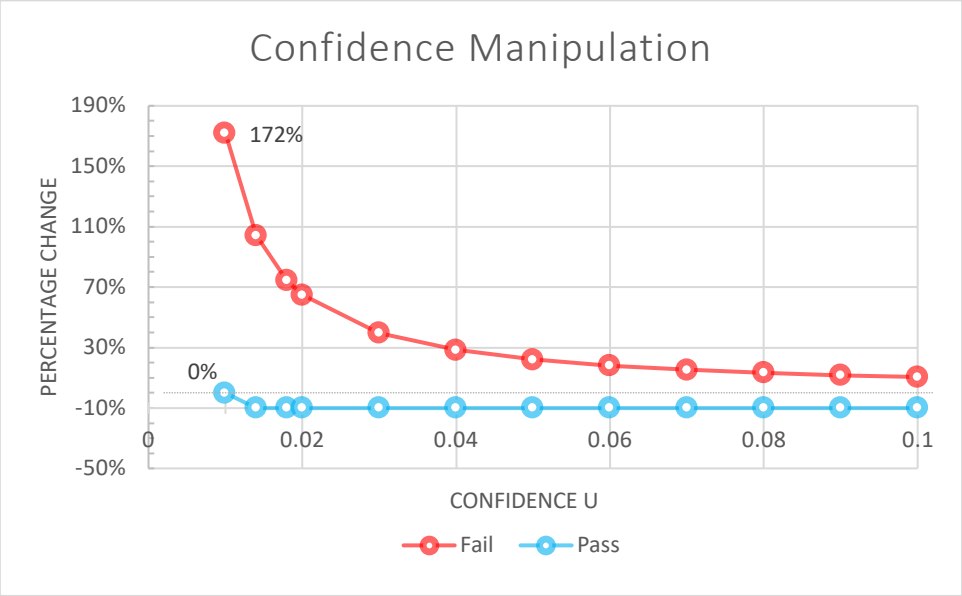

Figure 5. The effect of Passes and Failures on confidence values (U)

A factor of $e^{(\frac{0.01}{U})}$ is used to increase the confidence value upon failure. This punishes components which have a low starting confidence value, i.e. If a component which we believe to be *certain* fails, it is severely punished. This means that if a part were to fail consecutively, significant penalties are accumulated based on the increased uncertainty. This exponential curve provides the system with some agility. Here, U_{+1} denotes next iterations confidence value.

$$U_{+1} = U \times e^{\left(\frac{0.01}{U}\right)} \quad \textbf{(3)}$$

Conversely, if a component which is believed to be *uncertain* fails, the confidence is increased almost linearly. The component was expected to fail, and it did, so the initial prediction was relatively suitable. As we expect to witness passes several times more often than failures, a passing component receives a statutory 10% reduction in confidence value. Confidence values will never drop below 0.01, with the exception of *absolutely certain* parts which remain at confidence 0 for the entire program. With this stable system in place we can expect most confidence values to remain within a 0.01-0.05 range.

A general point of interest is that a more aggressive manipulation method could benefit online decision making by offering increased agility. Furthermore, the Algorithm discussed can be easily adapted for selective or partial disassembly, providing additional value in real time batch disassembly. Nevertheless, we are evaluating the program as an analytical tool, cycling thousands of times over, to determine an optimum DSP and steady state matrices.

4 Iterative Estimation and Ranking systems

There are two feedback loops within the program. The first determines passes and failures, defining each sequence while creating penalties and updating the Confidence matrix. The second extrapolates the final data and feeds it back into the first loop through both Feasibility and Confidence matrices. This section will describe this final data, how it is calculated, and fed back into the system.

4.1 Steady state Feasibility/Confidence

After simulating thousands of cycles steady state F and U (feasibility & confidence) matrices are acquired. This is achieved by storing a removed components data in Aggregate matrices. Then calculating the mean. Thus, at the end of the program two new matrices will more accurately depict product condition. Tables 5(A-C) present the steady state outputs for example 1 under *Training data 1a*. Initial F and U matrices are available for comparison in Appendix 3a: *Example 1a.*

Tables 5(A-C). Example 1a A) Training Data and B&C) Steady State Matrices

A) Training data 1a

	CoF	Actual fails
C1	8-12%	69
C2	8-12%	72
C3	8-12%	100
C4	8-12%	120
C5	8-12%	106
f1	9-15%	164
f2	8-12%	125
f3	8-12%	129
f4	8-12%	130

B) Compressed mean F 1a

	x+	x-	y+	y-
C1	239	239	206	381
C2	111	111	200	94
C3	180	180	229	180
C4	207	207	219	203
C5	332	332	480	310
f1	200	200	200	58
f2	181	180	30	200
f3	180	180	181	27
f4	200	181	200	36

C) Compressed mean U 1a

	x+	x-	y+	y-
C1	0.30	0.30	0.03	0.24
C2	0.03	0.03	0.00	0.31
C3	0.12	0.06	0.17	0.12
C4	0.15	0.15	0.22	0.02
C5	0.05	0.05	0.18	0.03
f1	0.00	0.00	0.00	0.30
f2	0.24	0.06	0.17	0.00
f3	0.12	0.12	0.24	0.05
f4	0.00	0.24	0.00	0.05

Once a cycle has finished, the data is fed back into the system in two ways:

1. The new confidence is set equal to *mean aggregate confidence*, resulting in an almost fixed U matrix being formed over time (Equation 4). Interestingly, if the Confidence matrix is given the freedom to maintain its values from one cycle to the next, an extremely dynamic system would be created for real time, in-situ use.

$$U_{new} = \frac{Total\_aggregate\_U}{Run\_size} \qquad \textbf{(4)}$$

Where,

U_{new}	New Confidence matrix for current cycle use
$Total\_aggregate\_U$	Sum of all Aggregate Confidence matrices
Run_size	The number of cycles completed

2. The new Feasibility matrix is updated to accommodate last iterations penalties. Then a *Balancer matrix,* which is equal to the sum of these penalties is subtracted. This prevents the matrix from converging. Please note that both the penalties and the Balancer matrix will *only* affect the directional feasibility of components with *contacting interference* (parallel, perpendicular or thread interference). The method is explained as follows:

$$F_{new} = (F_{store} + P_{aggregate}) - \left(\frac{\sum P_{aggregate}}{N_{U-contacts}} \begin{bmatrix} \frac{U^{c1}_{c1x+}}{U^{c1}_{c1x+}} & \frac{U^{c1}_{c1x-}}{U^{c1}_{c1x-}} & \cdots & \frac{U^{f4}_{c2x-}}{U^{f4}_{c1x-}} \\ \frac{U^{c1}_{c2x+}}{U^{c1}_{c2x+}} & \frac{U^{c1}_{c2x-}}{U^{c1}_{c2x-}} & \cdots & \frac{U^{f4}_{c2y-}}{U^{f4}_{c2y-}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{U^{c1}_{f4x+}}{U^{c1}_{f4x+}} & \frac{U^{c1}_{f4x+}}{U^{c1}_{f4x+}} & \cdots & \frac{U^{f4}_{f4x+}}{U^{f4}_{f4x+}} \end{bmatrix} \right) \quad \textbf{(5)}$$

Where $N_{U-contacts}$ is the number of contacting components with parallel, perpendicular or thread interference. This is multiplied by a *binary matrix* representing the location of contacts. The $P_{aggregate}$ matrix is set to zero after this calculation, and F_{new} becomes equal to F_{store} for next cycles use.

4.2 Recommended disassembly sequence plans

Continuing on with example 1a, the program was run for 1000 cycles with an approximate pass/fail ratio of 8.5:1. Outputs include a dictionary of sequences; a dictionary of directional sequences; a dictionary of all failed components; and a full timeline of failure events. All outputs and conditions for every example can be seen in Appendices 3-5. For Example 1a, the following DSP is recommended:

f4	f2	f3	C5	C4	C3	f1	C1	C2
Y-	Y+	Y-	Y-	Y-	Y-	Y-	Y+	Y-

For a product with less than 10 parts, the mode sequence over a significant number of cycles can be determined as the systems overall DSP recommendation. Given f1's failure rate and relationships with C1 and C2, the proposed DSP method is effective at adapting to this uncertainty by choosing to disassemble the bottom half of the product first, before dealing with f1, C2, and C1. Furthermore, the direction of disassembly is predominantly in the Y-, again a reasonable suggestion considering the internal elements are blocked by C1.

For a product of complexity greater than 10 parts, the *sequence ranking scheme* should be used to evaluate sequences. Here, *directional* sequences are ranked on initial feasibility, confidence, and their product upon removal respectively. The final results output the top 3 sequences in each bracket.

```
if decision is True: #Pass
        f_score += fi[thex, they]
    #Initial compressed F matrix with parts removed, not a dynamic matrix, but generates very feasible sequences
        u_score += s2_2[thex, they]
    #Scoring 2*standard deviation of mean compressed Confidence (dynamic & creative)
        fxu_score += (fi[thex, they])*(s2_2[thex, they])
    #the product of what is feasible and creative, these scores are then totalled at the end of the Cycle and logged
```

Figure 6. Python code for the calculation of directional removal scores

5 Experiments and Discussion

To verify the performance of the proposed re-planning, dynamic uncertainty method, three products varying in set up and complexity are considered. Firstly example 1 is re-visited with new training data for performance validation, then products B and C (Seen below) will be considered in six directions i.e. (X+, X-, Y+, Y-, Z+, Z-).

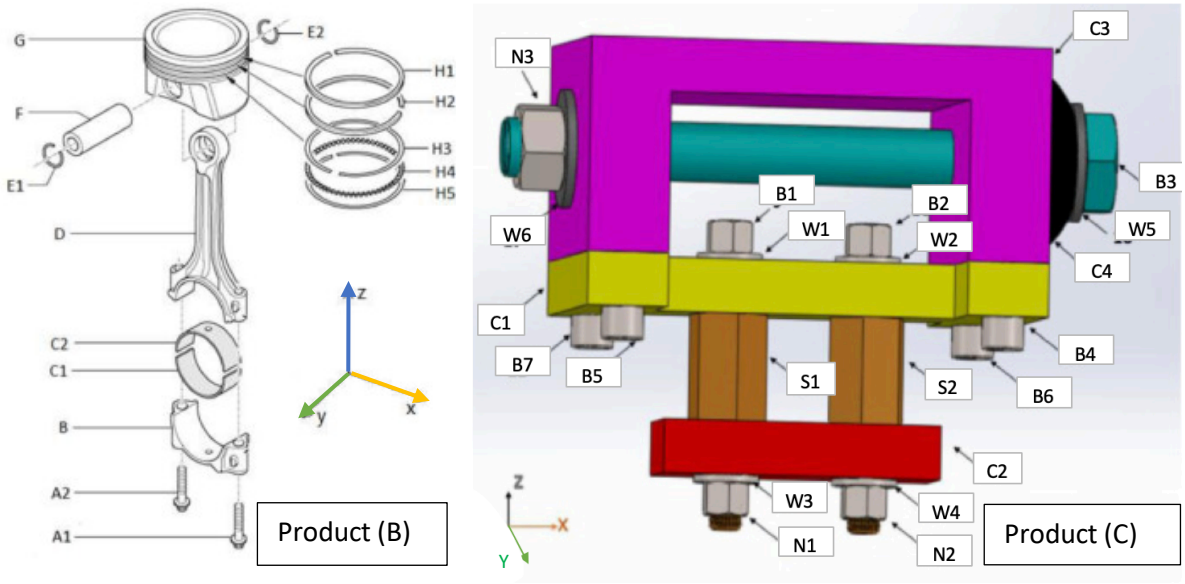

Figure 7. Two typical products applied to validate the method proposed in this study.

Products B and C are taken directly from Laili, Tao et al's. paper which allows for real-time optimisation by considering bees algorithm in combination with the two pointer-method[31].

5.1 Example 1b

To assess the plasticity of the system the training data was altered to increase frequency of failures within the bottom half of the assembly. F2 and F3's chance of failure (CoF) have been significantly increased. Here we would expect the system to adapt by dismantling the top half first. The overall results can be seen below, and a full list of all outputs can be seen in Appendix 3b: *Example1b*.

Tables 6(A-C). Example 1b B&C) Feasibility & Confidence matrices under A) stated training conditions

A) Training data 1b

	CoF	Actual fails
C1	8-12%	91
C2	8-12%	124
C3	8-12%	82
C4	8-12%	61
C5	8-12%	110
f1	8-12%	95
f2	12-20%	197
f3	12-20%	189
f4	8-12%	99

B) Compressed mean F 1b

	x+	x-	y+	y-
C1	242	242	205	377
C2	113	113	200	92
C3	176	176	239	176
C4	213	213	238	197
C5	339	339	476	313
f1	200	200	200	26
f2	177	176	60	200
f3	176	176	177	55
f4	200	177	200	23

C) Compressed mean U 1b

	x+	x-	y+	y-
C1	0.30	0.30	0.03	0.24
C2	0.03	0.03	0.00	0.31
C3	0.12	0.06	0.17	0.12
C4	0.15	0.15	0.22	0.02
C5	0.06	0.06	0.18	0.04
f1	0.00	0.00	0.00	0.30
f2	0.24	0.06	0.17	0.00
f3	0.12	0.12	0.24	0.05
f4	0.00	0.24	0.00	0.05

f4	f1	C2	C1	f3	f2	C3	C5	C4
Y-	Y-	Y-	Y+	Y-	Y+	Y+	Y-	X+

The sequence above demonstrates that the system has successfully adapted to the changes in product condition by dismantling the top half first. It is important to note that no run is the same, individual

cycles can vary greatly due to noise in the training data, and in many cases the program can be *creative*.

This mode sequence was performed 14.4% of the time, a substantial amount considering that any of the 1,048 failures could change the order at any interval. The next most common sequence occurred 90 times (9%) with the sole difference being that C4 was removed before C5. Hence, we can more or less conclude that a new and more efficient DSP was constructed based on the knowledge of past failures. Moreover, two new steady state matrices reflect this knowledge and enforce a DSP which prioritises certainty.

Interpreting the steady state Confidence matrix can be difficult because of remote interference. The large uncertainty value adds aesthetic noise to the matrix, as upon failure, only contacting relations receive a penalty. Thus, remote interference has a rare effect on penalties (A few instances where components share contact and remote interference, where remote is the greatest interference) yet it is still vital to the initial conditions. In any case, for easier interpretation, future examples will exclude remote interference from the presented data *only*.

5.2 The Piston Example 2a

Now let us consider a 3D example in 6 directions. For product B, when set up with somewhat balanced training conditions, i.e. CoF ranges between 8-20% for each component. The following sequence is observed:

*Tables 7(A-C). Example 2a results (no remote interference in **presented** confidence data **only**)*

A) Training 2a

C	CoF	F
A1	8-20%	351
A2	8-20%	291
B	8-20%	331
C1	8-20%	211
C2	8-20%	172
D	8-20%	272
E1	8-20%	305
E2	8-20%	315
F	8-20%	308
G	8-20%	203
H1	8-20%	296
H2	8-20%	288
H3	8-20%	297
H4	8-20%	307
H5	8-20%	231

B) Compressed mean feasibility 2a

	x+	x-	y+	y-	z+	z-
A1	200	200	200	200	200	12
A2	200	200	200	200	200	7
B	305	305	305	305	400	160
C1	114	114	114	114	84	88
C2	114	114	114	114	88	78
D	400	400	306	306	278	400
E1	200	200	-41	200	200	200
E2	200	200	200	-42	200	200
F	400	400	111	90	400	400
G	800	800	605	585	800	329
H1	100	100	100	100	2	100
H2	100	100	100	100	1	100
H3	100	100	100	100	5	100
H4	100	100	100	100	11	100
H5	100	100	100	100	10	100

C) Comp mean con 2a (no remote)

	x+	x-	y+	y-	z+	z-
A1	0	0	0	0	0	0.054
A2	0	0	0	0	0	0.051
B	0.034	0.034	0.034	0.034	0	0.032
C1	0.034	0.034	0.034	0.034	0.016	0.02
C2	0.034	0.034	0.034	0.034	0.02	0.014
D	0	0	0.031	0.031	0.014	0
E1	0	0	0.034	0	0	0
E2	0	0	0	0.028	0	0
F	0	0	0.06	0.077	0	0
G	0	0	0.029	0.046	0	0.149
H1	0	0	0	0	0.029	0
H2	0	0	0	0	0.029	0
H3	0	0	0	0	0.029	0
H4	0	0	0	0	0.029	0
H5	0	0	0	0	0.031	0

E1	E2	H3	H2	H4	H5	H1	A2	A1	B	C1	C2	D	F	G
Y+	Y-	Z+	Z+	Z+	Z+	Z+	Z-	Z-	Z-	Z-	Z-	Y+	Y+	X+

This is a reasonable and valid *mode* DSP. One critique would be that H1-H5 could be removed in a more *sensible* order, however, the program is removing them in order of how *guaranteed* it believes their removal to be.

Given what was intended to be a *balanced* set up actually returned a mixed bag. The program adapted logically and legally, returning a feasible DSP which again prioritised components it felt it could rely on. Interestingly E1 and E2 have high failure rates of 300+ yet their feasibility scores are drastically low. The parts only share one confidence score in the Y axis (perpendicular interference, 0.01) with their two contacts, F and G, perhaps this is the reasoning behind their score, which even though extremely feasible, is not necessarily bad considering they are key component for further disassembly.

This valid DSP was performed 86 times out of a total 2000 sequences. This variance in output is understandable due to the increase in product size and the wide range of failures from component to component (172 – 351). Looking at the data from the sequence ranking system it can be deducted that the program had reached a steady state or favoured DSP.

Table 8. Highest confidence and overall ranking sequences

Cycle no	Sequences based on U_score:														
1925	['E1'	'E2'	'H1'	'H2'	'H3'	'H4'	'H5'	'A2'	'A1'	'B'	'C1'	'C2'	'D'	'F'	'G']
1911	['E1'	'E2'	'H1'	'H2'	'H3'	'H4'	'H5'	'A2'	'A1'	'B'	'C1'	'C2'	'D'	'F'	'G']
1987	['E1'	'E2'	'H1'	'H2'	'A2'	'H3'	'H4'	'H5'	'A1'	'B'	'C1'	'C2'	'D'	'G'	'F']
	Sequences based on F*U score:														
1696	['E1'	'E2'	'H1'	'H2'	'H3'	'H4'	'H5'	'A2'	'A1'	'B'	'C1'	'C2'	'D'	'F'	'G']
1749	['E1'	'E2'	'H1'	'H2'	'H3'	'H4'	'H5'	'A2'	'A1'	'B'	'C1'	'C2'	'D'	'F'	'G']
1719	['E1'	'E2'	'H1'	'H2'	'A2'	'H3'	'H4'	'H5'	'A1'	'B'	'C1'	'C2'	'D'	'G'	'F']

This table is a section of the given outputs and illustrates that towards the end of the run, the program had settled on a pattern. The DSPs provided in cycles 1925 and 1696 are almost flawless in design.

5.3 The Piston Example 2b

So once again to validate the performance of the Algorithm and its' ability to adapt, let us assume that on average H1 – H5 and E1 & E2 are likely to fail almost 2.5 times as often. Here are the results:

Tables 9(A-C). Example 2b results

A) Training 2b

C	CoF	F
A1	5-10%	160
A2	5-10%	171
B	5-10%	150
C1	5-10%	171
C2	5-10%	130
D	5-10%	125
E1	15-25%	471
E2	15-25%	454
F	5-10%	175
G	5-10%	86
H1	15-25%	411
H2	15-25%	409
H3	15-25%	380
H4	15-25%	305
H5	15-25%	412

B) Compressed mean feasibility 2b

	x+	x-	y+	y-	z+	z-
A1	200	200	200	200	200	-46
A2	200	200	200	200	200	-26
B	290	290	290	290	400	147
C1	84	84	84	84	75	70
C2	84	84	84	84	70	72
D	400	400	290	290	272	400
E1	200	200	3	200	200	200
E2	200	200	200	7	200	200
F	400	400	79	76	400	400
G	800	800	589	595	800	557
H1	100	100	100	100	32	100
H2	100	100	100	100	30	100
H3	100	100	100	100	38	100
H4	100	100	100	100	46	100
H5	100	100	100	100	42	100

C) Comp mean confidence 2b (no remote)

	x+	x-	y+	y-	z+	z-
A1	0	0	0	0	0	0.041
A2	0	0	0	0	0	0.05
B	0.033	0.033	0.033	0.033	0	0.029
C1	0.026	0.026	0.026	0.026	0.015	0.011
C2	0.026	0.026	0.026	0.026	0.011	0.015
D	0	0	0.027	0.027	0.015	0
E1	0	0	0.037	0	0	0
E2	0	0	0	0.042	0	0
F	0	0	0.053	0.071	0	0
G	0	0	0.027	0.045	0	0.177
H1	0	0	0	0	0.034	0
H2	0	0	0	0	0.036	0
H3	0	0	0	0	0.037	0
H4	0	0	0	0	0.039	0
H5	0	0	0	0	0.038	0

A1	A2	B	C1	C2	D	E1	F	E2	H1	H2	H3	H4	G	H5
Z-	Z-	Z-	Z-	Z-	Y-	Y+	Y+	Y-	Z+	Z+	Z+	Z+	Z-	X+

A relative turnaround and almost a valid DSP but for D being removed prematurely. From the mean results this decision comes across as nonsensical, as even after removal of C2 the feasibility score of D should not drop below that of E1 and E2, however, it is difficult to read into the system, which is somewhat of a black box in this respect. D shares impossible contacts in both X and Z axes, and directional results show the part was removed in the Y axis. This highlights that component D, in believing this move was feasible, had little awareness of its remote constraints. A quick solution could entail raising the remote interference feasibility score and lowering the penalty cost. Nevertheless, improved definition of D's remote constraints with E1, E2 and G could have prevented this.

On a more successful note, the overall system adapted well to the changes in input. Initially disassembling the reliable portion of the product first. Feasibility scores of H1-H5 are unsurprisingly within a tight range and this has led to a variety of orders within the overall results. Still, the remote interference conditions and the *method of scoring* has led to sensible disassembly instructions being presented. Finally, G is removed from H5 which is practically unfeasible, but given the initial conditions the program believes it theoretically is. The implementation of priority relationships through geometric consideration and/or a tooling matrix would preclude this event.

5.4 The 22-part Example

In this final example, several tests allowing a wide CoF have been conducted, with promising results. Product C has 22 parts and consequently 2.5x10^{19} possible sequences that the program could output. Thus, the mode sequence method is unviable, especially considering the randomness within our training data. For complex products we are more inclined to evaluate the sequences' overall score:

Table 10. Sequence quality and frequency data for Product C

Example 3

	Entire Passes/Fails: 62999 / 9041 (7:1)																					
Cycle no	**Sequences based on F_score:**																					
10	['N3'	'N2'	'W4'	'N1'	'W3'	'W6'	'B7'	'B4'	'B6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B2'	'W2'	'B1'	'W1'	'C2'	'S1'	'C1'	'S2']
10	['N3'	'N2'	'W4'	'N1'	'W3'	'W6'	'B7'	'B4'	'B6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B2'	'W2'	'B1'	'W1'	'C2'	'S1'	'C1'	'S2']
10	['N3'	'N2'	'W4'	'N1'	'W3'	'W6'	'B7'	'B4'	'B6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B2'	'W2'	'B1'	'W1'	'C2'	'S1'	'C1'	'S2']
	Sequences based on U_score:																					
705	['N3'	'N2'	'W4'	'B7'	'B6'	'B4'	'N1'	'W3'	'W6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B1'	'W1'	'S1'	'B2'	'C2'	'S2'	'W2'	'C1']
735	['N3'	'N2'	'W4'	'B7'	'B6'	'B4'	'N1'	'W3'	'W6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B1'	'W1'	'S1'	'B2'	'C2'	'S2'	'C1'	'W2']
689	['N3'	'N2'	'W4'	'B7'	'B6'	'B4'	'N1'	'W3'	'W6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B1'	'W1'	'B2'	'C2'	'S2'	'S1'	'C1'	'W2']
	Sequences based on F*U score:																					
890	['N3'	'N2'	'W4'	'B7'	'B6'	'N1'	'W3'	'B4'	'B5'	'W6'	'B3'	'W5'	'C4'	'C3'	'B1'	'W1'	'B2'	'C2'	'S2'	'S1'	'C1'	'W2']
638	['N3'	'N2'	'W4'	'B7'	'B4'	'B6'	'N1'	'W3'	'W6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B1'	'W1'	'B2'	'C2'	'S2'	'S1'	'C1'	'W2']
724	['N3'	'N2'	'W4'	'B7'	'B6'	'B4'	'N1'	'W3'	'W6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B1'	'W1'	'B2'	'C2'	'S1'	'S2'	'W2'	'C1']
Frequency	**Dictionary of r_comps sequences:**																					
27	('N3'	'N2'	'W4'	'B7'	'B2'	'W2'	'B4'	'N1'	'W3'	'S2'	'B6'	'B1'	'C2'	'S1'	'W1'	'W6'	'B3'	'W5'	'C4'	'B5'	'C1'	C3')
26	('N3'	'N2'	'W4'	'B7'	'B2'	'W2'	'B4'	'S2'	'N1'	'W3'	'B6'	'B1'	'C2'	'S1'	'W1'	'W6'	'B3'	'W5'	'C4'	'B5'	'C1'	C3')
23	('N3'	'N2'	'W4'	'B7'	'B2'	'W2'	'B4'	'N1'	'W3'	'B6'	'S2'	'B1'	'C2'	'S1'	'W1'	'W6'	'B3'	'W5'	'C4'	'B5'	'C1'	C3')
	Dictionary of r_comps sequences with directions:																					
11	('N3x-'	'N2z-'	'W4z-'	'B1z+'	'W1z+'	'N1z-'	'W3z-'	'B7z-'	'B2z+'	'C2z-'	'S2z-'	'S1z-'	'W2z+'	'B4z-'	'B6z-'	'W6x-'	'B3x+'	'W5x+'	'C4x+'	'B5z-'	C1z-'	C3z+')
10	('N3x-'	'N2z-'	'W4z-'	'B7z-'	'B2z+'	'W2z+'	'B4z-'	'S2y+'	'N1z-'	'W3z-'	'B6z-'	'B1z+'	'C2z-'	'S1z-'	'W1z+'	'W6x-'	'B3x+'	'W5x+'	'C4x+'	'B5z-'	C1z-'	C3x+')
9	('N3x-'	'N2z-'	'W4z-'	'B7z-'	'B2z+'	'W2z+'	'B4z-'	'S2y-'	'N1z-'	'W3z-'	'B6z-'	'B1z+'	'C2z-'	'S1z-'	'W1z+'	'W6x-'	'B3x+'	'W5x+'	'C4x+'	'B5z-'	C1z-'	C3y-')

This is the sequence output from a 3000-cycle run. These strings illustrate how the stochastic data can lead to unfeasible sequences being performed regularly. This is shown by the greatest frequency sequences containing errors, i.e. the removal of B1 and B2 before the top half of the assembly is

unfeasible due to remote constraints. Nevertheless, the implementation of the initial feasibility matrix (which is not dynamic but does undergo data removal during cycles) and the dynamic confidence matrix allows for a combination of what is feasible and reliable. Cycle 890 received the best score of the run:

N3	N2	W4	B7	B6	N1	W3	B4	B5	W6	B3	W5	C4	C3	B1	W1	B2	C2	S2	S1	C1	W2
X-	Z-	Z-	Z-	Z-	Y-	Z-	Z-	Z-	X-	X+	X+	X+	Z+	Z+	Z+	Z+	X-	X+	X-	Z-	Z+

This is a feasible, sensible and realistic DSP method. Notable successes include the disassembly of N3, B3-B7, W5 & W6, and C4 & C3, to allow for the lower half of the product to be disassembled without remote interference. The successful navigation of components throughout the scoring system, in what is a relatively complex product, demonstrates a balance has been struck between feasibility and creativity.

5.5 Limitations

Based on the work described, the following limitations exist:

1. The use of probability ranges as *training data* allows for unfeasible sequences to occur, adding noise to our results.
2. The stochastic training data provides a degree of randomness, thus, contributing to a non-deterministic output.
3. After a significant number of cycles the search mechanism is unlikely to escape the local minima.
4. The Algorithm only considers initial conditions, passes and failures. Thus, some bizarre removal decisions concerning geometry, timing and tooling remain viable.
5. The Algorithm is being evaluated as an analytical tool and as such steady state conditions are realised. For real time use, mean aggregate confidence will not be fed back into the initial conditions, instead confidence will be maintained cycle to cycle, allowing the algorithm to remain agile.
6. Increases in product complexity will quickly render the Algorithm unsuitable for DSP generation due to its' iterative nature. Figure 7 shows the run times for each example.

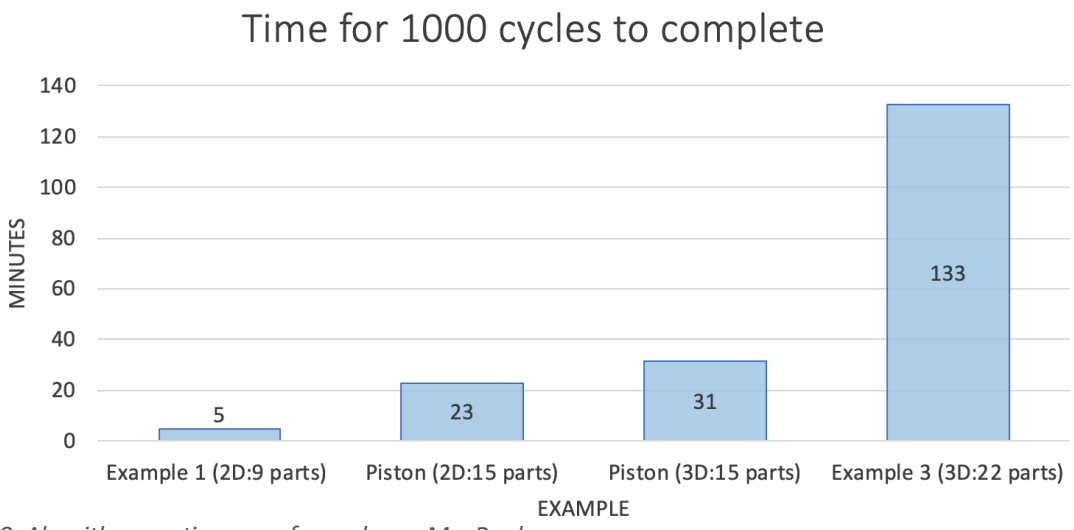

Figure 8. Algorithm run times, performed on a MacBook.

6 Conclusion

This paper proposed a fuzzy dynamic uncertainty matrix in combination a re-planning strategy to define an optimum DSP based upon past events. It has successfully provided valid and creative solutions, including steady state matrices, that depict the product condition with greater accuracy than predicted. The re-planning algorithm is designed for complete disassembly but can be easily modified for different products and 3 disassembly modes.

Utilised as an analytical tool in this study, the Algorithm demonstrated the generation of effective sequences through multiple outputs. However, due to its iterative nature and un-optimised search methods, when product complexity increases the algorithm plummets in efficiency. Future work could integrate evolutionary algorithms to advance sequences of greatest overall score, speeding up the exhaustive process.

Future implementation of size, weight and tooling constraints into the fuzzy matrices could allow for priority relationships between components. This may eradicate some of the unusual decisions we have seen. Increased efficiency in terms of timings and costs would allow for greater disassembly completion, and consequently greater environmental benefit. Furthermore, product life cycle aspects such as re-use, recycling and scrapping can be included to prioritise energy savings.

The unique selling point of this study is the real-time adaptation to predicted uncertainty, thus, in future work the non-deterministic program will consider size, weight, and tooling constraints, before being trialled with real training data. Successful execution could see robots make intelligent decisions on predicted product condition. Contributing to the circular economy and providing extraordinary flexibility for industry 4.0.

7 Acknowledgment

I would like to thank my supervisor Dr. Y.J. Wang for his continued support and guidance throughout this project.

8 References

[1] Y. Wang *et al.*, "Interlocking problems in disassembly sequence planning," *International Journal of Production Research,* pp. 1-13, 2020, doi: 10.1080/00207543.2020.1770892.

[2] "<SYR_AR5_FINAL_full_wcover.pdf> Climate change 2014 synthesis report 2014 ipcc."

[3] "<WPP2019_Highlights.pdf> World Population Prospects - Population Division - United Nations." [Online]. Available: Available at: https://population.un.org/wpp/Publications/ (Accessed: 3 December 2020).

[4] Y. de Grandbois, "Chapter 4 - Service Science for a Smarter Planet," in *Service Science and the Information Professional*, Y. de Grandbois Ed.: Chandos Publishing, 2016, pp. 75-97.

[5] J. Gowdy, "Our hunter-gatherer future: Climate change, agriculture and uncivilization," *Futures,* vol. 115, p. 102488, 2020/01/01/ 2020, doi: https://doi.org/10.1016/j.futures.2019.102488.

[6] L. Kemp, "Are we on the road to civilisation collapse?. Retrieved 27 November 2020," 2020. [Online]. Available: https://www.bbc.com/future/article/20190218-are-we-on-the-road-to-civilisation-collapse.

[7] G. Raz, A. Ovchinnikov, and V. Blass, "Economic, Environmental, and Social Impact of Remanufacturing in a Competitive Setting," *IEEE Transactions on Engineering Management,* vol. 64, no. 4, pp. 476-490, 2017, doi: 10.1109/TEM.2017.2714698.

[8] D. Singhal, S. Tripathy, and S. K. Jena, "Remanufacturing for the circular economy: Study and evaluation of critical factors," *Resources, Conservation and Recycling,* vol. 156, p. 104681, 2020/05/01/ 2020, doi: https://doi.org/10.1016/j.resconrec.2020.104681.

[9] R. Smith, "How Much Cheaper Are SpaceX Reusable Rockets? Now We Know | The Motley Fool," 2020. [Online]. Available: https://www.fool.com/investing/2020/10/05/how-much-cheaper-are-spacex-reusable-rockets-now-w/.

[10] X. Zhang, M. Zhang, H. Zhang, Z. Jiang, C. Liu, and W. Cai, "A review on energy, environment and economic assessment in remanufacturing based on life cycle assessment method," *Journal of Cleaner Production,* vol. 255, p. 120160, 2020/05/10/ 2020, doi: https://doi.org/10.1016/j.jclepro.2020.120160.

[11] C.-M. Lee, W.-S. Woo, and Y.-H. Roh, "Remanufacturing: Trends and issues," *International Journal of Precision Engineering and Manufacturing-Green Technology,* vol. 4, pp. 113-125, 01/01 2017, doi: 10.1007/s40684-017-0015-0.

[12] R. K. R. S. S. H. T. J. J. K. Parker D and et al., "Remanufacturing Market Study. European Enterprise Network," 2015.

[13] N. Abdullah, F. A. Jafar, and M. N. Maslan, "Analysis on Factors Impeding the Disassembly Process with Consideration on Automated Disassembly Planning," *Procedia Manufacturing,* vol. 2, pp. 191-195, 2015/01/01/ 2015, doi: https://doi.org/10.1016/j.promfg.2015.07.033.

[14] T. Kanehara, T. Suzuki, A. Inaba, and S. Okuma, "On algebraic and graph structural properties of assembly Petri net-searching by linear programming," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, 1993, vol. 3: IEEE, pp. 2286-2293.

[15] R. J. Linn, "Computer-aided assembly planning," in *Integrated Product, Process and Enterprise Design*, B. Wang Ed. Boston, MA: Springer US, 1997, pp. 266-303.

[16] A. J. D. Lambert, "Disassembly sequencing: A survey," *International Journal of Production Research - INT J PROD RES,* vol. 41, pp. 3721-3759, 11/10 2003, doi: 10.1080/0020754031000120078.

[17] A. J. D. Lambert and S. M. Gupta, W. Algemene, Ed. *Disassembly modeling for assembly, maintenance, reuse, and recycling* (The St. Lucie Press/APICS series on resource management). CRC Press (in en), 2005.

[18] L. S. Homen, "A Correct and Complete Algorithm for Generation of Mechanical Assembly Sequences," *IEEE Trans, RA-7-2,* 1991 1991. [Online]. Available: https://ci.nii.ac.jp/naid/10004215218/en/.

[19] A. Bourjault, "Contribution à une approche méthodologique de l'assemblage automatisé: élaboration automatique des séquences opératoires," *Thése d'Etat, Université de Franche-Comté,* 1984.

[20] T. De Fazio and D. Whitney, "Simplified generation of all mechanical assembly sequences," *IEEE Journal on Robotics and Automation,* vol. 3, no. 6, pp. 640-658, 1987.

[21] Y. Ren, C. Zhang, F. Zhao, H. Xiao, and G. Tian, "An asynchronous parallel disassembly planning based on genetic algorithm," *European Journal of Operational Research,* vol. 269, no. 2, pp. 647-660, 2018/09/01/ 2018, doi: https://doi.org/10.1016/j.ejor.2018.01.055.

[22] H.-E. Tseng, C.-C. Chang, S.-C. Lee, and Y.-M. Huang, "Hybrid bidirectional ant colony optimization (hybrid BACO): An algorithm for disassembly sequence planning," *Engineering Applications of Artificial Intelligence,* vol. 83, pp. 45-56, 2019/08/01/ 2019, doi: https://doi.org/10.1016/j.engappai.2019.04.015.

[23] D. Agrawal, S. Kumara, and D. Finke, "Automated assembly sequence planning and subassembly detection," *IIE Annual Conference and Expo 2014,* pp. 781-788, 01/01 2014.

[24] H.-E. Tseng, C.-C. Chang, S.-C. Lee, and Y.-M. Huang, "A Block-based genetic algorithm for disassembly sequence planning," *Expert Systems with Applications,* vol. 96, pp. 492-505, 2018/04/15/ 2018, doi: https://doi.org/10.1016/j.eswa.2017.11.004.

[25] M. K. M. T. N. A. 1, "Optimization of disassembly sequence planning for preventive maintenance," 2016. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s00170-016-9434-2.pdf.

[26] J. H. L. J.F. WANG, 2 S.Q. LI,1 and Y.F. ZHONG1, "Intelligent selective disassembly using the ant colony algorithm," 2003.

[27] X. Liu, G. Peng, X. Liu, and Y. Hou, "Disassembly sequence planning approach for product virtual maintenance based on improved max–min ant system," *The International Journal of Advanced Manufacturing Technology,* vol. 59, no. 5, pp. 829-839, 2012/03/01 2012, doi: 10.1007/s00170-011-3531-z.

[28] D. Agrawal, P. T. Nallamothu, S. R. Mandala, S. Kumara, and D. Finke, "Automated disassembly sequence planning and optimization," *IIE Annual Conference and Expo 2013,* pp. 122-131, 01/01 2013.

[29] H.-E. Tseng, Y.-M. Huang, C.-C. Chang, and S.-C. Lee, "Disassembly sequence planning using a Flatworm algorithm," *Journal of Manufacturing Systems,* vol. 57, pp. 416-428, 2020/10/01/ 2020, doi: https://doi.org/10.1016/j.jmsy.2020.10.014.

[30] Z. Zhou *et al.*, "Disassembly sequence planning: Recent developments and future trends," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* vol. 233, pp. 1450 - 1471, 2019.

[31] Y. Laili, F. Tao, D. T. Pham, Y. Wang, and L. Zhang, "Robotic disassembly re-planning using a two-pointer detection strategy and a super-fast bees algorithm," *Robotics and computer-integrated manufacturing,* vol. 59, pp. 130-142, doi: 10.1016/j.rcim.2019.04.003.

[32] J. Wang and V. Allada, "Hierarchial fuzzy neural network based serviceability evaluation," *International Journal of Agile Management Systems,* vol. 2, pp. 130-141, 08/01 2000, doi: 10.1108/14654650010337140.

[33] T. Ying, Z. MengChu, and G. Meimei, "Fuzzy-Petri-net-based disassembly planning considering human factors," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans,* vol. 36, no. 4, pp. 718-726, 2006, doi: 10.1109/TSMCA.2005.853508.

[34] S. Behdad, L. Berg, J. Vance, and D. Thurston, "Immersive Computing Technology to Investigate Tradeoffs Under Uncertainty in Disassembly Sequence Planning," *Journal of Mechanical Design,* vol. 136, no. 7, 2014, doi: 10.1115/1.4025021.

[35] G. Tian, M. Zhou, and P. Li, "Disassembly Sequence Planning Considering Fuzzy Component Quality and Varying Operational Cost," *IEEE Transactions on Automation Science and Engineering,* vol. 15, no. 2, pp. 748-760, 2018, doi: 10.1109/TASE.2017.2690802.

[36] X. F. Zhang, G. Yu, Z. Y. Hu, C. H. Pei, and G. Q. Ma, "Parallel disassembly sequence planning for complex products based on fuzzy-rough sets," *The International Journal of Advanced Manufacturing Technology,* vol. 72, no. 1-4, pp. 231-239, 2014, doi: 10.1007/s00170-014-5655-4.

[37] I. f. S. policy and Development, "<Made-in-China-Backgrounder.pdf>," 2018. [Online]. Available: https://isdp.eu/.

[38] M. Kerin and D. T. Pham, "A review of emerging industry 4.0 technologies in remanufacturing," *Journal of Cleaner Production,* vol. 237, p. 117805, 2019/11/10/ 2019, doi: https://doi.org/10.1016/j.jclepro.2019.117805.

[39] J. Cao, X. Chen, X. Zhang, Y. Gao, X. Zhang, and S. Kumar, "Overview of remanufacturing industry in China: Government policies, enterprise, and public awareness," *Journal of Cleaner Production,* vol. 242, p. 118450, 2020/01/01/ 2020, doi: https://doi.org/10.1016/j.jclepro.2019.118450.

[40] G. Jin, W. Li, S. Wang, and S. Gao, "A systematic selective disassembly approach for Waste Electrical and Electronic Equipment with case study on liquid crystal display televisions," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* vol. 231, no. 13, pp. 2261-2278, 2017, doi: 10.1177/0954405415575476.

Appendix 1: Nomenclature

Table 11: List of Abbreviations

Abbreviation	Full name
DSP	Disassembly Sequence Planning
CoF	Chance of Fail
EoL	End-of-Life
F matrix	Feasibility matrix
U matrix	Confidence matrix
C matrix	Contact matrix
R matrix	Relation matrix

Appendix 2: Equations

Table 12: List of Equations & descriptions

$$F_{C1X+} = \sum 0 + F_{C1x+}^{C2} + F_{C1x+}^{C3} + F_{C1x+}^{C4} + F_{C1x+}^{C5} + F_{C1x+}^{f1} + F_{C1x+}^{f2} + F_{C1x+}^{f3} + F_{C1x+}^{f4} \quad \textbf{(1)}$$

F_{C1X+}	Feasibility of C1 in the X+ direction
F_{C1x+}^{C2}	Feasibility interference between C1 and C2, when moving C1 in the X+ direction

$$S = \begin{bmatrix} F_{C1X+} & F_{C1X-} & F_{C1Y+} & F_{C1Y-} \\ F_{C2X+} & F_{C2X-} & F_{C2Y+} & F_{C2Y-} \\ \vdots & \vdots & \vdots & \vdots \\ F_{f4X+} & F_{f4X-} & F_{f4Y+} & F_{f4Y-} \end{bmatrix} + \begin{bmatrix} \mu_{C1X+} & \mu_{C1X-} & \mu_{C1Y+} & \mu_{C1Y-} \\ \mu_{C2X+} & \mu_{C2X-} & \mu_{C2Y+} & \mu_{C2Y-} \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{f4X+} & \mu_{f4X-} & \mu_{f4Y+} & \mu_{f4Y+} \end{bmatrix} + 2 \begin{bmatrix} \sigma_{C1X+} & \sigma_{C1X-} & \sigma_{C1Y+} & \sigma_{C1Y-} \\ \sigma_{C2X+} & \sigma_{C2X-} & \sigma_{C2Y+} & \sigma_{C2Y-} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{f4X+} & \sigma_{f4X-} & \sigma_{f4Y+} & \sigma_{f4Y-} \end{bmatrix} \quad \textbf{(2)}$$

S	Summation matrix

$$U_{+1} = U \times e^{\left(\frac{0.01}{U}\right)} \quad \textbf{(3)}$$

U_{+1}	Next iterations confidence value

$$U_{new} = \frac{total\_aggregate\_U}{Run\_size} \quad \textbf{(4)}$$

U_{new}	New Confidence matrix for current cycle use
$Total\_aggregate\_U$	Total matrix of all component confidence values upon removal
Run_size	The number of cycles completed

$$F_{new} = (F_{store} + P_{aggregate}) - \left(\frac{\sum P_{aggregate}}{N_{U-contacts}} \begin{bmatrix} \frac{U_{c1x+}^{c1}}{U_{c1x+}^{c1}} & \frac{U_{c1x-}^{c1}}{U_{c1x-}^{c1}} & \cdots & \frac{U_{c2x-}^{f4}}{U_{c1x-}^{f4}} \\ \frac{U_{c2x+}^{c1}}{U_{c2x+}^{c1}} & \frac{U_{c2x-}^{c1}}{U_{c2x-}^{c1}} & \cdots & \frac{U_{c2y-}^{f4}}{U_{c2y-}^{f4}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{U_{f4x+}^{c1}}{U_{f4x+}^{c1}} & \frac{U_{f4x+}^{c1}}{U_{f4x+}^{c1}} & \cdots & \frac{U_{f4x+}^{f4}}{U_{f4x+}^{f4}} \end{bmatrix} \right) \quad \textbf{(5)}$$

F_{new}	New Feasibility matrix for current cycle use
F_{store}	Stored Feasibility matrix for next cycle use
$P_{aggregate}$	Total penalties for last iteration
$N_{U-contacts}$	Number of contacts with associated confidence values

Appendix 3: Example 1 Contact, Feasibility & Confidence source matrices

Contact Matrix

	C1-X	C1-X	C1Y	C1-Y	C2X	C2-X	C2Y	C2-Y	C3X	C3-X	C3Y	C3-Y	C4X	C4-X	C4Y	C4-Y	C5X	C5-X	C5Y	C5-Y	f1X	f1-X	f1Y	f1-Y	f2X	f2-X	f2Y	f2-Y	f3X	f3-X	f3Y	f3-Y	f4X	f4-X	f4Y	f4-Y
C1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
C2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
C3	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
C4	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
C5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
f1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f2	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
f3	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
f4	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Feasibility Matrix

	C1-X	C1-X	C1Y	C1-Y	C2X	C2-X	C2Y	C2-Y	C3X	C3-X	C3Y	C3-Y	C4X	C4-X	C4Y	C4-Y	C5X	C5-X	C5Y	C5-Y	f1X	f1-X	f1Y	f1-Y	f2X	f2-X	f2Y	f2-Y	f3X	f3-X	f3Y	f3-Y	f4X	f4-X	f4Y	f4-Y
C1	0	0	0	0	0.5	0.5	0.01	100	0.5	0.5	0	0.5	0.5	0.5	0	0.5	0.2	0.2	0.01	100	100	100	100	100	0.5	0.5	0	0.5	0.5	0.5	0	0.5	100	100	100	100
C2	0.5	0.5	100	0.01	0	0	0	0	0	0	0	0.5	100	100	0.2	100	0	0	0	0.5	0	0	0.5	0	100	100	100	100	0	0	0.5	0.5	0.5	0.5	0	0
C3	0.5	0.5	100	0	0	0	0.5	0	0	0	0	0	0.2	0.2	100	0.01	0.2	0.2	0.01	100	0	0	0.5	0	0	0.5	0	0	100	100	100	100	100	100	100	100
C4	0.5	0.5	0.5	0	0	0	0.5	0	100	100	100	0.2	0	0	0	0	0	0	0	0	0	0	0	0	100	100	100	100	100	100	100	100	100	100	100	100
C5	0.2	0.2	100	0.01	0	0	0.5	0	0	0	0.5	0	0.2	0.2	100	0.01	0	0	0	0.5	0	0	0.5	0	0	0	0	0	0	0	0	0.5	0	0	0	0
f1	100	100	100	0.3	100	100	100	0.3	0	0	0	0.5	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f2	0.5	0.5	0.5	0	0	0	0.5	0	100	100	0.3	100	0.5	0	0	0	100	100	0.3	100	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0
f3	0.5	0.5	0.5	0	0	0	0.5	0	0	0	0.5	0	100	100	100	0.3	100	100	100	0.3	0	0	0.5	0	0	0.5	0	0	0	0	0	0	0.5	0	0	0
f4	100	100	100	0.3	0	0	0	0	0	0	0	0	0	0	0	0	100	100	0.3	0	0	0	0	0	0	0.5	0	0	0	0.5	0	0	0	0	0	0

Confidence Matrix

	C1-X	C1-X	C1Y	C1-Y	C2X	C2-X	C2Y	C2-Y	C3X	C3-X	C3Y	C3-Y	C4X	C4-X	C4Y	C4-Y	C5X	C5-X	C5Y	C5-Y	f1X	f1-X	f1Y	f1-Y	f2X	f2-X	f2Y	f2-Y	f3X	f3-X	f3Y	f3-Y	f4X	f4-X	f4Y	f4-Y
C1	0.00	0.00	0.00	0.00	0.06	0.06	0.01	0.00	0.06	0.06	0.00	0.06	0.06	0.06	0.00	0.06	0.04	0.04	0.01	0.00	0.00	0.00	0.00	0.00	0.06	0.06	0.00	0.06	0.06	0.06	0.00	0.06	0.00	0.00	0.00	0.00
C2	0.06	0.06	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00
C3	0.06	0.06	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.04	0.04	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00
C4	0.06	0.06	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.04	0.04	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.06	0.00	0.00	0.00
C5	0.04	0.04	0.00	0.01	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f1	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f2	0.06	0.06	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.04	0.00	0.06	0.06	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.06	0.00	0.00	0.06	0.00	0.00	0.00
f3	0.06	0.06	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.04	0.00	0.06	0.06	0.00	0.04	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.06	0.00	0.00	0.06	0.00	0.00	0.00
f4	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Appendix 3a: Example 1a

R = 1001 (Run_size), Pc = 15 (Penalty cost), sigk = 0.33

Example 1a Entire Passes/Fails: 8999 / 1029

Cycle no	Sequences based on F_score:								
1	['f4'	'f3'	'f1'	'C1'	'C2'	'f2'	'C5'	'C4'	'C3']
0	['f3'	'f4'	'f2'	'C5'	'C4'	'C3'	'f1'	'C2'	'C1']
304	['f4'	'f2'	'f3'	'C5'	'f1'	'C1'	'C2'	'C4'	'C3']
	Sequences based on U_score:								
997	['f4'	'f2'	'f3'	'C5'	'C4'	'C3'	'C1'	'C2'	'f1']
580	['f4'	'f3'	'f2'	'C5'	'C4'	'C3'	'C1'	'C2'	'f1']
917	['f4'	'f2'	'f3'	'C5'	'C3'	'C4'	'f1'	'C2'	'C1']
	Sequences based on F*U score:								
917	[f4'	'f2'	'f3'	'C5'	'C3'	'C4'	'f1'	'C2'	C1']
857	['f2'	'f4'	'f3'	'C5'	'C3'	'C4'	'f1'	'C1'	'C2']
915	['f4'	'f3'	'f2'	'C5'	'C3'	'C4'	'f1'	'C1'	'C2']
Frequency	Dictionary of r_comps sequences:								
199	f4'	'f2'	'f3'	'C5'	'C4'	'C3'	'f1'	'C1'	C2'
162	f4'	'f2'	'f3'	'C5'	'C4'	'C3'	'f1'	'C2'	C1'
52	f4'	'f3'	'f2'	'C5'	'C4'	'C3'	'f1'	'C2'	C1'
	Dictionary of r_comps sequences with directions:								
58	('f4y-'	'f2y+'	'f3y-'	'C5y-'	'C4y-'	'C3y-'	'f1y-'	'C1y+'	C2y-)
50	('f4y-'	'f2y+'	'f3y-'	'C5y-'	'C4y-'	'C3y-'	'f1y-'	'C1y+'	C2x+')
49	('f4y-'	'f2y+'	'f3y-'	'C5y-'	'C4y-'	'C3y-'	'f1y-'	'C1y+'	C2x-')

compressed_mean_f1a

	x+	x-	y+	y-
C1	239	239	206	381
C2	111	111	179	94
C3	180	180	229	180
C4	207	207	219	203
C5	332	332	480	310
f1	179	179	179	58
f2	181	180	30	179
f3	180	180	181	27
f4	179	181	179	36

compressed_mean_u1a

	x+	x-	y+	y-
C1	0.30	0.30	0.03	0.24
C2	0.03	0.03	0.00	0.31
C3	0.12	0.06	0.17	0.12
C4	0.15	0.15	0.22	0.02
C5	0.05	0.05	0.18	0.03
f1	0.00	0.00	0.00	0.30
f2	0.24	0.06	0.17	0.00
f3	0.12	0.12	0.24	0.05
f4	0.00	0.24	0.00	0.05

Training data 1a

	CoF	Actual fails
C1	0.08-0.12	69
C2	0.08-0.12	72
C3	0.08-0.12	100
C4	0.08-0.12	120
C5	0.08-0.12	106
f1	0.09-0.15	164
f2	0.08-0.12	125
f3	0.08-0.12	129
f4	0.08-0.12	130

Sigmoid of Feasibility 1a

	x+	x-	y+	y-
C1	0.45	0.45	0.35	0.85
C2	0.13	0.13	0.27	0.11
C3	0.27	0.27	0.42	0.27
C4	0.35	0.35	0.39	0.34
C5	0.74	0.74	0.95	0.68
f1	0.27	0.27	0.27	0.07
f2	0.28	0.27	0.05	0.27
f3	0.27	0.27	0.28	0.05
f4	0.27	0.28	0.27	0.05

Appendix 3b: Example 1b

R = 1001 (Run_size), Pc = 15 (Penalty cost), sigk = 0.33

Example 1b Entire Passes/Fails: 8999 / 1048

Cycle no	Sequences based on F_score:								
1	f3'	'f2'	'f4'	'C5'	'C4'	'C3'	'f1'	'C1'	C2'
3	f3'	'f4'	'f2'	'C5'	'C4'	'C3'	'f1'	'C2'	C1'
9	f3'	'f4'	'f2'	'C5'	'C4'	'C3'	'f1'	'C2'	C1'
	Sequences based on U_score:								
2	f3'	'f4'	'f2'	'C5'	'C4'	'C3'	'C2'	'f1'	C1'
33	f3'	'f2'	'f4'	'C5'	'C4'	'C3'	'C1'	'f1'	C2'
16	f3'	'f4'	'f2'	'C5'	'C4'	'C3'	'C1'	'f1'	C2'
	Sequences based on F*U score:								
1	f3'	'f2'	'f4'	'C5'	'C4'	'C3'	'f1'	'C1'	C2'
3	f3'	'f4'	'f2'	'C5'	'C4'	'C3'	'f1'	'C2'	C1'
11	f3'	'f4'	'f2'	'C5'	'C4'	'C3'	'f1'	'C2'	C1'
Frequency	Dictionary of r_comps sequences:								
144	f4'	'f1'	'C2'	'C1'	'f3'	'f2'	'C3'	'C5'	C4'
90	f4'	'f1'	'C2'	'C1'	'f3'	'f2'	'C3'	'C4'	C5'
86	f4'	'f1'	'C2'	'C1'	'f3'	'C4'	'C5'	'f2'	C3'
	Dictionary of r_comps sequences with directions:								
35	f4y-'	'f1y-'	'C2y-'	'C1y+'	'f3y-'	'f2y+'	'C3y+'	'C5y-'	C4x+'
33	f4y-'	'f1y-'	'C2y-'	'C1y+'	'f3y-'	'f2y+'	'C3y+'	'C4y+'	C5x-'
30	f4y-'	'f1y-'	'C2y-'	'C1y+'	'f3y-'	'f2y+'	'C3y+'	'C5y-'	C4y-'

comp_mean_u1b

	x+	x-	y+	y-
C1	0.30	0.30	0.03	0.24
C2	0.03	0.03	0.00	0.31
C3	0.12	0.06	0.17	0.12
C4	0.15	0.15	0.22	0.02
C5	0.06	0.06	0.18	0.04
f1	0.00	0.00	0.00	0.30
f2	0.24	0.06	0.17	0.00
f3	0.12	0.12	0.24	0.05
f4	0.00	0.24	0.00	0.05

comp_mean_f1b

	x+	x-	y+	y-
C1	242	242	205	377
C2	113	113	175	92
C3	176	176	239	176
C4	213	213	238	197
C5	339	339	476	313
f1	175	175	175	26
f2	177	176	60	175
f3	176	176	177	55
f4	175	177	175	23

Training data 1b

	CoF	Actual fails
C1	0.08-0.12	91
C2	0.08-0.12	124
C3	0.08-0.12	82
C4	0.08-0.12	61
C5	0.08-0.12	110
f1	0.08-0.12	95
f2	0.12-0.20	197
f3	0.12-0.20	189
f4	0.08-0.12	99

Sigmoid of Feasibilty 1b

	x+	x-	y+	y-
C1	0.47	0.47	0.36	0.85
C2	0.14	0.14	0.28	0.11
C3	0.28	0.28	0.50	0.28
C4	0.39	0.39	0.49	0.34
C5	0.78	0.78	0.95	0.71
f1	0.28	0.28	0.28	0.05
f2	0.28	0.28	0.09	0.28
f3	0.28	0.28	0.28	0.09
f4	0.28	0.28	0.28	0.05

Appendix 4a: Example 2a

R = 2001 (Run_size), Pc = 10 (Penalty cost)

Example Piston 2a (Balanced set up)

	Entire Passes/Fails: 27335 / 4208 (6.5:1)														
Cycle no	**Sequences based on F initial score:**														
18	['E1'	'E2'	'A2'	'H1'	'H2'	'H3'	'F'	'H4'	'G'	'A1'	'D'	'C2'	'C1'	'B'	'H5']
18	['E1'	'E2'	'A2'	'H1'	'H2'	'H3'	'F'	'H4'	'G'	'A1'	'D'	'C2'	'C1'	'B'	'H5']
18	['E1'	'E2'	'A2'	'H1'	'H2'	'H3'	'F'	'H4'	'G'	'A1'	'D'	'C2'	'C1'	'B'	'H5']
	Sequences based on U_score:														
1925	['E1'	'E2'	'H1'	'H2'	'H3'	'H4'	'H5'	'A2'	'A1'	'B'	'C1'	'C2'	'D'	'F'	'G']
1911	['E1'	'E2'	'H1'	'H2'	'H3'	'H4'	'H5'	'A2'	'A1'	'B'	'C1'	'C2'	'D'	'F'	'G']
1987	['E1'	'E2'	'H1'	'H2'	'A2'	'H3'	'H4'	'H5'	'A1'	'B'	'C1'	'C2'	'D'	'G'	'F']
	Sequences based on F*U score:														
1696	['E1'	'E2'	'H1'	'H2'	'H3'	'H4'	'H5'	'A2'	'A1'	'B'	'C1'	'C2'	'D'	'F'	'G']
1749	['E1'	'E2'	'H1'	'H2'	'H3'	'H4'	'H5'	'A2'	'A1'	'B'	'C1'	'C2'	'D'	'F'	'G']
1719	['E1'	'E2'	'H1'	'H2'	'A2'	'H3'	'H4'	'H5'	'A1'	'B'	'C1'	'C2'	'D'	'G'	'F']
Frequency	**Dictionary of r_comps sequences:**														
86	('E1'	'E2'	'H3'	'H2'	'H4'	'H5'	'H1'	'A2'	'A1'	'B'	'C1'	'C2'	'D'	'F'	G')
75	('E1'	'E2'	'H3'	'H2'	'H4'	'H5'	'H1'	'A2'	'A1'	'B'	'C1'	'C2'	'D'	'G'	F')
46	('E1'	'E2'	'H3'	'H2'	'H4'	'H5'	'H1'	'A1'	'A2'	'B'	'C1'	'C2'	'D'	'F'	G')
	Dictionary of r_comps sequences with directions:														
27	('E1y+'	'E2y-'	'H3z+'	'H2z+'	'H4z+'	'H5z+'	'H1z+'	'A2z-'	'A1z-'	'Bz-'	'C1z-'	'C2z-'	'Dy+'	'Gy-'	'Fx+')
26	('E1y+'	'E2y-'	'H3z+'	'H2z+'	'H4z+'	'H5z+'	'H1z+'	'A2z-'	'A1z-'	'Bz-'	'C1z-'	'C2z-'	'Dy-'	'Fy-'	'Gx+')
21	('E1y+'	'E2y-'	'H3z+'	'H2z+'	'H4z+'	'H5z+'	'H1z+'	'A2z-'	'A1z-'	'Bz-'	'C1z-'	'C2z-'	'Dy+'	'Fy+'	'Gy-')

Initial Feasibility Example 2

	x+	x-	y+	y-	z+	z-
A1	200	200	200	200	200	0.6
A2	200	200	200	200	200	0.6
B	300.2	300.2	300.2	300.2	400	200.02
C1	100.2	100.2	100.2	100.2	100	100.01
C2	100.2	100.2	100.2	100.2	100.01	100.01
D	400	400	300.2	300.2	300.01	400
E1	200	200	0.02	200	200	200
E2	200	200	200	0.02	200	200
F	400	400	100.4	100.41	400	400
G	800	800	600.2	600.21	800	301
H1	100	100	100	100	0.2	100
H2	100	100	100	100	0.2	100
H3	100	100	100	100	0.2	100
H4	100	100	100	100	0.2	100
H5	100	100	100	100	0.2	100

Initial Con Ex.2 (no remote)

	x+	x-	y+	y-	z+	z-
A1	0	0	0	0	0	0.08
A2	0	0	0	0	0	0.08
B	0.05	0.05	0.05	0.05	0	0.02
C1	0.05	0.05	0.05	0.05	0.01	0.01
C2	0.05	0.05	0.05	0.05	0.01	0.01
D	0	0	0.05	0.05	0.01	0
E1	0	0	0.02	0	0	0
E2	0	0	0	0.02	0	0
F	0	0	0.1	0.11	0	0
G	0	0	0.05	0.06	0	0.25
H1	0	0	0	0	0.05	0
H2	0	0	0	0	0.05	0
H3	0	0	0	0	0.05	0
H4	0	0	0	0	0.05	0
H5	0	0	0	0	0.05	0

Training 2a

C	CoF	F
A1	8-20%	351
A2	8-20%	291
B	8-20%	331
C1	8-20%	211
C2	8-20%	172
D	8-20%	272
E1	8-20%	305
E2	8-20%	315
F	8-20%	308
G	8-20%	203
H1	8-20%	296
H2	8-20%	288
H3	8-20%	297
H4	8-20%	307
H5	8-20%	231

Compressed mean feasibility 2a

	x+	x-	y+	y-	z+	z-
A1	200	200	200	200	200	12
A2	200	200	200	200	200	7
B	305	305	305	305	400	160
C1	114	114	114	114	84	88
C2	114	114	114	114	88	78
D	400	400	306	306	278	400
E1	200	200	-41	200	200	200
E2	200	200	200	-42	200	200
F	400	400	111	90	400	400
G	800	800	605	585	800	329
H1	100	100	100	100	2	100
H2	100	100	100	100	1	100
H3	100	100	100	100	5	100
H4	100	100	100	100	11	100
H5	100	100	100	100	10	100

Comp mean con 2a (no remote)

	x+	x-	y+	y-	z+	z-
A1	0	0	0	0	0	0.054
A2	0	0	0	0	0	0.051
B	0.034	0.034	0.034	0.034	0	0.032
C1	0.034	0.034	0.034	0.034	0.016	0.02
C2	0.034	0.034	0.034	0.034	0.02	0.014
D	0	0	0.031	0.031	0.014	0
E1	0	0	0.034	0	0	0
E2	0	0	0	0.028	0	0
F	0	0	0.06	0.077	0	0
G	0	0	0.029	0.046	0	0.149
H1	0	0	0	0	0.029	0
H2	0	0	0	0	0.029	0
H3	0	0	0	0	0.029	0
H4	0	0	0	0	0.029	0
H5	0	0	0	0	0.031	0

Appendix 4b: Example 2b

R = 2001 (Run_size), Pc = 10 (Penalty cost)

Example Piston 2b (~2.5x Mixed set up)

	Entire Passes/Fails: 27335 / 4341 (6.3:1)														
Cycle no	**Sequences based on F initial score:**														
75	['A1'	'E1'	'F'	'E2'	'H1'	'H2'	'A2'	'D'	'C2'	'C1'	'H3'	'H4'	'H5'	'B'	'G']
40	['E1'	'A1'	'F'	'E2'	'A2'	'D'	'C2'	'C1'	'H1'	'H2'	'H3'	'H4'	'G'	'B'	'H5']
40	['E1'	'A1'	'F'	'E2'	'A2'	'D'	'C2'	'C1'	'H1'	'H2'	'H3'	'H4'	'G'	'B'	'H5']
	Sequences based on U_score:														
1880	['A1'	'B'	'C1'	'C2'	'A2'	'D'	'E1'	'F'	'E2'	'H1'	'H2'	'H3'	'H4'	'G'	'H5']
1909	['A1'	'B'	'C1'	'C2'	'A2'	'D'	'E1'	'F'	'E2'	'H1'	'H2'	'H3'	'H4'	'G'	'H5']
1941	['A1'	'A2'	'B'	'C1'	'C2'	'D'	'E1'	'F'	'E2'	'H1'	'H2'	'H3'	'H4'	'G'	'H5']
	Sequences based on F*U score:														
1944	['A1'	'A2'	'B'	'C1'	'C2'	'D'	'E1'	'F'	'E2'	'H1'	'H2'	'H3'	'H4'	'H5'	'G']
1941	['A1'	'A2'	'B'	'C1'	'C2'	'D'	'E1'	'F'	'E2'	'H1'	'H2'	'H3'	'H4'	'G'	'H5']
1994	['A1'	'A2'	'B'	'C1'	'C2'	'D'	'E1'	'F'	'E2'	'H1'	'H2'	'H3'	'H4'	'G'	'H5']
Frequency	**Dictionary of r_comps sequences:**														
48	('A1'	'A2'	'B'	'C1'	'C2'	'D'	'E1'	'F'	'E2'	'H2'	'H1'	'H3'	'H4'	'G'	'H5')
43	('A1'	'A2'	'B'	'C1'	'C2'	'D'	'E1'	'F'	'E2'	'H2'	'H1'	'H3'	'H5'	'G'	'H4')
37	('A1'	'A2'	'B'	'C1'	'C2'	'D'	'E1'	'F'	'E2'	'H1'	'H2'	'H3'	'H5'	'H4'	'G')
	Dictionary of r_comps sequences with directions:														
25	('A1z-'	'A2z-'	'Bz-'	'C1z-'	'C2z-'	'Dy-'	'E1y+'	'Fy+'	'E2y-'	'H2z+'	'H1z+'	'H3z+'	'H5z+'	'Gz-'	H4z+')
25	('A1z-'	'A2z-'	'Bz-'	'C1z-'	'C2z-'	'Dy-'	'E1y+'	'Fy+'	'E2y-'	'H2z+'	'H1z+'	'H3z+'	'H4z+'	'Gz-'	H5z+')
25	('A1z-'	'A2z-'	'Bz-'	'C1z-'	'C2z-'	'Dy-'	'E1y+'	'Fy+'	'E2y-'	'H1z+'	'H2z+'	'H3z+'	'H5z+'	'Gz-'	H4z+')

Initial Feasibility Example 2.

	x+	x-	y+	y-	z+	z-
A1	200	200	200	200	200	0.6
A2	200	200	200	200	200	0.6
B	300.2	300.2	300.2	300.2	400	200.02
C1	100.2	100.2	100.2	100.2	100	100.01
C2	100.2	100.2	100.2	100.2	100.01	100.01
D	400	400	300.2	300.2	300.01	400
E1	200	200	0.02	200	200	200
E2	200	200	200	0.02	200	200
F	400	400	100.4	100.41	400	400
G	800	800	600.2	600.21	800	301
H1	100	100	100	100	0.2	100
H2	100	100	100	100	0.2	100
H3	100	100	100	100	0.2	100
H4	100	100	100	100	0.2	100
H5	100	100	100	100	0.2	100

Initial Con Ex.2 (With remote)

	x+	x-	y+	y-	z+	z-
A1	0	0.09	0	0	0.24	0.08
A2	0.09	0	0	0	0.24	0.08
B	0.05	0.05	0.05	0.05	0.24	0.02
C1	0.08	0.08	0.05	0.05	0.1	0.07
C2	0.08	0.08	0.05	0.05	0.07	0.1
D	0	0	0.05	0.05	0.19	0.06
E1	0	0	0.02	0.06	0	0
E2	0	0	0.06	0.02	0	0
F	0	0	0.1	0.11	0	0.09
G	0.03	0.03	0.08	0.09	0	0.23
H1	0	0	0	0	0.01	0.24
H2	0	0	0	0	0.04	0.21
H3	0	0	0	0	0.07	0.18
H4	0	0	0	0	0.1	0.15
H5	0	0	0	0	0.13	0.12

Training 2b

C	CoF	F
A1	5-10%	160
A2	5-10%	171
B	5-10%	150
C1	5-10%	171
C2	5-10%	130
D	5-10%	125
E1	15-25%	471
E2	15-25%	454
F	5-10%	175
G	5-10%	86
H1	15-25%	411
H2	15-25%	409
H3	15-25%	380
H4	15-25%	305
H5	15-25%	412

Compressed mean Feasibility 2b

	x+	x-	y+	y-	z+	z-
A1	200	200	200	200	200	-46
A2	200	200	200	200	200	-26
B	290	290	290	290	400	147
C1	84	84	84	84	75	70
C2	84	84	84	84	70	72
D	400	400	290	290	272	400
E1	200	200	3	200	200	200
E2	200	200	200	7	200	200
F	400	400	79	76	400	400
G	800	800	589	595	800	557
H1	100	100	100	100	32	100
H2	100	100	100	100	30	100
H3	100	100	100	100	38	100
H4	100	100	100	100	46	100
H5	100	100	100	100	42	100

Comp mean Confidence 2b (no remote)

	x+	x-	y+	y-	z+	z-
A1	0	0	0	0	0	0.041
A2	0	0	0	0	0	0.05
B	0.033	0.033	0.033	0.033	0	0.029
C1	0.026	0.026	0.026	0.026	0.015	0.011
C2	0.026	0.026	0.026	0.026	0.011	0.015
D	0	0	0.027	0.027	0.015	0
E1	0	0	0.037	0	0	0
E2	0	0	0	0.042	0	0
F	0	0	0.053	0.071	0	0
G	0	0	0.027	0.045	0	0.177
H1	0	0	0	0	0.034	0
H2	0	0	0	0	0.036	0
H3	0	0	0	0	0.037	0
H4	0	0	0	0	0.039	0
H5	0	0	0	0	0.038	0

Appendix 5: Example 3

R = 3001 (Run_size), Pc = 1 (Penalty cost)

Example 3

	Entire Passes/Fails: 62999 / 9041 (7:1)																					
Cycle no	**Sequences based on F_score:**																					
10	['N3'	'N2'	'W4'	'N1'	'W3'	'W6'	'B7'	'B4'	'B6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B2'	'W2'	'B1'	'W1'	'C2'	'S1'	'C1'	'S2']
10	['N3'	'N2'	'W4'	'N1'	'W3'	'W6'	'B7'	'B4'	'B6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B2'	'W2'	'B1'	'W1'	'C2'	'S1'	'C1'	'S2']
10	['N3'	'N2'	'W4'	'N1'	'W3'	'W6'	'B7'	'B4'	'B6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B2'	'W2'	'B1'	'W1'	'C2'	'S1'	'C1'	'S2']
	Sequences based on U_score:																					
705	['N3'	'N2'	'W4'	'B7'	'B6'	'B4'	'N1'	'W3'	'W6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B1'	'W1'	'S1'	'B2'	'C2'	'S2'	'W2'	'C1']
735	['N3'	'N2'	'W4'	'B7'	'B6'	'B4'	'N1'	'W3'	'W6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B1'	'W1'	'S1'	'B2'	'C2'	'S2'	'C1'	'W2']
689	['N3'	'N2'	'W4'	'B7'	'B6'	'B4'	'N1'	'W3'	'W6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B1'	'W1'	'B2'	'C2'	'S2'	'S1'	'C1'	'W2']
	Sequences based on F*U score:																					
890	['N3'	'N2'	'W4'	'B7'	'B6'	'N1'	'W3'	'B4'	'B5'	'W6'	'B3'	'W5'	'C4'	'C3'	'B1'	'W1'	'B2'	'C2'	'S2'	'S1'	'C1'	'W2']
638	['N3'	'N2'	'W4'	'B7'	'B4'	'B6'	'N1'	'W3'	'W6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B1'	'W1'	'B2'	'C2'	'S2'	'S1'	'C1'	'W2']
724	['N3'	'N2'	'W4'	'B7'	'B6'	'B4'	'N1'	'W3'	'W6'	'B3'	'W5'	'C4'	'B5'	'C3'	'B1'	'W1'	'B2'	'C2'	'S1'	'S2'	'W2'	'C1']
Frequency	**Dictionary of r_comps sequences:**																					
27	('N3'	'N2'	'W4'	'B7'	'B2'	'W2'	'B4'	'N1'	'W3'	'S2'	'B6'	'B1'	'C2'	'S1'	'W1'	'W6'	'B3'	'W5'	'C4'	'B5'	'C1'	C3')
26	('N3'	'N2'	'W4'	'B7'	'B2'	'W2'	'B4'	'S2'	'N1'	'W3'	'B6'	'B1'	'C2'	'S1'	'W1'	'W6'	'B3'	'W5'	'C4'	'B5'	'C1'	C3')
23	('N3'	'N2'	'W4'	'B7'	'B2'	'W2'	'B4'	'N1'	'W3'	'B6'	'S2'	'B1'	'C2'	'S1'	'W1'	'W6'	'B3'	'W5'	'C4'	'B5'	'C1'	C3')
	Dictionary of r_comps sequences with directions:																					
11	('N3x-'	'N2z-'	'W4z-'	'B1z+'	'W1z+'	'N1z-'	'W3z-'	'B7z-'	'B2z+'	'C2z-'	'S2z-'	'S1z-'	'W2z+'	'B4z-'	'B6z-'	'W6x-'	'B3x+'	'W5x+'	'C4x+'	'B5z-'	C1z-'	C3z+')
10	('N3x-'	'N2z-'	'W4z-'	'B7z-'	'B2z+'	'W2z+'	'B4z-'	'S2y+'	'N1z-'	'W3z-'	'B6z-'	'B1z+'	'C2z-'	'S1z-'	'W1z+'	'W6x-'	'B3x+'	'W5x+'	'C4x+'	'B5z-'	C1z-'	C3x+')
9	('N3x-'	'N2z-'	'W4z-'	'B7z-'	'B2z+'	'W2z+'	'B4z-'	'S2y-'	'N1z-'	'W3z-'	'B6z-'	'B1z+'	'C2z-'	'S1z-'	'W1z+'	'W6x-'	'B3x+'	'W5x+'	'C4x+'	'B5z-'	C1z-'	C3y-')

Initial Feasibility 3

	x+	x-	y+	y-	z+	z-
C1	601	601	601	601	900	802
S1	101	100	100	100	201	101
S2	100	101	100	100	201	100
C2	201	201	201	201	402	401
W1	101	101	100	100	101	102
W2	101	101	100	100	101	102
W3	101	100	100	100	202	100
W4	100	101	100	100	202	100
B1	602	600	600	600	3	600
B2	601	602	600	600	3	600
N1	201	200	200	200	202	0
N2	200	201	200	200	202	0
C3	602	602	501	501	500	605
C4	200	203	100	100	100	100
B3	1	500	501	501	500	505
W5	100	101	100	100	100	100
W6	201	100	100	100	100	100
N3	201	0	100	100	100	100
B4	200	200	200	201	200	1
B5	201	200	200	201	200	1
B6	200	200	201	200	200	1
B7	201	200	201	200	200	1

Initial Confidence 3

	x+	x-	y+	y-	z+	z-
C1	0.25	0.25	0.25	0.25	0.21	0.28
S1	0.16	0.1	0.1	0.1	0.1	0.15
S2	0.1	0.16	0.1	0.1	0.1	0.09
C2	0.2	0.2	0.2	0.2	0.14	0.18
W1	0.14	0.08	0.05	0.05	0.07	0.12
W2	0.08	0.14	0.05	0.05	0.07	0.12
W3	0.16	0.1	0.1	0.1	0.16	0.06
W4	0.1	0.16	0.1	0.1	0.16	0.06
B1	0.18	0.03	0	0	0.26	0
B2	0.06	0.18	0	0	0.26	0
N1	0.06	0	0	0	0.18	0.05
N2	0	0.06	0	0	0.18	0.05
C3	0.21	0.26	0.15	0.15	0.1	0.43
C4	0.01	0.3	0.1	0.1	0.1	0.1
B3	0.18	0	0.06	0.06	0	0.36
W5	0.01	0.14	0.05	0.05	0.05	0.05
W6	0.07	0.06	0.1	0.1	0.1	0.1
N3	0.09	0.05	0.05	0.05	0.05	0.05
B4	0.03	0.03	0	0.06	0	0.08
B5	0.06	0	0	0.06	0	0.08
B6	0.03	0.03	0.06	0	0	0.08
B7	0.06	0	0.06	0	0	0.08

