EMOGA: a Hybrid Genetic Algorithm with Extremal Optimization Core for Multiobjective Disassembly Line Balancing

Francesco Pistolesi, Beatrice Lazzerini, *Member, IEEE,* Michela Dalle Mura, and Gino Dini

***Abstract*—In a world where products get obsolescent ever more quickly, discarded devices produce million tons of electronic waste. Improving how end-of-life products are dismantled helps reduce this waste, as resources are conserved and fed back into the supply chain, thereby promoting reuse and recycling. This paper presents the Extremal MultiObjective Genetic Algorithm (EMOGA), a hybrid nature-inspired optimization technique for a multiobjective version of the Disassembly Line Balancing Problem (DLBP). The aim is to minimize the number of workstations, and to maximize profit and disassembly depth, when dismounting products in disassembly lines. EMOGA is a Pareto-based genetic algorithm (GA) hybridized with a module based on extremal optimization (EO), which uses a tailored mutation operator and a continuous relaxation-based seeding technique. The experiments involved the disassembly of a hammer drill and a microwave oven. Performance evaluation was carried out by comparing EMOGA to various efficient algorithms. The results showed that EMOGA is faster or gets closer to the Pareto front, or both, in all comparisons.**

***Index Terms*—Disassembly, intelligent manufacturing systems, extremal optimization, genetic algorithms, multiobjective optimization, recycling, waste reduction.**

I. Introduction

DISASSEMBLY is key for product recovery as it helps limit the ever-increasing waste coming from end-of-life goods. Today more than ever, many products — particularly electronic goods — have short life cycles as they quickly go out of fashion, and are replaced as soon as they start showing issues or signs of wear. Some defects could be repaired, but few people choose repair over upgrade.

The quantity of discards has thus rapidly increased. Smartphones, tablets, laptops, LEDs, LCDs, DVD and music players are discarded at the slightest inconvenience, such as breakage, slow-down, or just the availability of a newer model. If not reused or recycled, this huge amount of end-of-life products can pose a serious threat to the environment for years to come.

Electronic waste is also an important resource in terms of its potential for recovering valuable materials, such as aluminum, copper, and gold. However, electronic products are not designed to efficiently recover these materials.

Only 15% of global electronic waste is fully recycled [1], thus used electronics mainly end up as waste. It is also estimated that in 2017 the global electronic waste will have reached $\sim$65 million tons (+33% w.r.t. 2012) due to poor recycling and high turnover [2]. Recovering products has thus become a strict priority.

Product recovery typically starts with the *disassembly*, which extracts valuable parts/materials from products through a series of *tasks* performed in a *disassembly line* [3]. Disassembly lines are made up of sequential workstations. Determining which tasks to perform at each workstation so as to meet the precedences among tasks and optimize some measure of effectiveness is known as the Disassembly Line Balancing Problem (DLBP) [3]. The DLBP is an NP-hard combinatorial problem [4], whose objectives include profit, cycle time, number of workstations, and leveled line utilization [5].

Due to complexity, exact methods are not suitable for large-scale DLBPs. Various heuristic techniques have thus been proposed. Recent examples are hybrid techniques that blend stochastic simulation with neural networks and genetic algorithms [6], scatter search [7], genetic simulated annealing [8], probability analysis and stochastic simulation [9], self-adaptive swarm optimization [10], artificial bee colony [11], and advanced supply chain models [12]. These approaches consider the precedences among tasks as constraints. Some of them deal with multiple objectives, but get a single objective by using cost functions, penalties, or by making scalarizations. Scalarization methods have several drawbacks [13]:

1) the need to prioritize the objectives, e.g. with weights;
2) the solution may not reflect the priorities;
3) the solution is very sensitive to the weights;
4) multiple runs are needed in order to get multiple solutions, which then may be not uniform;
5) no solution may be found in non-convex regions of the Pareto front.

Pareto-based evolutionary techniques overcome these problems as they provide a uniform set of (near) Pareto-optimal solutions per run, with no weights to specify.

Genetic Algorithms (GAs) [14] are one of the most widely-used evolutionary techniques. GAs are efficient for large-scale combinatorial problems, can easily deal with multiple objectives, are robust to the discontinuities in the search space, and can be computed in parallel. However, GAs may be time consuming and exhibit premature convergence to local optima.

Manuscript received July 10, 2017; revised September 26, 2017; accepted October 27, 2017. This work was supported by the PRA 2016 project "Analysis of Sensory Data: from Traditional Sensors to Social Sensors", funded by the University of Pisa.

(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)

M. Dalle Mura and G. Dini are with the Department of Civil and Industrial Engineering, University of Pisa, Largo L. Lazzarino 1, 56122 Pisa, Italy, (email: m.dallemura@ing.unipi.it; dini@ing.unipi.it).

This latter drawback is less likely the higher the diversity of the solutions. Mutation is one way to keep diversity in GAs by replacing part of the population with randomly-generated individuals. Increasing the mutation rate may reduce local convergence [14], but mutation rates that are too high hinder convergence and make GAs perform like a random walk.

Extremal Optimization (EO) [15] is a relatively recent nature-inspired technique that evolves an individual solution by mutation alone. EO has been appropriately designed for combinatorial optimization. Unlike most other optimization techniques, EO removes the poor components of the solution, rather than preserving the good ones.

Hybrid techniques have been proposed to make the most of GAs and EO, thereby overcoming their drawbacks. For instance, a hybrid multiobjective approach jointly based on EO and combinatorial local search has been proposed in [16]. This algorithm uses scalarization, and is also not suitable for complex large-scale problems (such as the DLBP). In [17], EO has been modified to make it population-based. This algorithm is suitable for unconstrained continuous problems, so it is not appropriate to deal with the DLBP, as it is a large-scale discrete problem, with multiple constraints. Finally, in [18] the authors propose an interesting bi-objective hybrid evolutionary technique based on GAs and EO. Even though it combines the population-based capacity of GAs and the fine-grained local search of EO, the algorithm scalarizes the objectives by using a weighted sum, with the consequent drawbacks.

One way to overcome these drawbacks may be to blend EO and a GA in a Pareto-based evolutionary algorithm. This would boost the exploration without excessively increasing the mutation rate, and at the same time improves the diversity, makes the exploration wider, and reduces the chances of getting stuck in local optima. Based on this idea, this paper presents the Extremal MultiObjective Genetic Algorithm (EMOGA), a hybrid nature-inspired technique for the DLBP. EMOGA is a Pareto-based multiobjective GA with an EO core that uses a tailored mutation operator. The DLBP version considered here minimizes the number of workstations (NoW), while maximizing the profit and the disassembly depth. The first two objectives are among the most discussed in the literature [3]. The third one is a novelty w.r.t. existing Pareto-based approaches for disassembly, which use penalties that require additional parameters and can reduce the exploration [19]. The third objective thus makes it possible to relax the precedence constraints and look for disassembly sequences that remove as many parts as possible from the product.

This paper considers real-world industrial scenarios where:

- nondestructive disassembly is carried out;
- the recoverable components and materials are resold;
- lines have a serial layout, and one operator per station;
- products may undergo partial disassembly;
- lines are paced, i.e. operators can only work on a product for a limited time;
- workstations are flexible, with similar equipment.

The paper is structured as follows: Section II contains a background on multiobjective optimization (MOO), GAs and EO; Section III outlines the problem formulation; Section IV presents EMOGA; in Section V the results of the experiments are discussed; Section VI describes performance evaluation and statistical validation. Section VII draws the conclusions.

II. Preliminaries

A. Multiobjective optimization

An MOO problem can be written as $\max_{\mathbf{x}\in\mathcal{X}} \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})]$, with $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \leq 0, h_j(\mathbf{x}) = 0, \forall i = 1, \ldots, G, \forall j = 1, \ldots, H$, where G and H are the number of inequality and equality constraints, respectively. Function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ contains the objective functions.

Let $\mathbf{x^1}, \mathbf{x^2} \in \mathcal{X}$. Solution $\mathbf{x^1}$ 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 so-called *Pareto front* in the objective space.

B. Genetic algorithms

GAs simulate biological evolution [14] to solve optimization problems. GAs encode possible solutions as vectors (*individuals*) made up of binary, integer or real elements (*genes*). A *fitness function* measures each individual's goodness.

A GA generates a random *initial population* where seeding techniques may inject good solutions. The higher an individual's fitness, the more likely it will be selected for reproduction. Selected individuals evolve through *crossover* and *mutation*, thus producing one or more *offspring* which replace part of the population on the basis of their fitness. GAs iterate until a stop condition is met and the fittest individual in the last population is generally the optimal solution.

C. Extremal optimization

EO is a relatively recent nature-inspired optimization technique [20] based on the Bak-Sneppen model of self-organized criticality [21]. EO evolves a single solution $\mathbf{s} = (s_1, \ldots, s_n)$, the *ecosystem*. Each s_i is a *species*, and is assigned a *local fitness* ϕ_i to measure its contribution to the *global fitness* $\Phi(\mathbf{s})$. Evolution evolves $\mathbf{s}$ into $\tilde{\mathbf{s}}$ by modifying the species that gives the worst contribution. If $\Phi(\tilde{\mathbf{s}})$ is higher than the fitness of the best solution found so far, then $\tilde{\mathbf{s}}$ becomes the new best one and the evolution continues. In order to avoid local optima, a modified version (so-called τ-EO) has been proposed to make a probabilistic selection of the species to mutate, where the lower its local fitness, the more likely each species is to mutate.

III. