TeMA: a Tensorial Memetic Algorithm for Many-Objective Parallel Disassembly Sequence Planning in Product Refurbishment

Francesco Pistolesi, and Beatrice Lazzerini, *Member, IEEE*

***Abstract*—The refurbishment market is rich in opportunities, the global refurbished smartphones market alone will be $38.9 billion by 2025. Refurbishing a product involves disassembling it to test the key parts and replacing those that are defective or worn. This restores the product to like-new conditions, so that it can be put on the market again at a lower price. Making this process quick and efficient is crucial. This paper presents a novel formulation of parallel disassembly problem that maximizes the degree of parallelism, the level of ergonomics, and how the workers' workload is balanced, while minimizing the disassembly time and the number of times the product has to be rotated. The problem is solved using the Tensorial Memetic Algorithm (TeMA), a novel two-stage many-objective (MaO) algorithm which encodes parallel disassembly plans by using third-order tensors. TeMA first splits the objectives into *primary* and *secondary* on the basis of a decision maker's preferences, and then finds Pareto-optimal compromises (*seeds*) of the primary objectives. In the second stage, TeMA performs a fine-grained local search that explores the objective space regions around the seeds, to improve the secondary objectives. TeMA was tested on two real-world refurbishment processes involving a smartphone and a washing machine. The experiments showed that, on average, TeMA is statistically more accurate than various efficient MaO algorithms in the decision maker's area of preference.**

***Index Terms*—Disassembly, Evolutionary computation, Many-objective optimization, Memetic algorithm, Product refurbishment, Tensor.**

I. Introduction

TODAY, the refurbishment market for consumer electronics is estimated to be $10 billion and it shows no sign of slowing down [1]. According to the Global E-waste Monitor, the volume of end-of-life electronics generated in the USA is 7.1 million tonnes every year. Refurbishing products can help reduce this environmental impact for years to come.

When thinking about refurbished electronics, people imagine products with some visible defect, which operate at degraded performance. In reality, refurbished products are restored to like-new conditions, and this provides consumers with quality products at a more affordable price. Companies generally list refurbished products offering savings that range from 15% to 30% off the original price.

Many leading global brands are entering the refurbished market, thus gaining new customers that previously could not access the brand on the primary market. Manufacturers and researchers have thus increased their interest in refurbishment, proposing new standard methodologies [2]. At the same time, closed-loop fashion supply chains are modeled to optimize the service rates and refurbishment facilities [3].

As soon as products are returned to companies, they can enter the refurbishment cycle. The first step is to disassemble the products, in order to test specific components and ensure quality standards, before putting them again up for sale. Finding the best way to disassemble all or part of these components is a *Disassembly Sequence Planning (DSP) problem*. The DSP is an *NP*-hard combinatorial problem [4] with multiple objectives, typically more than three. This makes the DSP a *many-objective (MaO)* problem.

Disassembly processes are key in the industry for refurbishment, end-of-life dismantling, remanufacturing and so on, and various purposely-designed algorithms have been proposed in the last years. Due to the high complexity of the problem, these approaches are widely based on evolutionary computation, such as ant colony optimization [5], tabu search [6] and genetic algorithms (GAs) [7]. Hybrid techniques have also been proposed. For instance, in [8] a GA is hybridized with extremal optimization. Scatter search is used with Petri nets in [9], and with dual objective program in [10]. A hybrid algorithm integrating fuzzy simulation and artificial bee colony optimization is proposed in [11]. An artificial bee colony optimization algorithm is finally used in [12].

Although these approaches are interesting and efficient, they limit the number of objectives, calculate scalarizations of the objectives, or both. When the objectives are scalarized, certain solutions that better represent a decision maker's preferences might be impossible to find [13]. On the other hand, limiting the number of objectives is detrimental, as real-world problems typically deal with many objectives. Moreover, the previous approaches, as well as most techniques for DSPs in the literature, focus on sequential disassembly, where workers remove one component at a time. This is quite inefficient. Parallel disassembly is becoming of interest, as it employs several workers to simultaneously remove several components, thus making the process considerably faster [14].

The contribution of this paper is twofold, as it proposes:

1) *an MaO formulation of a novel parallel DSP* that minimizes the disassembly time and the number of times the product is rotated, while maximizing the degree of parallelism, the way the workers' workload is balanced, and the workers' level of ergonomics;

Manuscript received July 31, 2018; revised November 30, 2018; accepted February 26, 2019. Date of publication ; date of current version . Paper no. . *(Corresponding author: Francesco Pistolesi.)*

F. Pistolesi and B. Lazzerini are with the Department of Information Engineering, University of Pisa, Largo L. Lazzarino 1, 56122 Pisa, Italy, (email: f.pistolesi@iet.unipi.it; b.lazzerini@iet.unipi.it)

2) *the Tensorial Memetic Algorithm (TeMA)*, an MaO algorithm that blends a genetic approach with a local search process to go straight to the decision maker's area of preference and discover Pareto-optimal solutions that put the most important objectives first.

The paper is structured as follows: Section II contains a background on MaO, memetic algorithms and GAs; Section III outlines the formulation of the problem; Section IV presents TeMA; Section V discusses the experiments; Section VI describes the performance evaluation. Section VII draws the conclusions.

II. Background

A. Many-objective optimization (MaOO)

MaOO problems are multi-objective problems with more than three objectives, formulated as $\max_{\mathbf{x}\in\mathcal{X}} \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}),\ldots,f_m(\mathbf{x})]$, s.t. $m \geq 4$, where $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \leq 0, h_j(\mathbf{x}) = 0, \forall i = 1,\ldots,G, \forall j = 1,\ldots,H\}$ is the feasible region, where G and H are the number of inequality and equality constraints, respectively. Vector function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ contains the objective functions. Given $\mathbf{x^1}, \mathbf{x^2} \in \mathcal{X}$, solution $\mathbf{x^1}$ *Pareto-dominates* $\mathbf{x^2}$ if $f_i(\mathbf{x^1}) \geq f_i(\mathbf{x^2})$, $\forall i \in \{1,\ldots,m\}$, and $\exists\, j \in \{1,\ldots,m\}$ s.t. $f_j(\mathbf{x^1}) > f_j(\mathbf{x^2})$. Non-dominated solutions are *Pareto-optimal* and form the *Pareto front*, in the objective space.

MaOO problems are hard because the number of non-dominated solutions becomes huge when the number of objectives is increased [15]. Using exact methods is not practical because: i) they are not applicable (e.g., due to non-differentiable objective functions); ii) they fail to converge, as in the case of real-world problems, which typically have huge quantities of variables, objective functions and constraints; iii) they involve a large set of Pareto-optimal solutions in order to find one that faithfully represents the decision maker's preferences [13].

Evolutionary algorithms are very efficient in multi-objective optimization. However, their performance degrades considerably when dealing with MaOO problems because of the high dimensionality, which causes a huge increase in non-dominated solutions, makes crossover and mutation less efficient, and requires a higher computational power [16]. Various heuristic algorithms have thus been proposed which are based on preference ordering relations, objective reduction, preference incorporation, or decomposition: no single solution has been shown to be superior to the others [17].

B. Genetic algorithms (GAs)

GAs mimic evolution [18] to solve complex problems. Potential solutions are encoded as data structures (*individuals*) made up of binary/integer/real elements (*genes*). One or more *fitness functions* assess each individual's goodness.

A GA first generates a *population* of individuals. The better the fitness, the more likely an individual is selected for reproduction. Those selected evolve via *crossover* and *mutation*. Crossover mixes the genetic information of several individuals to get one or more *offspring*. Mutation makes slight changes to some, as occurs throughout the evolution of a species. The offspring replace part of the population on the basis of the fitness. GAs iterate until a stop condition is met.

C. Memetic algorithms (MAs)

MAs are one of the most recent growing areas in evolutionary computation [19]. MAs are population-based metaheuristics that are made up of an evolutionary core equipped with a set of local search algorithms that work alongside the evolutionary core to find better solutions [20]. Although pure evolutionary algorithms are not well suited to fine-grained search in complex combinatorial spaces, hybridization with other techniques can improve the search efficiency [21].

MAs can tackle many complex optimization problems across a wide range of industrial application areas, such as assembly [22], vehicle routing [23], real-time production scheduling [24], and wireless sensor networks [25]. In all cases, MAs were shown to be more efficient and/or converge to better solutions compared to their purely-evolutionary counterparts.

III. Problem formulation

A. Basic concepts

Consider a product with F *faces* that lies on one of these (the *ground face*), which cannot be accessed. Consider W workers who, if necessary, can work in parallel. Refurbishment policies entail removing and testing some parts of the product.

Let $\mathcal{T} = \{1,\ldots,T\}$ be the set of the *disassembly tasks*, each removing one or more parts from the product. Let $\mathcal{T}^{TARGET} \subseteq \mathcal{T}$ be the set of the *target tasks*, i.e., those that remove the parts to test. A task i has a duration d_i. In order to perform a given task, a worker must act on a specific face. If this is the ground face, the task entails rotating the product. Also, all *preceding tasks* must have already been performed. These tasks determine the *precedence constraints (PCs)*.

Let b be a *branch*, i.e. a sequence of tasks that meets the PCs, which is performed by one worker and may be executed in parallel with other branches, if both of the following hold:

1) the tasks of the branches do not act on the ground face;
2) no branch contains any preceding task of the tasks that make up the other branches.

A group of branches executed in parallel forms a *phase*, and a series of phases forms a *disassembly schedule (DS)*. A DS may contain phases with just one branch. The maximum number of branches per phase is equal to the number W of workers.

B. Notation

Let vector $\mathbf{s} \in \{0,1\}^{T\times T\times B\times P}$ be a DS, where T is the number of tasks to disassemble the product entirely, B is the maximum number of branches per phase, and P is the maximum number of phases of a DS. Note that $B = W$ and $P = T$. The elements of $\mathbf{s}$ are in lexicographic order, and the generic element is:

$$s_{ikbp} = \begin{cases} 1 & \text{if } i \text{ is the } k\text{-th task in branch } b \text{ of phase } p \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Let $\mathbf{C} = [c_{ij}]$ be a $T \times T$ matrix where $c_{ij} = 1$ if task i must be performed before task j, and $c_{ij} = 0$ otherwise.

C. Model

The proposed parallel DSP problem is formulated as:

$$\max_{\mathbf{s}} \mathbf{f}(\mathbf{s}) = [-f_1(\mathbf{s}), -f_2(\mathbf{s}), f_3(\mathbf{s}), f_4(\mathbf{s}), -f_5(\mathbf{s})] \tag{2a}$$

subject to:

$$\sum_{p=1}^{P}\sum_{b=1}^{B}\sum_{k=1}^{T} s_{ikbp} \le 1, \quad \forall i = 1, \dots, T \tag{2b}$$

$$\sum_{p'<p}\sum_{b'=1}^{B}\sum_{k'=1}^{T}\sum_{h:c_{hi}=1} s_{hk'b'p} + \sum_{k'<k}\sum_{h:c_{hi}=1} s_{hk'bp} - \sum_{h\neq i} c_{hi} = 0, \quad \forall i, \forall k, \forall b, \forall p \tag{2c}$$

$$\sum_{p=1}^{P}\sum_{b=1}^{B}\sum_{k=1}^{T} s_{ikbp} = 1, \quad \forall i \in \mathcal{T}^{TARGET} \tag{2d}$$

$$\sum_{b'>b}\sum_{k=1}^{T}\sum_{i=1}^{T} s_{ikb'p} = 0, \quad \forall p, \forall b : \textstyle\sum_k \sum_i s_{ikbp} = 0 \tag{2e}$$

$$s_{ikbp} \in \{0,1\}, \ \forall i, \forall k, \forall b, \forall p. \tag{2f}$$

1) Objective functions: Equation (2a) contains the objective functions, whose formulations are given in the next sections.

a) Number of rotations: Function $f_1(\mathbf{s})$ counts how many times the product is rotated to perform $\mathbf{s}$. There is more than one sequence of rotations that allows access to the faces required while performing $\mathbf{s}$. Function $f_1(\mathbf{s})$ finds a sequence with the lowest number of rotations, and takes this number as a value.

This nested optimization problem can be formulated as the one of finding the shortest path from a node s to a node d in a layered graph, called *ground face graph* (GFG). A GFG has the same number of layers as the number of phases of $\mathbf{s}$. Each layer p is associated with phase p of $\mathbf{s}$, and has the same number of nodes as there are faces that are not required by phase p. These faces can be the ground face during p. Each node of a layer is connected to all nodes of the next layer. The arc that connects two nodes u and v has a cost $r^{\mathbf{s}}_{uv} \in \{0,1,2\}$ that corresponds to the number of rotations required to change the ground face from u to v. The product needs no rotation if $u = v$, 1 rotation if u and v are adjacent, and 2 rotations if u and v are parallel. An example of GFG is shown in Fig. 1.

Let $\mathcal{A}^{\mathbf{s}}$ and $\mathcal{N}^{\mathbf{s}}$ be the sets of the arcs and nodes, respectively. The objective function can be modeled as:

$$f_1(\mathbf{s}) = \begin{cases} \min \displaystyle\sum_{(u,v)\in\mathcal{A}^{\mathbf{s}}} r^{\mathbf{s}}_{uv} x_{uv} \\ \displaystyle\sum_{(u,z)\in\mathcal{A}^{\mathbf{s}}} x_{uz} - \sum_{(z,v)\in\mathcal{A}^{\mathbf{s}}} x_{zv} = \begin{cases} -1, & z = s \\ 1, & z = d \\ 0, & z \in \mathcal{N}^{\mathbf{s}} \setminus \{s,d\} \end{cases} \\ x_{uv} \in \{0,1\} \ \forall (u,v) \in \mathcal{A}^{\mathbf{s}}. \end{cases} \tag{3}$$

b) Disassembly time: Function $f_2(\mathbf{s})$ measures the time required to perform $\mathbf{s}$, which depends on the longest branch of each phase, and can be calculated as follows:

$$f_2(\mathbf{s}) = \sum_{p=1}^{P} \max_{b=1}^{B} \left(\sum_{i=1}^{T}\sum_{k=1}^{T} d_i s_{ikbp} \right). \tag{4}$$

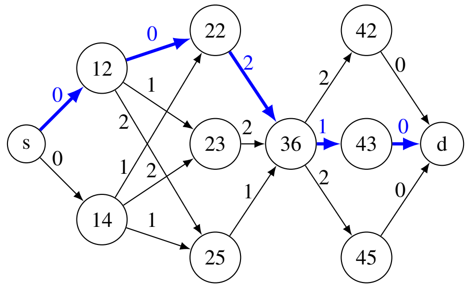

Figure 1. GFG of a DS with four phases. The label of each node is made up of two ciphers: the first one is the phase id, the second is the face id, assuming a product with six faces. For example, the first layer (nodes 12 and 14) indicates that faces 2 and 4 are not required by the tasks in phase 1. The path in blue is an example of sequence that minimizes the number of rotations, where the ground faces, one per phase, are: 2, 2, 6, and 3, in order.

c) Level of ergonomics: Function $f_3(\mathbf{s})$ is the workers' level of ergonomics when performing $\mathbf{s}$. Each task acts on a specific face of the product. While task i is being performed by one worker on a face, other workers may be performing other tasks on that face, or on parallel/adjacent faces. At any instant, the more the workers that are performing tasks on the same face, the less comfortable they are, as they are too close to each other and get in each other's way. Conversely, the less close they are on the faces that are working on, the higher the ergonomics.

The level of ergonomics is measured by first discretizing time into intervals with length $\Delta\tau$. The set of the tasks of $\mathbf{s}$ performed in each time interval $[\tau, \tau + \Delta\tau]$ is

$$\mathcal{T}_\tau = \{i : s_{ikbp} = 1 \wedge \tau \le D^{p^i} + D^{k^i} + d_i\}, \tag{5}$$

where terms $D^{p^i} = \sum_{p<p^i} \max_{b=1}^{B}(\sum_{k=1}^{T}\sum_{i=1}^{T} d_i s_{ikbp})$ and $D^{k^i} = \sum_i \sum_{k<k^i} d_i s_{ikb^ip^i}$ are, respectively, the total duration of the phases that precede the one where task i is performed (i.e., p^i), and the total duration of the tasks that precede i in the same branch where i is performed (i.e., b^i).

Each task $i \in \mathcal{T}_\tau$ is assigned a maximum level of ergonomics if it is the only one being performed, or there are only tasks $j \neq i$ being performed on the parallel face of the product. For each task j being performed on the same face as that of i, or on adjacent faces, the level of ergonomics of i is decreased by a penalty coefficient $q_{ij} \ge 0$. This coefficient has two levels of penalty. The high level is used for the tasks j performed on the same face of i; the low level is used for the tasks performed on the faces adjacent to that of i.

Let $D^{\mathbf{s}} \in \mathbb{R}^+$ be the duration of $\mathbf{s}$, and let φ_i be the face on which task i operates. The formulation of the objective is

$$f_3(\mathbf{s}) = \frac{D^{\mathbf{s}}}{\sum_{\tau=0}^{D^{\mathbf{s}}} 1/\exp\left(\sum_{i\in\mathcal{T}_\tau}\sum_{j\in\mathcal{T}_\tau, j\neq i} q_{ij}\right)}, \tag{6}$$

where $q_{ij} = 0$, $q_{ij} = 1$, or $q_{ij} = 1.5$ if φ_i and φ_j are parallel, adjacent, or are the same face, respectively. Eq. (6) calculates the harmonic mean.

d) Degree of parallelism: Function $f_4(\mathbf{s})$ measures the degree of parallelism. The average number of workers employed per phase is used to measure the degree of parallelism:

$$f_4(\mathbf{s}) = \frac{1}{P}\sum_{p=1}^{P}\left(\frac{\sum_{b=1}^{B}\max_{i,k=1}^{T} s_{ikbp}}{W}\right). \tag{7}$$

e) Branch smoothness: Function $f_5(\mathbf{s})$ measures how leveled the idle times are of the branches of all phases of s. The idle time of branch b of phase p is the time that, once finished, the associated worker has to wait until the longest branch of phase p is completed. The leveling is measured as the sum of the squared idle times of all branches of all phases:

$$f_5(\mathbf{s}) = \sum_{p=1}^{P}\sum_{b=1}^{B}\left(\max_{b'=1}^{B}\left(\sum_{k=1}^{T}\sum_{i=1}^{T} d_i s_{ikb'p}\right) - \sum_{k=1}^{T}\sum_{i=1}^{T} d_i s_{ikbp}\right)^2. \tag{8}$$

2) Constraints: Constraints (2b) ensure that each task is performed at most in one branch of one phase. Constraints (2c) prevent each task i from being performed as k-th in branch b of phase p if its preceding tasks are not performed either in the same branch at positions $k' < k$, or in a branch of the previous phases $p' < p$. Constraints (2d) force each target task in $\mathcal{T}^{TARGET}$ to be performed. Constraints (2e), given a phase p, guarantee that an empty branch is preceded by one or more non-empty branches. Constraints (2f) force binary variables.

IV. The Tensorial Memetic Algorithm: TeMA

This section describes TeMA, the novel two-stage memetic algorithm to solve the problem presented in Section III-C.

A. Encoding

A DS is encoded using a third-order tensor $\mathbf{T} \in \{0, \ldots, T\}^{S\times S\times W}$, where $S = \lfloor \frac{T}{W} \rfloor$. This is an evolution of the permutation encoding—typically used in combinatorial problems,—that encodes a chromosome as a vector of integers, without duplicates. Given that problem (2a)-(2f) introduces parallelism, there is not a single sequence of tasks, but there are multiple sequences of tasks (branches) performed in parallel. These branches are then followed by further branches which, in turn, are performed in parallel with other branches, and so on. A third-order tensor can easily map this situation and can guarantee efficient indexing and manipulation. These aspects are key to obtaining a fast recombination and a fast vectorized fitness evaluation, which boost the evolution.

Each element t_{ijk} of $\mathbf{T}$ either contains a task identifier or is empty. The non-empty element t_{ijk} contains the identifier of the task performed as j-th in branch k of phase i. Non-empty elements contain task identifiers, whereas empty elements contain 0 and are key throughout the evolution in terms of moving tasks from branch to branch, and phase to phase. This gives the algorithm the power to explore the largest possible set of combinations and parallelizations of the tasks. An example of a chromosome is shown in Fig. 2, which can be written as

$$\mathbf{T} = [(\langle 1,2\rangle, \langle 6\rangle, \langle 11\rangle), (\langle 3\rangle, \langle 5,7\rangle), (\langle 4\rangle, \langle 8,9,10\rangle)], \tag{9}$$

where $\langle\cdot\rangle$ denotes a branch, and $(\cdot)$ denotes a phase.

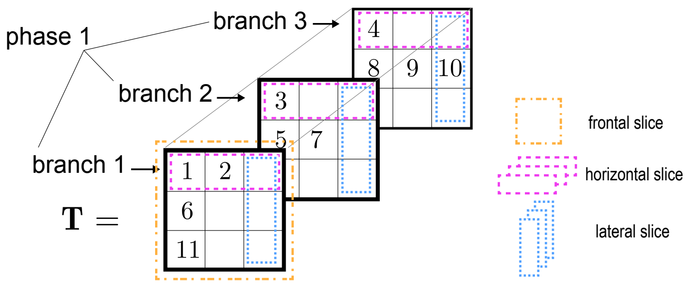

Figure 2. Tensor that encodes a DS with three phases: the first one has three branches that respectively perform tasks $\langle 1,2\rangle$, $\langle 3\rangle$, and $\langle 4\rangle$; the second one has three branches, performing tasks $\langle 6\rangle$, $\langle 5,7\rangle$, and $\langle 8,9,10\rangle$, respectively; the third phase has one branch that performs task $\langle 11\rangle$.

B. Decoding

In order to decode a tensor $\mathbf{T}$ into the corresponding DS s, TeMA performs the steps of Algorithm 1.

Algorithm 1: Decoding tensor $\mathbf{T}$ into schedule s

```
input  : $\mathbf{T}, S, T, B, P$
output: s
s := $0^{T\times T\times B\times P}$;
for $w := 1$ to $P$ do        /* ∀ horizontal slice (phase) $w$ of $\mathbf{T}$ */
    for $u := 1$ to $S$ do        /* ∀ row fiber (branch) $u$ of $w$ */
        for $v := 1$ to $S$ do        /* ∀ column (position) $v$ of $u$ */
            if $t_{uvw} \neq 0$ then
                $i := t_{uvw}$;  $k := v$;  $b := u$;  $p := w$;
                $s_{ikbp} := 1$;
            end if
            $v := v + 1$;
        end for
        $u := u + 1$;
    end for
    $w := w + 1$;
end for
return s
```

C. Tensorial genetic operators

TeMA evolves the individuals by means of tensorial genetic operators that were purposely developed to work with the third-order tensors explained in the previous section.

1) 3-mode Tensorial Crossover (3TX): The 3TX operates in three modes: *cut&paste*, *replacement* and *slicing*. The mode used by the 3TX is chosen randomly. The following tensors are used in the next sections to explain the three modes:

$$\mathbf{T} = \left[\begin{bmatrix}1&2&0\\6&0&0\\11&0&0\end{bmatrix}\begin{bmatrix}3&0&0\\5&7&0\\0&0&0\end{bmatrix}\begin{bmatrix}4&0&0\\8&9&10\\0&0&0\end{bmatrix}\right], \mathbf{S} = \left[\begin{bmatrix}1&0&0\\4&6&0\\0&0&0\end{bmatrix}\begin{bmatrix}2&3&5\\7&0&0\\8&11&0\end{bmatrix}\begin{bmatrix}0&0&0\\9&10&0\\0&0&0\end{bmatrix}\right]$$

where frontal slices are in square brackets, and t_{ijk} is the element of $\mathbf{T}$ in row i, column j and frontal slice k, and the i-th row of the k-th frontal slice is denoted as $t_{i:k}$.

a) Cut&paste: This mode first selects one row $t_{i^t:k^t}$ of $\mathbf{T}$, one row $s_{i^s:k^s}$ of $\mathbf{S}$, and a cut-off index $h \in \{1, \dots, S\}$. Two offspring $\mathbf{o^1}$ and $\mathbf{o^2}$ are generated by exchanging the second part of the genetic information of the rows between the two chromosomes, so that $o^1_{i^t j k^t} = s_{i^s j k^s}$ and $o^2_{i^s j k^s} = t_{i^t j k^t}, \forall j > h$. The other genes of $\mathbf{o^1}$ and $\mathbf{o^2}$ are inherited from $\mathbf{T}$ and $\mathbf{S}$, respectively. The genes of $o^1_{i^t:k^t}$ and $o^2_{i^s:k^s}$ with $j > h$ form the *crossing region* of $\mathbf{o^1}$ and $\mathbf{o^2}$, respectively.

For example, consider rows $t_{2:1}$ and $s_{1:2}$, and $h = 1$. The offspring are

$$\mathbf{o^1} = \left[\begin{bmatrix} 1 & 2 & 0 \\ 6 & \mathbf{3} & \mathbf{5} \\ 11 & 0 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 \\ 5 & 7 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 \\ 8 & 9 & 10 \\ 0 & 0 & 0 \end{bmatrix}\right], \mathbf{o^2} = \left[\begin{bmatrix} 1 & 0 & 0 \\ 4 & 6 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & \mathbf{0} & \mathbf{0} \\ 7 & 0 & 0 \\ 8 & 11 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 9 & 10 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right]$$

where the genes in bold of $\mathbf{o^1}$ and $\mathbf{o^2}$ are the ones inherited from $\mathbf{S}$ and $\mathbf{T}$, respectively. The 3TX then builds the *mapping matrix* $\mathbf{M} = \left[\begin{smallmatrix} 3 & 5 \\ 0 & 0 \end{smallmatrix}\right]$, whose row m_{1j} (m_{2j}) contains the genes of $\mathbf{o^1}$ ($\mathbf{o^2}$) that come from $\mathbf{S}$ ($\mathbf{T}$). Each duplicated gene of $\mathbf{o^1}$ that is outside its crossing region is replaced with the value in the other row of $\mathbf{M}$. Offspring $\mathbf{o^1}$ has two duplicated genes, i.e., $o^1_{112} = 3$ and $o^1_{212} = 5$, which are both replaced with 0, according to $\mathbf{M}$. In order to reintroduce the missing genes in $\mathbf{o^2}$, the 3TX randomly selects two genes equal to 0 outside the crossing region, and assigns 3 and 5 to them, respectively.

b) Replacement: The replacement mode selects one row $t_{i^t:k^t}$ of $\mathbf{T}$ and one row $s_{i^s:k^s}$ of $\mathbf{S}$. Two offspring $\mathbf{o^1}$ and $\mathbf{o^2}$ are generated. Offspring $\mathbf{o^1}$ inherits the genetic information from $\mathbf{T}$, except row $o^1_{i^t:k^t}$, which is inherited by $\mathbf{S}$, so $o^1_{i^t:k^t} = s_{i^s:k^s}$. Offspring $\mathbf{o^2}$ is generated in the same way, exchanging the roles of $\mathbf{T}$ and $\mathbf{S}$.

For example, consider rows $t_{1:3}$ and $s_{1:2}$. The following offspring are generated

$$\mathbf{o^1} = \left[\begin{bmatrix} 1 & 2 & 0 \\ 6 & 0 & 0 \\ 11 & 0 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 & 0 \\ 5 & 7 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{2} & \mathbf{3} & \mathbf{5} \\ 8 & 9 & 10 \\ 0 & 0 & 0 \end{bmatrix}\right], \mathbf{o^2} = \left[\begin{bmatrix} 1 & 0 & 0 \\ 4 & 6 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{4} & \mathbf{0} & \mathbf{0} \\ 7 & 0 & 0 \\ 8 & 11 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 9 & 10 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right]$$

where the genes in bold in $\mathbf{o^1}$ and $\mathbf{o^2}$ are the ones inherited from $\mathbf{S}$ and $\mathbf{T}$, respectively. Duplicated and missing genes are repaired using the mapping matrix $\mathbf{M} = \left[\begin{smallmatrix} 4 & 0 & 0 \\ 2 & 3 & 5 \end{smallmatrix}\right]$, as explained for the cut-paste mode.

c) Slicing: In this mode, the 3TX selects one horizontal, lateral or frontal slice of $\mathbf{T}$, and one slice (of the same type) of $\mathbf{S}$. The first offspring $\mathbf{o^1}$ inherits the genetic information from $\mathbf{T}$, except the selected slice, which is inherited from $\mathbf{S}$. Offspring $\mathbf{o^2}$ is generated by exchanging the role of $\mathbf{T}$ with the role of $\mathbf{S}$.

As an example, consider lateral slices $t_{:2:}$ and $s_{:1:}$. The two offspring generated are

$$\mathbf{o^1} = \left[\begin{bmatrix} 1 & \mathbf{1} & 0 \\ 6 & \mathbf{4} & 0 \\ 11 & \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} 3 & \mathbf{2} & 0 \\ 5 & \mathbf{7} & 0 \\ 0 & \mathbf{8} & 0 \end{bmatrix} \begin{bmatrix} 4 & \mathbf{0} & 0 \\ 8 & \mathbf{9} & 10 \\ 0 & \mathbf{0} & 0 \end{bmatrix}\right], \mathbf{o^2} = \left[\begin{bmatrix} \mathbf{2} & 0 & 0 \\ \mathbf{0} & 6 & 0 \\ \mathbf{0} & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0} & 3 & 5 \\ \mathbf{7} & 0 & 0 \\ \mathbf{0} & 11 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0} & 0 & 0 \\ \mathbf{9} & 10 & 0 \\ \mathbf{0} & 0 & 0 \end{bmatrix}\right]$$

and the mapping matrix to repair the duplicates and missing genes is $\mathbf{M} = \left[\begin{smallmatrix} 1 & 4 & 0 & 2 & 7 & 8 & 0 & 9 & 0 \\ 2 & 0 & 0 & 0 & 7 & 0 & 0 & 9 & 0 \end{smallmatrix}\right]$.

2) Mutation: Given a mutation probability p_m, each gene t_{ijk} of a chromosome $\mathbf{T}$ is assigned a random number σ_{ijk} with uniform probability distribution in $[0, 1]$. The genes with $\sigma_{ijk} \leq p_m$ are exchanged with one gene that is randomly selected from $\mathbf{T}$, which may be the gene that is mutating.

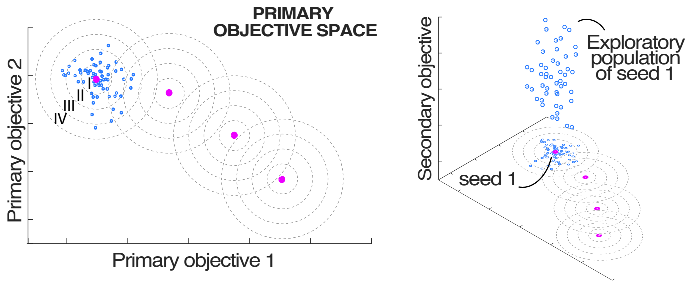

Figure 3. Seeds and projection of the exploratory population of one of them on the primary objective space (left-hand side). The roman numerals indicate the zone rank. The right-hand side shows the exploratory population of the same seed in the full objective space (considering a problem with three objectives).

D. Description of TeMA

Consider an MaO problem with five objectives measured by a vector of tensorial fitness functions $\mathbf{g}(\mathbf{T}) = [g_1(\mathbf{T}), g_2(\mathbf{T}), g_3(\mathbf{T}), g_4(\mathbf{T}), g_5(\mathbf{T})]$ that evaluate (3), (4), and (6)-(8) by representing a disassembly schedule $\mathbf{s}$ as a third-order tensor $\mathbf{T}$.

TeMA first splits the objectives into *primary* and *secondary*, according to a vector of weights $\mathbf{w} = (w_1, w_2, w_3, w_4, w_5)$, with $w_i \in (0, 1)$, derived from a decision-maker's preferences. The two stages of TeMA are explained in the next sections.

1) First stage: The first stage only takes the primary objectives into account. In this stage, the aim of TeMA is to look for a set of solutions that are Pareto-optimal with respect to these objectives (*primary front*).

TeMA creates an initial population of n tensorial individuals by generating permutations of the tasks that are then encoded as third-order tensors, as described in Section IV-A. This population evolves via the genetic operators explained in Sections IV-C1 and IV-C2, and achieves the primary front. Each solution $\mathbf{T}$ of the primary front is called *seed*.

2) Second stage:

a) Generating the exploratory populations: The second stage begins generating a set $\mathcal{E}_\mathbf{T}$ of E solutions to associate with each seed $\mathbf{T}$. These solutions $\mathbf{E}^\mathbf{T}_i \in \mathcal{E}_\mathbf{T}$ are called *explorers* and form the *exploratory population* of the seed. The exploratory population $\mathcal{E}_\mathbf{T}$ of seed $\mathbf{T}$ is generated by performing a local search that explores the neighborhood of $\mathbf{T}$. The idea is to try to improve the secondary objectives by slightly degrading the primary objectives. The local search considers each seed $\mathbf{T}$ and performs permutations of its genes, so that the projection $\mathbf{g}^\perp(\mathbf{E}^\mathbf{T}_i)$ of the fitness vector $\mathbf{g}(\mathbf{E}^\mathbf{T}_i)$ of each generated explorer $\mathbf{E}^\mathbf{T}_i$ on the primary objective space falls close to the seed. The maximum distance from the seed derives from the preferences in $\mathbf{w}$.

b) Ranking explorers: Once all the exploratory populations have been created, each explorer is assigned a *first-order rank* whose performance increases in inverse proportion to the number of explorers that dominate it. The top-ranked explorers are thus non-dominated. Those explorers (*infeasible*) that violate some constraints are assigned the worst rank, and are sorted in ascending order of average constraint violation.

The top-ranked explorers are then further ranked by assigning each of them a *zone rank*. The zone rank is designed to increase the selection pressure, and makes it possible to prune the exploratory population, by only retaining those explorers that are closest to the seeds, in the primary objective space.

Without loss of generality, consider a set of primary objectives made up of two objectives, and a seed $\mathbf{T}$. In order to determine the zone rank of each non-dominated explorer, TeMA places C concentric circles centered in the seed, in the primary objective space. This generates C rings around the seed. These are the zones where the zone rank works. In more detail, the zone rank of an explorer $\mathbf{E}_i^{\mathbf{T}}$ is better the closer to the seed the ring is where the projection $\mathbf{g}^{\perp}(\mathbf{E}_i^{\mathbf{T}})$ of the objective vector $\mathbf{g}(\mathbf{E}_i^{\mathbf{T}})$ of the explorer falls. The closer a non-dominated explorer is to the seed, the greater the chances of survival and reproduction. If the number of primary objectives is higher than two, the concentric circles become concentric (hyper)spheres, and the rings become (hyper)spherical shells. But nothing changes in the procedure. The stages of TeMA are shown in Fig. 3. Fig. 4 shows the overall flowchart of TeMA.

c) Local search performed by the explorers: The top-ranked explorers of each exploratory population have the highest probability of being selected for reproduction, thus enabling the search to concentrate in the immediate vicinity of the seeds, in the primary objective space.

When the time comes for selecting explorers to evolve, TeMA uses a binary tournament. Given two explorers $\mathbf{E}_i^{\mathbf{T}}$ and $\mathbf{E}_j^{\mathbf{T}}$ that take part in the tournament, there may be three cases. If both explorers are feasible, the better one is selected on the basis of their rank. If only one explorer is feasible, that one is selected. If both explorers are infeasible, then the one with the lower average constraint violation is selected. The selected explorers evolve within parallel genetic loops, undergoing the tensorial operators in Section IV-C. After recombination, the explorers to retain are chosen on the basis of the rank. The number of explorers in each exploratory population is limited to E by discarding the worst explorers, i.e. the ones furthest from the seeds (bad zone rank). Each seed $\mathbf{T}$ has a *spam archive* where it throws away the discarded explorers.

The parallel genetic loops synchronize whenever a certain number of generations have passed. Synchronization stops all loops before the phase where they select the explorers to retain. When this time comes, the two closest seeds to each seed $\mathbf{T}$ (*neighboring seeds*) are given a chance to pick some explorers from the spam archive of $\mathbf{T}$. The closer an explorer in the spam archive is to one of the neighboring seeds, the more likely this seed selects the explorer, and brings it to its exploratory population. The exploratory populations thus cooperate thereby enabling the algorithm to increasingly refine the search in the immediate vicinities of the seeds.

After collaborating, if a stop condition is met, e.g. a maximum number of generations is achieved or exceeded, the various exploratory populations are merged and the algorithm terminates. Otherwise, the genetic loops resume working in parallel, until the next synchronization.

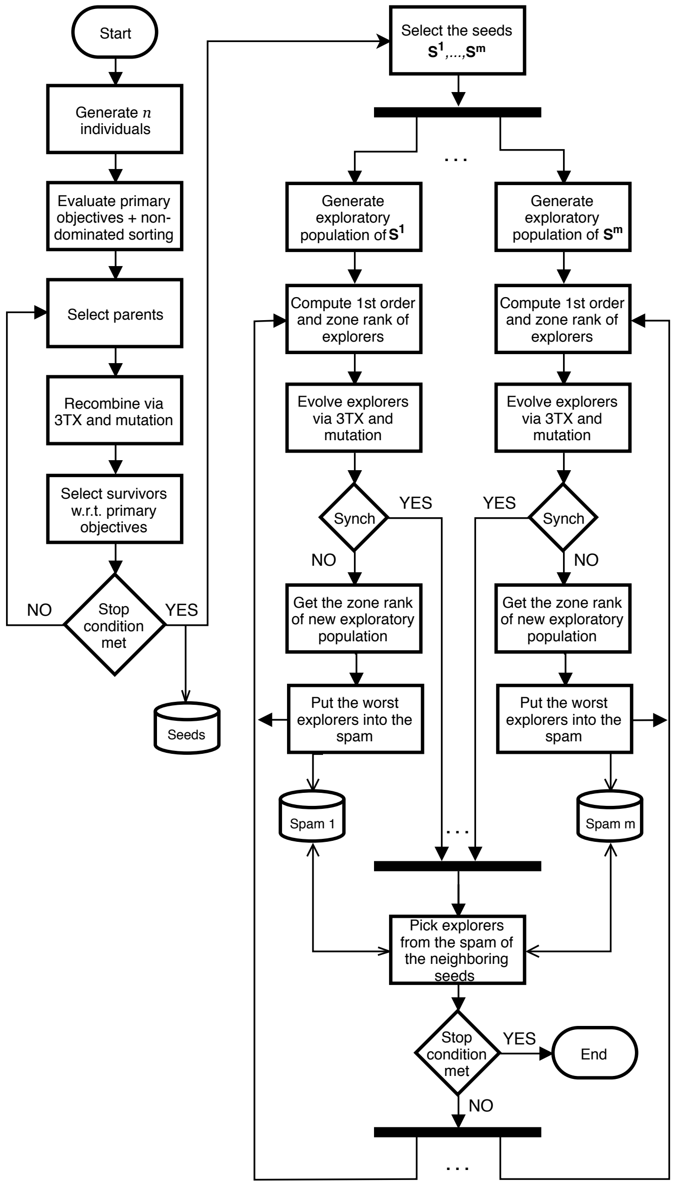
Start
Generate n individuals
Evaluate primary objectives + non-dominated sorting
Select parents
Recombine via 3TX and mutation
Select survivors w.r.t. primary objectives
Stop condition met
NO
YES
Seeds
Select the seeds $\mathbf{s}^1,...,\mathbf{s}^m$
Generate exploratory population of $\mathbf{S}^1$
Generate exploratory population of $\mathbf{S}^m$
Compute 1st order and zone rank of explorers
Evolve explorers via 3TX and mutation
Synch
YES
NO
Get the zone rank of new exploratory population
Put the worst explorers into the spam
Spam 1
Spam m
Pick explorers from the spam of the neighboring seeds
Stop condition met
YES
NO
End

Figure 4. Flowchart of TeMA. The left-hand side and the right-hand side contain the first and second stages, respectively.

V. Experiments

TeMA was developed in MATLAB, and was tested on the refurbishment of two products: a smartphone and a washing machine. The experiments were carried out using a virtual machine running Linux Debian OS, with four quad-core CPUs and 64 GB of RAM. This section presents the results.

A. Parameters

To find the best values for the crossover rate P^X and mutation probability P^M, an initial sampling phase was carried out to get a uniformly-distributed subset of all the infinite combinations of the parameter values. Crossover rates from 0.5 up to 1 (with step 0.1), and mutation probabilities equal to $P^M = 0.01P^X$ were considered. Lower crossover rates were

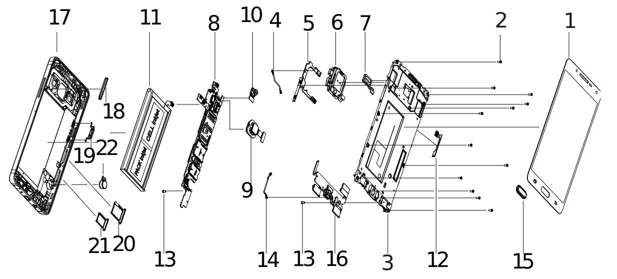

Figure 5. Exploded view drawing of the Samsung Galaxy A5.

Table I
REMOVED PART, TIME, FACE AND PRECEDING TASKS OF EACH TASK ID

ID	Part	Time	Face	Prec	ID	Part	Time	Face	Prec
1	Display	10”	T	-	12	Antenna cable	5”	T	11
2	Screws	2’30”	T	1	13	Screw	10”	B	11
3	Bracket	5”	T	2	14	Wire	30”	B	13
4	Coax cable	30”	B	5	15	Button	2”	T	2
5	Volume key	10”	B	3	16	PBA-FPCB module	10”	B	14,15
6	Speaker	5”	B	5	17	Rear speaker	5”	B	9,10,16
7	Aux receiver	15”	B	3	18	Key volume	5”	T	17
8	Logic board	5”	B	4,6,7	19	Key power	5”	T	17
9	Camera	10”	T	8	20	Sim slot	3”	T	17
10	Camera	8”	T	8	21	Sim slot	3”	T	17
11	Battery	15”	B	8	22	Power key module	5”	T	20,21

ignored as they are well-known to be insufficient, whereas mutation probabilities generally have to be one/two orders of magnitude lower than the crossover rate to make it less likely that good solutions are perturbed [18].

A total of 36 parameter vectors were obtained. The racing technique [26] was then used as a tuner, with 100 as the maximum number of tests per parameter vector. This meant that $P^X \in \{0.6, 0.7, 0.8\}$ were retained. A total of 30 executions of TeMA were then run for each parameter vector retained. The performance of each run was measured by calculating the hypervolume (HV) of the Pareto front. The (P^X, P^M) pair with the highest mean HV was chosen. Student’s t-test with a 95% confidence interval was used for validation.

B. Case study I: smartphone

The smartphone considered is the Samsung Galaxy A5. This device was chosen because it is in high demand. People change their smartphones frequently, just because they go out of fashion. Refurbished smartphones have started to be sold by the most famous brands.

1) Dataset: The device is made up of 22 parts, its exploded view drawing is shown in Fig. 5. Table I contains the entire dataset. This product has two faces, *top (T)* and *bottom (B)*.

2) Results: The refurbishment policy considered in the experiments entails replacing the battery and testing the screen, logic board and antenna.

Disassembly time and the degree of parallelism were considered as the primary objectives. A smartphone is a thin and light device, which is quick and easy to rotate. There are no ergonomics issues when disassembling a device like this. The weights that were assigned to the objectives were $\mathbf{w} = (0.05, 0.4, 0.05, 0.4, 0.1)$.

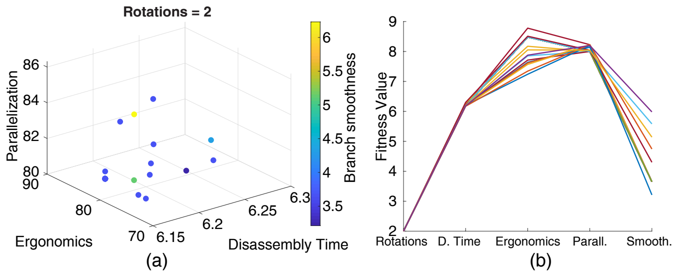

Figure 6. Scatter plot (a) and parallel coordinates plot (b) of the solutions obtained in the experiments involving the smartphone. In (b), the values of the third and fourth objectives have been divided by 10 to make the plot clearer. “D. time”, “Parall.” and “Smooth.” are abbreviations for “Disassembly time”, “Degree of parallelism” and “Branch smoothness”.

The first stage was run for 500 generations, considering 2 workers and B as the ground face. The crossover rate and mutation probability were set to 0.6 and 0.06, respectively. TeMA found 11 seeds. The second stage was run for 1000 generations with 20 explorers per seed. TeMA took 5’38” to converge.

Fig. 6 shows a scatter plot and a parallel coordinates plot of the solutions found. The number of rotations is constant (it is the only one found) and the smoothness is represented with color. The parallel coordinates plot highlights that TeMA accurately explored the regions close to the seeds. The values of the primary objectives are very close to each other, whereas those of the secondary objectives are spread around the region, over a wider range of values.

In accordance with the relative importance of each objective, the decision maker chose solution $\mathbf{S} = [(\langle 1,2\rangle), (\langle 3,5,7\rangle, \langle 15\rangle), (\langle 4\rangle, \langle 6\rangle), (\langle 8\rangle), (\langle 11,13\rangle, \langle 9\rangle), (\langle 14,16\rangle, \langle 10\rangle), (\langle 17\rangle, \langle 12\rangle), (\langle 19,20\rangle, \langle 21\rangle), (\langle 22,18\rangle)]$, whose fitness is $\mathbf{g}(\mathbf{S}) = (-2, -6.63, 72.92, 88.89, -3.99)$. This solution was chosen because it takes the best values of the secondary objectives. It is thus the Pareto-optimal solution that achieves the best compromise between primary and secondary objectives. The degree of parallelism is high, as required, since almost all branches of each phase have another branch in parallel. The total duration of all tasks performed in sequence would normally be 7’76”. The chosen solution completes the disassembly in 6’63”, which corresponds to a time saving of 14.6%. The sequence entails only two rotations of the smartphone.

C. Case study II: washing machine

This case study considers the refurbishment of the Whirlpool WFW87HEDW0 washing machine, which was chosen to test TeMA on a highly complex product that involves almost five times more components than the smartphone.

1) Dataset: The washer is made up of 103 parts. Its exploded view drawing is in Fig. 7. In order to prevent damage, the manufacturer recommends that the washer should lie on only four of its six faces: *top (T)*, *bottom (B)*, *front (F)* and *rear (R)*. The dataset is in Table II. Each row is associated with a part, which belongs to one of the subassemblies (A, B, C, D or

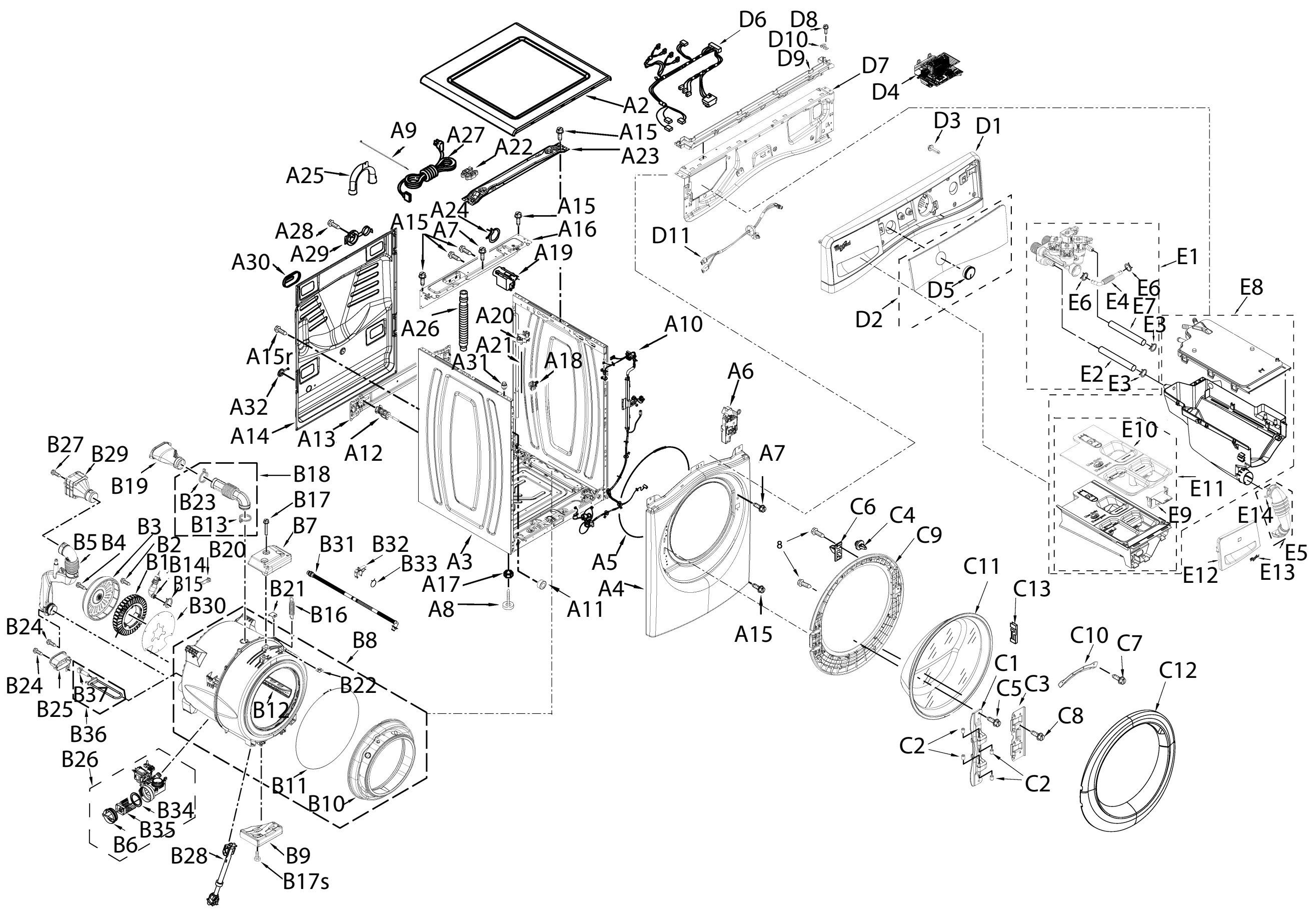

Figure 7. Exploded view drawing of the Whirlpool WFW87HEDW0 washing machine. The letter that precedes the task identifier indicates the subassembly: 'A' is the cabinet, 'B' is the basket, 'C' is the door, 'D' is the control panel and 'E' is the dispenser.

E) shown in Fig. 7. The preceding tasks are listed starting from those involving the same subassembly as the part associated with the row, omitting the letter. The remaining tasks are grouped according to the subassembly that they belong to. The letter is only reported before the first task in each subassembly.

2) Results: The refurbishment process tests the water pump, resistance, motor, cables, grounding, bearings and basket. These are key parts of a washing machine, and were chosen as they are typically included in refurbishment policies.

The primary objectives chosen were the degree of parallelism and the number of rotations. A washing machine is heavy, rotating it takes time and physical effort: limiting the rotations is thus crucial. On the other hand, the volume of the product lends itself well to performing many tasks in parallel. The weights assigned to the objectives were thus $\mathbf{w} = (0.35, 0.1, 0.1, 0.35, 0.1)$.

TeMA was run for 1000 generations, considering 2 workers. The crossover rate and mutation probability were 0.7 and 0.07, respectively. A total of 10 seeds were found. In the second stage, 40 explorers were associated with each seed. The total running time was 21'43". The algorithm might seem slow, but problem (2a)-(2f) is extremely complex here, with over two million variables and it has to meet many more constraints due to the larger set of precedences of Table II.

Fig. 8 shows two scatter plots and two radar plots of the solutions found. Scatter plots 8a and 8b show that the solutions are spread over the objective space and are characterized by good values of the primary objectives. The DSs only require two or three rotations, which minimizes the workers' effort and shortens the disassembly time. The sequential disassembly of

$$\begin{aligned}\mathbf{S} = [&(\langle A28, A2\rangle, \langle A15r, A14\rangle), (\langle D6, B17s, B7, B16, B21, B33\rangle\langle A7\rangle), (\langle B11\rangle, \langle B32, B31, B13, B23, A22, A23, D1\rangle), (\langle D8\rangle,\\ &\langle\rangle), (\langle\rangle, \langle D10, E3, E2\rangle), (\langle E7\rangle, \langle E14\rangle), (\langle E1, E8\rangle, \langle\rangle), (\langle\rangle, \langle E6, B19\rangle), (\langle A15\rangle, \langle\rangle), (\langle D9\rangle, \langle A21\rangle), (\langle A24\rangle, \langle A20\rangle), (\langle\rangle, \langle E5\rangle),\\ &(\langle A4\rangle, \langle E11\rangle), (\langle A5\rangle, \langle\rangle), (\langle A6\rangle, \langle\rangle), (\langle A16\rangle, \langle E4\rangle), (\langle A33\rangle, \langle D11\rangle), (\langle A34\rangle, \langle D3\rangle), (\langle A35\rangle, \langle B18\rangle), (\langle B15\rangle, \langle\rangle), (\langle B14\rangle,\\ &\langle\rangle), (\langle\rangle, A26), (\langle B26\rangle, \langle D4\rangle), (\langle B6\rangle, \langle B22\rangle), (\langle B35, B28, B17\rangle, \langle\rangle), (\langle B4\rangle, \langle C7\rangle), (\langle\rangle, \langle B24, B5\rangle), (\langle A11, B27,\\ &B29\rangle, \langle\rangle), (\langle B25\rangle, \langle B3\rangle), (\langle B37, B36\rangle, \langle\rangle), (\langle B2\rangle, \langle D5, B10\rangle), (\langle B1, B30, B9\rangle, \langle\rangle), (\langle\rangle, \langle B8, PL, E12, E10, E9\rangle), (\langle A19, A18\rangle,\\ &\langle\rangle), (\langle C10\rangle, \langle E13\rangle), (\langle\rangle, \langle C8, C4\rangle), (\langle C6\rangle, \langle B34\rangle), (\langle C11, C13, C5\rangle, \langle\rangle), (\langle C2\rangle, \langle A10\rangle), (\langle\rangle, \langle C1\rangle), (\langle C8\rangle, \langle D2\rangle), (\langle C3\rangle, \langle\rangle)]\end{aligned} \tag{10}$$

Table II
REMOVED PART, TIME, FACE AND PRECEDING TASKS OF EACH TASK ID

ID	Part	Time	Face	Prec.	ID	Part	Time	Face	Prec.
A2	Top panel	10”	T	-	22	Push nut	10”	T	31
3	Cabinet	-	-	-	23	Clamp	5”	R	13
4	Front panel	20”	F	7,15 B11	24	Screw	25”	B	A35
5	Clamp	10”	F	4	25	Cover	5”	B	24
6	Lock	20”	T	4	26	Drain pump	1’15”	B	A26
7	Screw	1’10”	R	-	27	Screw	55”	R	5
8	Leveling leg	1’30”	B	-	28	Shock absorber	8”	B	A35
9	Cable tie	20”	T	1	29	Housing	15”	R	27
10	Main harness	15”	F	18,20 21	30	Shield	5”	R	1
11	Spacer washer	35”	F	4	31	Pump drain hose	15”	T	32
12	Shipping bolt kit				32	Dryer clip	8”	T	33
13	Bracket				33	Hose clamp	5”	T	16,21
14	Panel	20”	R	15r	34	O-ring cap	5”	B	35
15	Screw	32”	F	2	35	Fluid element	15”	B	6
16	Bracket	15”	T	7	36	Heater assembly	20”	B	37
17	Leg lock nut				37	Temp. sensor	20”	B	25
18	Clip	9”	R	23	C1	Door hinge	25”	F	2
19	Noise filter	10”	R	16	2	Hinge pin cap	20”	F	5
20	Water switch	8”	T	23	3	Hinge support	10”	F	8
21	Water switch hose	8”	R	23	4	Bumper	15”	F	A4
22	Retainer	12”	R	2	5	Screw	20”	F	4
24	Hose clamp	5”	R	-	7	Screw	20”	F	A4
25	U-bend bracket				8	Screw	15”	F	A4
26	Drain hose	12”	R	B14	9	Door inner panel	10”	F	A4
27	Power cord				10	Glass retainer	10”	F	7
28	Screw	1’20”	R	-	11	Glass	10”	F	8,10
29	Cover	5”	R	-	12	Outer door	15”	F	-
30	Power cord cover	5”	R	-	13	Clamp	5”	F	-
31	Top panel screw	20”	T	2	D1	Control panel	10”	F	A2
32	Cover	5”	R	-	2	Interface	15”	F	5
B1	Motor stator	45”	R	2	3	Screw	20”	R	A2
2	Screw	1’18”	R	3	4	Logic board	1’10”	T	6
3	Rotor	15”	R	4	5	Control knob	2”	F	-
4	Screw	1’20”	R	A14	6	Main harness	1’30”	T	A2
5	Vent pipe	13”	R	24	7	Bracket	10”	F	9
6	Filter cap	20”	B	26	8	Screw	1’05”	T	1
7	Counterweight	45”	T	17s	9	Water channel	5”	T	10
8	Tub and basket	25”	T	9,18,21 22,29 30,36	10	Dryer clip	15”	T	8
9	Counterweight	5”	B	17	11	Wire harness	10”	F	1
10	Boot	40”	F	A5	E1	Water inlet valve	15”	F	2,7,A2
11	Boot spring clamp	1’03”	F	-	2	Hose	5”	T	3
12	Baffle	30”	F	-	3	Drain hose clamp	15”	F	D10
13	Hose clamp	5”	T	A14	4	Hose	3”	F	6
14	Trap	8”	R	15	5	Hose	5”	F	14
15	Trap hose clamp	5”	R	A35	6	Drain hose clamp	10”	F	1
16	Suspension spring	15”	T	7	7	Hose	5”	T	3
17	Screw	40”	B	28	8	Drawer assembly	20”	T	14
18	Vent bellows	15”	T	19	9	Drawer divider	5”	F	10
19	Vent pipe	10”	R	23	10	Housing cover	3”	F	12
20	Clip	10”	T	7	11	Drawer assembly	2”	F	8
21	Clamp	10	T	31					

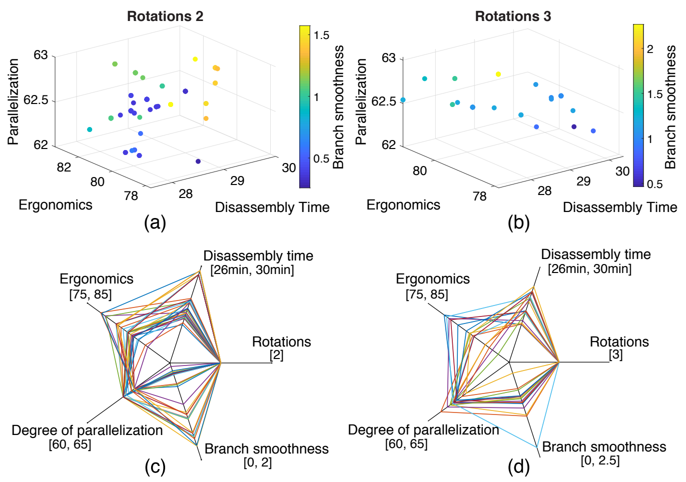

Figure 8. Scatter plots (a)-(b), and radar plots (c)-(d) of the solutions obtained by the experiments involving the washing machine.

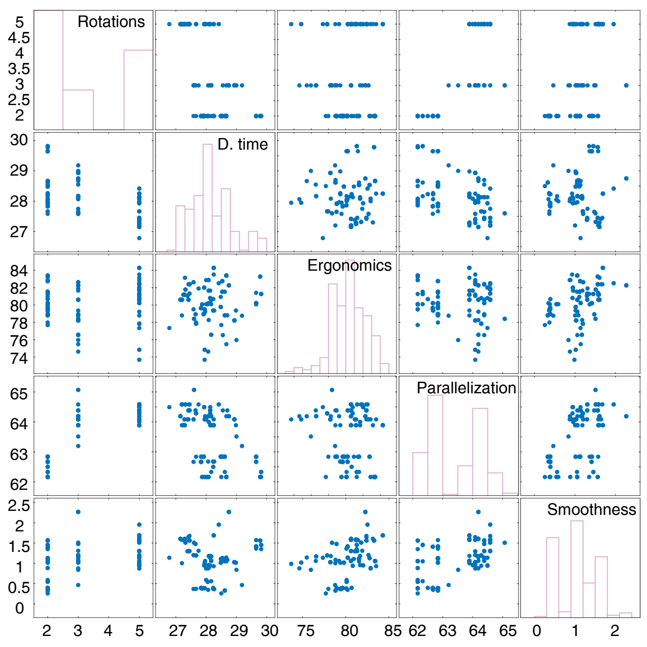

Figure 9. Scatter plot matrix of the solutions obtained by the experiments on the washing machine.

the washer would normally require 46.18 minutes. Fig. 8c and 8d highlight that the disassembly time can be almost halved. In fact, with TeMA it averages 28.54 minutes. The level of ergonomics is also very high. Fig. 8c and 8d show that it ranges from 75 to 85. This means that most of the tasks performed in parallel are on parallel or adjacent faces, thus minimizing the presence of tasks simultaneously performed on the same face of the product, with a lower chance of workers getting in each other’s way. Finally, both the scatter and radar plots in Fig. 8 show that the degree of parallelism of the solutions found averages 62.58, which means that just under two thirds of the branches are performed in parallel with another branch. This is a very promising result because disassembling a washing machine entails dealing with a huge number of precedence constraints [see Table 2].

In accordance with the preferences, the final solution chosen was the one in (10), whose fitness is $\mathbf{g}(\mathbf{S}) = [-2, -27.45, 82.39, 64.38, -1.56]$. This solution achieves an excellent compromise between the primary objectives, as it disassembles all the parts to test with only two rotations of the machine, with a degree of parallelism—higher than the average—that guarantees that just under 65% of the tasks are performed in parallel. Higher values could not be obtained due to the precedence constraints. Thanks to TeMA and the proposed parallel DSP formulation, the washing machine is disassembled in just 27.45 minutes, i.e. just over half of the 46.18 minutes required by sequential disassembly procedures, which represent the vast majority of the current literature. This time saving could be used to disassemble another machine, thus almost doubling productivity.

Finally, a pairwise comparison of the objectives was made to assess the ability of TeMA to accurately search the objective space. Fig. 9 shows a scatter plot matrix that summarizes the results and highlights that TeMA efficiently searches the objective space in the decision maker's area of preference. Each row i (column j) is made up of five cells, and relates to the objective reported in cell (i,i) (cell (j,j)). Each cell (i,j) with $i \neq j$, contains a scatter plot of the solutions obtained, whose x-axis and y-axis relate to objectives i and j, respectively. The cells in the diagonal show the histogram distribution of the solutions for each objective. The numerical values on the left of each row i (below each column j) refers to cells (i,j) with $i \neq j$ (i.e, all cells in column j).

VI. Performance evaluation

The performance of TeMA was compared to the performances of MOEA/D [27], NSGA-III [28], SPEA2+SDE [29], PICEA-g [30], and HypE [31]. These algorithms were chosen as they are among the best ones for MaO optimization [32].

A. Performance evaluation procedure

A total of 30 executions of TeMA were first run. At the end of each execution k, the solutions of the Pareto front obtained were normalized in [0,1] in order to obtain the normalized front $\mathcal{P}^k$. The ideal objective vector $\mathbf{z_i^{ideal,k}}$ was calculated, whose elements are $z_i^{ideal,k} = \sup_{\mathbf{x}\in\mathcal{P}^k} f_i(\mathbf{x}), \forall i = 1,\ldots,5$. The centroid of the ideal objective vectors is $\mathbf{r} = \sum_k \mathbf{z^{ideal,k}}/30$.

As the true Pareto front is unknown, the performance of TeMA was evaluated using the hypervolume (HV) and spacing (SP) measures. For each run, the HV was estimated using a Monte Carlo method using reference point $\mathbf{r}$. The average of all HVs (SPs) was calculated: the higher (lower) the average, the better the performance. The same procedure was carried out for the algorithms compared. These algorithms—which are problem-independent MaO algorithms—were set up as recommended in the original papers and in [33], considering the values suggested for problem instances that are similar to those tested in this work. In particular: MOEA-D considers a population size $n = 210$ and $T = \lceil \frac{n}{10} \rceil$; SPEA2+SDE considers a population size $n = 200$; PICEA-g considers $N_{goal} = 100$; HypE considers 10,000 poits for HV estimation. NSGA-III uses the genetic parameters of TeMA, and this also holds for the previous algorithms that are based on genetic evolution.

The distribution of the HVs and SPs of the executions of TeMA and the other algorithms are shown in Fig. 10. As can be seen, TeMA is more efficient than the compared algorithms, as it averages a wider (lower) HV (SP).

The results were validated by using Student's t-test with a 95% confidence, considering that the difference in mean is due to chance, as the null hypothesis. With the exception of the comparisons relating to SP that were made between TeMA and PICEA-g, and between TeMA and HypE, for all the other comparisons it was possible to reject the null hypothesis.

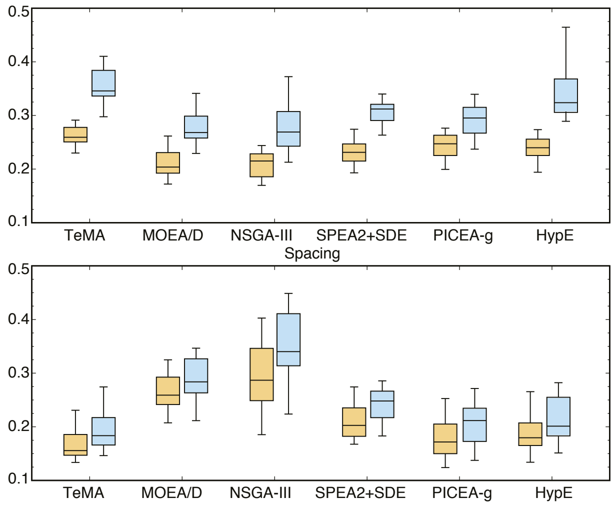

Figure 10. Box plot of HV (top) and SP (bottom) values obtained by TeMA and the other algorithms. The two boxes of each algorithm relate to the smartphone and washing machine refurbishments, from left to right.

The comparisons relating to SP confirmed that TeMA can find solutions that are spread around a wider or equally wide region of the objective space. The comparisons relating to HV showed that the solutions found by TeMA are better in any case: TeMA is thus more accurate than the other algorithms in the decision maker's area of preference.

VII. Conclusions

This paper has presented a novel MaO parallel DSP formulation and TeMA, a tensorial MaO memetic algorithm for highly efficient refurbishment-oriented DSP. TeMA guarantees quick and convenient parallelized DSs, outperforming other commonly-used MaO algorithms.

Companies can thus design more efficient industrial processes to refurbish complex consumer electronics up to two times faster, so as to enter the very profitable refurbishment market.

Acknowledgment

The authors wish to thank Andrea Stevanato for developing part of the MATLAB code.

References

[1] *https://www.usa.gov/statistics*.
[2] N. Resmi and K. Fasila, "E-waste management and refurbishment prediction (EMARP) model for refurbishment industries," *J. of Environmental Manag.*, vol. 201, pp. 303 – 308, 2017.
[3] R. M. Difrancesco *et al.*, "Optimizing the return window for online fashion retailers with closed-loop refurbishment," *Omega*, vol. 78, pp. 205 – 221, 2018.
[4] S. McGovern and S. Gupta, *The disassembly line: balancing and modeling*. New York, USA: McGraw-Hill, 2011.
[5] S. M. McGovern and S. M. Gupta, "Ant colony optimization for disassembly sequencing with multiple objectives," *The Int. J. of Adv. Manuf. Technol.*, vol. 30, no. 5, pp. 481–496, Sep 2006.
[6] M. Alshibli *et al.*, "Disassembly sequencing using tabu search," *J. of Intell. & Robot. Syst.*, vol. 82, no. 1, pp. 69–79, Apr 2016.
[7] J. L. Rickli and J. A. Camelio, "Multi-objective partial disassembly optimization based on sequence feasibility," *J. of Manuf. Syst.*, vol. 32, no. 1, pp. 281 – 293, 2013.

[8] F. Pistolesi *et al.*, "EMOGA: A hybrid genetic algorithm with extremal optimization core for multiobjective disassembly line balancing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 1089–1098, 2018.

[9] X. Guo *et al.*, "Disassembly sequence optimization for large-scale products with multiresource constraints using scatter search and petri nets," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2435–2446, 2016.

[10] X. Guo *et al.*, "Dual-objective program and scatter search for the optimization of disassembly sequences subject to multiresource constraints," *IEEE Trans. Autom. Sci. and Eng.*, vol. 15, no. 3, pp. 1091–1103, 2018.

[11] G. Tian *et al.*, "Disassembly sequence planning considering fuzzy component quality and varying operational cost," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 748–760, 2018.

[12] Y. Ren *et al.*, "Selective cooperative disassembly planning based on multi-objective discrete artificial bee colony algorithm," *Eng. Appl. of AI*, vol. 64, pp. 415 – 431, 2017.

[13] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Struct. Multidisc. Optim.*, vol. 26, no. 6, pp. 369–395, 2004.

[14] Y. Ren *et al.*, "An asynchronous parallel disassembly planning based on genetic algorithm," *European J. of Operational Research*, vol. 269, no. 2, pp. 647 – 660, 2018.

[15] Y. Tian *et al.*, "Effectiveness and efficiency of non-dominated sorting for evolutionary multi- and many-objective optimization," *Complex Intell. Syst.*, vol. 3, pp. 247–263, 2017.

[16] S. Bechikh *et al.*, *Many-objective Optimization Using Evolutionary Algorithms: A Survey*. Cham: Springer, 2017, pp. 105–137.

[17] K. Li *et al.*, "Evolutionary many-objective optimization: A comparative study of the state-of-the-art," *IEEE Access*, vol. 6, pp. 26 194–26 214, 2018.

[18] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[19] X. Chen *et al.*, "A multi-facet survey on memetic computation," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 591–607, 2011.

[20] P. Moscato, "On evolution, search, optimization, gas and martial arts: toward memetic algorithms," California Inst. Technol., Pasadena, CA, Tech. Rep. Caltech Concurrent Comput. Prog. Rep. 826, 1989.

[21] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," *IEEE Trans. Evol. Comput.*, vol. 9, no. 5, pp. 474–488, 2005.

[22] S. Wang and L. Wang, "An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 1, pp. 139–149, 2016.

[23] J. Wang *et al.*, "A hybrid multiobjective memetic algorithm for multiobjective periodic vehicle routing problem with time windows," *IEEE Trans. Syst., Man, Cybern., Syst.*, p. IN PRESS, 2018.

[24] X. Gong *et al.*, "Energy and labor aware production scheduling for industrial demand response using adaptive multi-objective memetic algorithm," *IEEE Trans. Ind. Informat.*, p. IN PRESS, 2018.

[25] C. Liao and C. Ting, "A novel integer-coded memetic algorithm for the set$k$-cover problem in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 2245–2258, 2018.

[26] O. Maron and A. W. Moore, "The racing algorithm: Model selection for lazy learners," *Artificial Intell. Rev.*, vol. 11, no. 1, pp. 193–225, 1997.

[27] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, 2007.

[28] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, 2014.

[29] M. Li *et al.*, "Shift-based density estimation for Pareto-based algorithms in many-objective optimization," *IEEE Trans. on Evol. Comput.*, vol. 18, no. 3, pp. 348–365, 2014.

[30] R. Wang *et al.*, "Preference-inspired co-evolutionary algorithm using adaptively generated goal vectors," in *IEEE Congr. Evol. Comput.*, 2013, pp. 916–923.

[31] J. Bader and E. Zitzler, "Hype: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, 2011.

[32] K. Li *et al.*, "Evolutionary many-objective optimization: A comparative study of the state-of-the-art," *IEEE Access*, vol. 6, pp. 26 194–26 214, 2018.

[33] H. Ishibuchi *et al.*, "Many-objective test problems to visually examine the behavior of multiobjective evolution in a decision space," in *Parallel Problem Solving from Nature*, 2010, pp. 91–100.

PLACE
PHOTO
HERE

Francesco Pistolesi is a Postdoctoral Researcher with the Department of Information Engineering, University of Pisa, Pisa, Italy. He received a Master of Science degree, summa cum laude, in computer engineering (enterprise curriculum) and a Ph.D. degree in information engineering (computer system architectures curriculum), both from the University of Pisa. His research interests are in artificial intelligence and data mining, with applications to decision support and multiobjective optimization. Francesco is currently working on innovative solutions for the smart industry that range from the optimization of industrial processes to the enhancement of the workers' safety and health.

PLACE
PHOTO
HERE

Beatrice Lazzerini (M'98) is a Full Professor of computer engineering with the Department of Information Engineering, University of Pisa, Pisa, Italy. She has coauthored seven books and has contributed to more than 210 papers in international journals and conferences. She is coeditor of two books. She was involved and had roles of responsibility in several national and international research projects and scientific events. Her research interests include computational intelligence, with a particular emphasis on fuzzy systems, neural networks, and evolutionary computation, and their applications to pattern classification, risk analysis and management, diagnosis, forecasting, and multicriteria decision making.