Training data 3b

	CoF	Actual fails
C1	10-15%	347
S1	10-15%	431
S2	10-15%	400
C2	10-15%	416
W1	10-15%	402
W2	10-15%	428
W3	10-15%	405
W4	10-15%	437
B1	10-15%	412
B2	10-15%	316
N1	10-15%	440
N2	10-15%	424
C3	10-15%	194
C4	10-15%	395
B3	10-15%	450
W5	10-15%	447
W6	10-15%	450
N3	10-15%	418
B4	10-15%	434
B5	10-15%	442
B6	10-15%	434
B7	10-15%	438

Compressed mean Feasibility 3

	x+	x-	y+	y-	z+	z-
C1	604	604	604	604	894	795
S1	103	102	102	102	198	99
S2	100	101	100	100	198	98
C2	202	202	202	202	395	396
W1	102	101	101	101	98	102
W2	102	103	102	102	99	102
W3	103	102	102	102	200	96
W4	101	102	101	101	199	97
B1	602	600	600	600	1	600
B2	601	602	600	600	1	600
N1	201	200	200	200	202	-0
N2	200	201	200	200	202	-3
C3	602	601	505	505	504	608
C4	197	201	102	102	102	102
B3	3	500	501	501	500	505
W5	97	102	100	100	100	100
W6	197	102	103	103	103	103
N3	201	-4	100	100	100	100
B4	200	200	200	201	200	-0
B5	201	200	200	201	200	3
B6	200	200	201	200	200	0
B7	201	200	201	200	200	-2

Compressed mean Confidence 3

	x+	x-	y+	y-	z+	z-
C1	0.143	0.143	0.143	0.143	0.163	0.255
S1	0.118	0.058	0.058	0.058	0.106	0.135
S2	0.057	0.117	0.057	0.057	0.105	0.074
C2	0.114	0.114	0.114	0.114	0.149	0.15
W1	0.118	0.058	0.028	0.028	0.075	0.12
W2	0.059	0.119	0.029	0.029	0.076	0.12
W3	0.121	0.061	0.061	0.061	0.17	0.04
W4	0.057	0.117	0.057	0.057	0.164	0.045
B1	0.18	0.03	0	0	0.185	0
B2	0.06	0.18	0	0	0.186	0
N1	0.06	0	0	0	0.18	0.05
N2	0	0.06	0	0	0.18	0.038
C3	0.202	0.222	0.095	0.095	0.066	0.396
C4	0.016	0.281	0.056	0.056	0.056	0.056
B3	0.11	0	0.06	0.06	0	0.36
W5	0.014	0.117	0.027	0.027	0.027	0.027
W6	0.074	0.059	0.063	0.063	0.063	0.063
N3	0.09	0.036	0.027	0.027	0.027	0.027
B4	0.03	0.03	0	0.06	0	0.05
B5	0.06	0	0	0.06	0	0.061
B6	0.03	0.03	0.06	0	0	0.05
B7	0.06	0	0.06	0	0	0.045

Appendix 6: Pseudocode/Python Appendix

Algorithm 1: Python of compression function

```
(run, rise) = f.shape #rows, columns of source matrix
component = ["C1","C2","C3","C4","C5","f1","f2","f3","f4"]
john = list(range(len(component))) #List from 0 to total N components, increment 1

def compress(uncompressed):
    c = np.zeros((run, d_s))
    x = 0
    y = 0
    n = 0
    while x < run:
        while n < (len(john)):
            c[x,y] += uncompressed[x,((d_s*john[n])+y)]
            n+=1
        y += 1
        n = 0
        if y == d_s:
            x += 1
            y = 0
    return c
```

Algorithm 2: Python of the Penalty value matrix creation

```
[j] = np.where(r[x] == 1) #j is a list of row co-ordinates where 1's exist in Relation matrix
count_components = 0 #counts contacting components
coords = [] #co-ordinates of contacting relations
for z in j:
            count_components += 1
            coords.append([int(x),z])
            coords.append([z,int(x)])

penalty_matrix = np.zeros((run,rise)) #run-rise is the row-columns of source data
for [a,b] in coords:
            penalty_matrix[a,(b*d_s):(b*d_s+d_s)] += u[a,(b*d_s):(b*d_s+d_s)]
            #duplicating contacting parts confidence data (in all directions)
            penalty_matrix = penalty_matrix*penalty_cost

comp_p_matrix = compress(penalty_matrix)
f += penalty_matrix #penalty matrix applied to uncompressed feasibility
agg_penalty += penalty_matrix #storing all penalties to add to initial F in next cycle
```

Algorithm 3: Python of the Recursion Stopper

```
NIS = total + comp_p_matrix  #Next iteration summation
NIS_sorted = sorted(NIS[np.nonzero(NIS)].ravel())
            if NIS_sorted[0] in {NIS[thex, 0], NIS[thex, 1], NIS[thex, 2], NIS[thex, 3], NIS[thex, 4], NIS[thex, 5]}:
                recursion_stopper[thex] += 500 #to the whole component row
```

Algorithm 4: Pseudocode of Feasibility manipulation and balancer

```
Total the aggregate confidence matrices (no remote) and take mean
Create new duplicate matrix
Where confidence values exist (!=0):
      Create binary matrix
      Count number of non-zero elements

f+=(f_store+agg_penalty)-((sum1(agg_penalty)/ones_count)*dup_mean_agg_u)

Reset F_store
Reset agg_penalty
Store new f
```

Algorithm 5: Pseudocode for application of stochastic training data

```
component = ["C1","C2","C3","C4","C5","f1","f2","f3","f4"]
prob_array_min = [0.08, 0.08, 0.08, 0.08, 0.12, 0.08, 0.12, 0.12, 0.08]
prob_array_max = [0.12, 0.12, 0.12, 0.12, 0.20, 0.12, 0.20, 0.20, 0.12]

prob = choose a number between prob_array_min & prob_array_max
generate a random number between prob_array_min & prob_array_max
decision = rand_dec > prob

if decision is False: #Fail
if decision is True: #Pass
```

Appendix 7: Full code of Example 3. in Python

```
%%time
import collections
import numpy as np
from numpy import exp
import pandas as pd
import math
import matplotlib.pyplot as pl
import random

df = pd.read_excel('3.Contact.xlsx')
c = np.array(df)
print("Contact matrix:\n", c, "\n")

df = pd.read_excel('3.1.xlsx')
f = np.array(df).round(2)
print("3.1.xlsx:\n", f, "\n")

df = pd.read_excel('3.2.xlsx')
u = np.array(df).round(2)
print("3.2.xlsx:\n", u, "\n")

df = pd.read_excel('3.2noremote.xlsx')
noru = np.array(df).round(2)
print("no_remote_u.xlsx:\n", noru)

def sum1(input):
    return sum(map(sum, input))
def norm(x):
    # normalise x to range [-1,1]
    nom = (x - x.min())*2
    denom = x.max() - x.min()
    return  (nom/denom) - 1
def sigmoid(x, k):
    return 1/(1+exp(-x/k))
def compress(uncompressed):
    c = np.zeros((run, d_s))
    x = 0
    y = 0
    n = 0
    while x < run:
        while n < (len(john)):
            c[x,y] += uncompressed[x,((d_s*john[n])+y)]
            n+=1
        y += 1
        n = 0
        if y == d_s:
            x += 1
            y = 0
    return c
```

```
#USER ENTRY:
component = ["C1","S1","S2","C2","W1","W2","W3","W4","B1","B2","N1","N2","C3","C4","B3","W5","W6","N3","B4","
B5","B6","B7"]
direction = ["x+", "x-", "y+", "y-", "z+", "z-"]
prob_array_min = [0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0
.10, 0.10, 0.10, 0.00]
prob_array_max = [0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25,
0.25, 0.25, 0.25, 0.25]

d_s = 6 #directions
R = 5 #Run_size
Pc = 5 #Penalty cost
sigk = 0.27
#USER ENTRY END.

(run, rise) = f.shape #rows, columns
frm1 = np.zeros((run,rise)) #copy1 for ranking sequences
frm2 = np.zeros((run,rise)) #copy2 for ranking sequences
frm1 += f

john = list(range(len(component)))
print(john)

fi = compress(f)
u2 = compress(u)
no_ru = compress(noru)

print("\ninital Feasibility: sum:", sum1(fi).round(3), "\n", pd.DataFrame(data=fi, index=component, columns=direction))
print("\ninitial Confidence:\n", pd.DataFrame(data=u2, index=component, columns=direction), "\n")
print("\ninitial Confidence no remote:\n", pd.DataFrame(data=no_ru, index=component, columns=direction), "\n")
pd.DataFrame(data=fi.round(3), index=component, columns=direction).to_csv("ini_f3.csv")
pd.DataFrame(data=u2.round(3), index=component, columns=direction).to_csv("ini_u3.csv")
norm_fi = norm(fi)
norm_u2 = norm(u2)
pd.DataFrame(data=sigmoid(norm_fi,sigk).round(2), index=component, columns=direction).to_csv("ini_sig_f3.csv")
pd.DataFrame(data=sigmoid(norm_u2,sigk).round(2), index=component, columns=direction).to_csv("ini_sig_u3.csv")

run_size = 0 #initial
total_agg_u = np.zeros((run,rise)) #sum of all u matrices, after each cycle
total_agg_f = np.zeros((run,rise)) #sum of all f matrices, after each cycle
total_agg_noru = np.zeros((run,rise)) #sum of all f matrices, after each cycle
mean_agg_u = np.zeros((run,rise)) #^^divided by number of cycles
mean_agg_noru = np.zeros((run,rise)) #^^divided by number of cycles
mean_agg_f = np.zeros((run,rise)) #^^divided by number of cycles
aggregate_u = np.zeros((run,rise))
agg_penalty = np.zeros((run,rise))
f_store = np.zeros((run,rise))
c_store = np.zeros((run,rise))
no_ru_store = np.zeros((run,rise))
no_ru_store += noru
f_store += f
c_store += c

entire_passes = 0
entire_fails = 0
r_comps = []
f_comps = []

r_comps_list = []
r_comps_direction_list = []
r_comps_direction_sequences_list = []
#f_comps_events = []
```

```
f_comps_list = []

total_f_score = []
total_u_score = []
total_fxu_score = []

while True:
   if run_size > 0: #misses first pass
      entire_passes += passes
      entire_fails += total_fails

      r_comps_list += [r_comps] #creating total sequences list of 'lists'
      r_comps_direction_list += r_comps_direction
      r_comps_direction_sequences_list += [r_comps_direction]

      #f_comps_events.append([run_size, f_comps])
      f_comps_list += f_comps #List of all faluires

      total_f_score += [f_score] #ranking system list
      total_u_score += [u_score]
      total_fxu_score += [fxu_score]

      total_agg_f += aggregate_f

      total_agg_noru += agg_noru
      mean_agg_noru = total_agg_noru / run_size
      dup_mean_agg_u = np.zeros((run, rise))
      dup_mean_agg_u += mean_agg_noru #duplicate of mean agg_u
      non_zeros = dup_mean_agg_u != 0
      dup_mean_agg_u[non_zeros] = 1 #dup_mean_u is now identity matrix of confidence
      ones_count = np.count_nonzero(non_zeros == 1) #Counting one's in dup_mean_u (identity matrix) to divide agg pe
nalty by for steady state balancer
      f += (f_store + agg_penalty) - ((sum1(agg_penalty)/ones_count)*dup_mean_agg_u)#taking away equivalent value o
f penalties, to make room for fresh variable penalties

      f_store = np.zeros((run,rise)) #Re-setting F_store
      agg_penalty = np.zeros((run,rise)) #Re-setting agg_penalty
      f_store += f #Storing new F
      noru += no_ru_store #re-storing no remote u matrix
      total_agg_u += aggregate_u
      mean_agg_u = total_agg_u / run_size #gathering mean u so far, to then implement as current u

   run_size += 1
   u += mean_agg_u #initially no affect, later mean-a-u becomes u
   c = np.zeros((run,rise))
   c += c_store #re-storing contact relationships
   frm2 += frm1 #ranking uses a copy of initial feasiiblity: which will be removed piece by piece for online ranking

   if run_size == R:

      print("Entire Passes/Fails: ", entire_passes, "/", entire_fails)
      #print("\nAll sequences:\n", r_comps_list)
      mean_agg_f = total_agg_f / run_size
      mean_agg_noru = total_agg_noru / run_size

      comp_mean_agg_f = compress(mean_agg_f)
      comp_mean_agg_u = compress(mean_agg_u)
      comp_mean_agg_noru = compress(mean_agg_noru)

      print("\ncomp_mean_agg_f sum:", sum1(comp_mean_agg_f).round(3), "after", run_size, "cycles:\n", pd.DataFram
e(data=comp_mean_agg_f.round(3), index=component, columns=direction))
      print( "\n\ncomp_mean_agg_u after", run_size, "cycles:\n", pd.DataFrame(data=comp_mean_agg_u.round(3), ind
ex=component, columns=direction))
```

```
        print( "\n\ncomp_mean_agg_no_remote_u after", run_size, "cycles:\n", pd.DataFrame(data=comp_mean_agg_no
ru.round(3), index=component, columns=direction))

        norm_mean_f = norm(comp_mean_agg_f)
        norm_mean_u = norm(comp_mean_agg_u)
        print("\nsigmoid_of_mean_agg_f after", run_size, "cycles: (0 being the most removable, 1 being interlocked):\n", p
d.DataFrame(data=sigmoid(norm_mean_f,sigk).round(2), index=component, columns=direction))
        print("\nsigmoid_of_mean_agg_u after", run_size, "cycles: (0 being the confident, 1 being uncertain):\n", pd.DataF
rame(data=sigmoid(norm_mean_u,sigk).round(2), index=component, columns=direction))
        pd.DataFrame(data=sigmoid(norm_mean_f,sigk).round(2), index=component, columns=direction).to_csv("sig_f3b.c
sv")
        pd.DataFrame(data=sigmoid(norm_mean_u,sigk).round(2), index=component, columns=direction).to_csv("sig_u3b.
csv")
        pd.DataFrame(data=comp_mean_agg_f.round(3), index=component, columns=direction).to_csv("comp_mean_f3b.
csv")
        pd.DataFrame(data=comp_mean_agg_u.round(3), index=component, columns=direction).to_csv("comp_mean_u3
b.csv")
        pd.DataFrame(data=comp_mean_agg_noru.round(3), index=component, columns=direction).to_csv("comp_mean_
no_ru2b.csv")

        top_3 = sorted(total_f_score)[:3] #calculating the top 3 f scores and indexes
        index = []
        for k in top_3:
            index += [(total_f_score.index(k))]
        print("\ntop_3 and indicies:", top_3, index, "\ntop 3 sequences based on 'f_score' i.e. the sum of the feasibility of a
ll suggested parts which pass:\n", r_comps_list[index[0]], "\n", r_comps_list[index[1]], "\n", r_comps_list[index[2]])

        top_3u = sorted(total_u_score)[:3]
        undex = []
        for l in top_3u:
            undex += [(total_u_score.index(l))]
        print("\ntop_3u and indexes:", top_3u, undex, "\ntop_3u sequences based on 'u_score' i.e. the sum of the uncertai
ty of all parts which pass:\n", r_comps_list[undex[0]], "\n", r_comps_list[undex[1]], "\n", r_comps_list[undex[2]])

        top_3fu = sorted(total_fxu_score)[:3]
        fundex = []
        for w in top_3fu:
            fundex += [(total_fxu_score.index(w))]
        print("\ntop3_fu and indicies:", top_3fu, fundex, "\ntop_3f*u sequences based on 'f*u score':\n", r_comps_list[fun
dex[0]], "\n", r_comps_list[fundex[1]], "\n", r_comps_list[fundex[2]])

        r_comps_dict = collections.Counter([tuple(x) for x in r_comps_list])
        total_f_comps_dict = collections.Counter([f for f in f_comps_list if f])
        r_comps_direction_dict = collections.Counter([f for f in r_comps_direction_list if f])
        r_comps_direction_sequences_dict = collections.Counter([tuple(x) for x in r_comps_direction_sequences_list])
        print("\nDictionary of failed_comps:\n", total_f_comps_dict)
        print("\nDictionary of r_comps sequences:\n", r_comps_dict)
        #print("\nFailed comps events:\n", f_comps_events)
        print("\nDictionary of r_comps_directions:\n", r_comps_direction_dict)
        print("\nDictionary of r_comps sequences with directions:\n", r_comps_direction_sequences_dict)
        print("END")
        break

    passes = 0
    fails = 0
    total_fails = 0
    r_comps_direction = []
    r_comps = []
    f_comps = []
    #o_comps = []
    total = [[1,0,0],[0,0,0]]
    coords = 0
    count_components = 0
```

```
    aggregate_u = np.zeros((run,rise))
    aggregate_f = np.zeros((run,rise))
    agg_noru = np.zeros((run,rise))
    recursion_stopper = np.zeros((run, d_s))
    f_score = 0
    u_score = 0
    fxu_score = 0

    while True:
        #if run_size > 0:
            #print("\n Sum of summation:", sum1(total))
        if sum1(total) == 0:
            for item in component:
                if not item in r_comps:
                    r_comps.append(item)
                    #o_comps.append(item + ":P")
            #print("passes, total fails", passes, total_fails)
            #print(r_comps, "removed in chronological order")
            #print(f_comps, "failed")
            #print(o_comps, "Disassembly story")
            #print("\n\n\n********************   END of CYCLE:  ************************", run_size, "\n\n\n")
            break

        while True:
            r = np.zeros((run, run))
            x = 0
            y = 0
            while x < run:
                r[x,y] = c[x,(d_s*y)] or c[x,(d_s*y)+1] or c[x,(d_s*y)+2] or c[x,(d_s*y)+3] or c[x,(d_s*y)+4] or c[x,(d_s*y)+5]
                y += 1
                if y == run:
                    x += 1
                    y = 0
            #print("\nrelation matrix:\n", r) #taking rowXcolumn contact matrix and scaling to rowXrow relation matrix

            s1 = np.zeros((run, rise))
            s2 = np.zeros((run, rise))
            N = 1000
            i = 0
            j = 0
            while i < run:
                dist = u[i,j]*np.random.normal(size=N) #iterating 10000x within confidence tolerance-Monte-Carlo propagation
                s1[i,j] = dist.mean() #S1 - mean of normal distrobution for U
                s2[i,j] = 2*dist.std() #S2 - standard deviation
                j += 1
                if j == rise:
                    i += 1
                    j = 0

            fi = compress(frm2) #Compressed initial F
            f1 = compress(f) #Compressed SS Feasability
            s1_1 = compress(s1) #Compressed Mean of 10000 samples(row x directions)
            s2_2 = compress(s2) #Compressed Mean Agg Uncertainty± 2*standard deviation

            total = f1 + s1_1 + s2_2 + recursion_stopper #summing the feasibility, uncertainty mean & standard deviation arr
ays + recusion stopper
            recursion_stopper = np.zeros((run, d_s))
            np.set_printoptions(suppress=True)
            #print("\npasses;fails;totalfails:", passes, fails, total_fails, "\nSummation matrix: (f1 + s1_1 + s2_2 + recursion_sto
pper)\n", total.round(2))
            if sum1(total) == 0: #when there is one component left sum of summation will equal zero
                break
```

```
        #print(*np.where(total == np.min(total[np.nonzero(total)])), np.min(total[np.nonzero(total)]))# minval = np.min(t
otal[np.nonzero(total)])
        #x value here is vital to the "pass" stage

        [x,y] = np.where(total == np.min(total[np.nonzero(total)]))
        index = np.random.randint(0,len(x))
        x = x[index]
        y = y[index]
        #print("\n\nparts already removed:", r_comps, "\nparts failed ", f_comps, "\nOverall Story:", o_comps)
        #print("\n---------Removing component:", component[x], "in direction", direction[int(y)])

        a = int(x*d_s)
        [j] = np.where(r[x] == 1) #j is a list of row coords in relation where 1's exist
        thex = x #copying x & y for later use
        they = y

        count_components = 0 #counts contacting components
        coords = [] #coordinates of contacting relations
        for z in j:
            count_components += 1
            coords.append([int(x),z])
            coords.append([z,int(x)]) #need both [x,y] and [y,x]

        ys = [d_s*g[1] for g in coords] #Compiling and scaling y values from size row to rise
        xs = [x[0] for x in coords] #list comprehension, g is used to represent y here
        all_ys = []
        for n in ys:
            all_ys += n, n+1, n+2, n+3, n+4, n+5

        i = 0 #used to cycle through the associated y values
        confidence_coords = [] #coords used to update the confidence matrix
        for x in xs:
            confidence_coords += [x,all_ys[i]], [x ,all_ys[i+1]], [x ,all_ys[i+2]], [x, all_ys[i+3]], [x ,all_ys[i+4]], [x, all_ys[i+5]]
            i += d_s #x+, x-, y+, y-, z+, z-

        #print(thex, prob_array_min[0], prob_array_max[0]) #looking at the 'training data' to see pass or fail
        prob = random.uniform(prob_array_min[thex], prob_array_max[thex])
        rand_dec = random.random() #random number between 0 and 1
        decision = rand_dec > prob

        if decision is False: #Fail
            f_comps.append(component[int(thex)]) #appending failed comp to failed comps list
            #o_comps.append(component[int(thex)] + ":F") #appending failed comp to the story
            #print("parts failed:", f_comps)
            fails += 1
            total_fails += 1

            penalty_matrix = np.zeros((run,rise))
            penalty_cost = Pc #Penalty_cost: i.e. the Weight applied
            for [a,b] in coords:
                penalty_matrix[a,(b*d_s):(b*d_s+d_s)] += u[a,(b*d_s):(b*d_s+d_s)]
            #copying over all contacting parts confidence into blank matrix
            penalty_matrix = penalty_matrix*penalty_cost
            agg_penalty += penalty_matrix #storing all penalties to add to initial f in next cycle

            comp_p_matrix = compress(penalty_matrix)
            #print("\n", "compressed penalty matrix:\n", comp_p_matrix)
            f += penalty_matrix

            NIS = total + comp_p_matrix  #Next iteration summation
            NIS_sorted = sorted(NIS[np.nonzero(NIS)].ravel())
            if NIS_sorted[0] in {NIS[thex, 0], NIS[thex, 1], NIS[thex, 2], NIS[thex, 3], NIS[thex, 4], NIS[thex, 5]}:
                recursion_stopper[thex] += 500 #to the whole component row
```

```
            for z,p in confidence_coords:
                if u[z,p] != 0:
                    u[z,p] = u[z,p]*exp(0.01/u[z,p]) #the function deciding the increase in uncertainty
                if noru[z,p] != 0:
                    noru[z,p] = noru[z,p]*exp(0.01/noru[z,p]) #doing the same for no remote u for clarity in end results
            #print("confidence_coords::", confidence_coords)
            #print("Overall story:", o_comps, "\n", u.round(3))

        if decision is True: #Pass
            f_score += fi[thex, they] #initial F matrix with parts removed and compressed, not a dynamic matrix, will allow f
or unique scores and creative solutions
            #print("F_SCORE:", f_score)
            u_score += s2_2[thex, they] #scoring 2*standard deviation of mean confidence
            #print("U_SCORE:", u_score)
            fxu_score += (fi[thex, they])*(s2_2[thex, they])
            #print(decision, "min/max:", prob_array_min[thex], prob_array_max[thex],"random_dec:", rand_dec, "chosen_
faluire_prob:", prob)
            r_comps.append(component[int(thex)])
            #o_comps.append(component[int(thex)] + ":P")
            #print("parts removed:", r_comps)
            r_comps_direction.append(component[int(thex)] + direction[int(they)])
            passes += 1
            fails = 0

            for x,g in confidence_coords: #taking a blanket 10% off u[x,g] for passing
                if u[x,g] >= 1/90:
                    u[x,g] = u[x,g]*0.9
                if noru[x,g] >= 1/90:
                    noru[x,g] = noru[x,g]*0.9

            aggregate_u += u #adding the entire u matrix to the aggregate matrix
            aggregate_f += f #adding the entire f matrix to the aggregate matrix
            agg_noru += noru #'' No Remote U

            frm2[thex] = 0 #used as ranking feasibilty matrix
            for col in frm2:
                col[a:(a+d_s)] = 0
            noru[thex] = 0
            for col in noru:
                col[a:(a+d_s)] = 0
            #print("\nSequence", passes, "NO REMOTE uncertainty matrix after", component[int(thex)], "removed:\n", u)
            f[thex] = 0
            for col in f:
                col[a:(a+d_s)] = 0
            #print("\nSequence", passes, "Feasability matrix after component", component[int(thex)], "removed:\n", f)
            u[thex] = 0
            for col in u:
                col[a:(a+d_s)] = 0
            #print("\nSequence", passes, "Uncertainty matrix after component", component[int(thex)], "removed:\n", u)
            c[thex] = 0
            for col in c:
                col[a:(a+d_s)] = 0
            #print("\nSequence", passes, "contact matrix after", component[int(thex)], "removed:\n", c)

            aggregate_u -= u #removing 'confidence' of components that are yet to be removed
            aggregate_f -= f #removing final 'feasability' of components that are yet to be removed
            agg_noru -= noru

            #f_comps_dict = collections.Counter(f for f in f_comps if f)
            #print("\nDictionary of faluires:", f_comps_dict)
            break
```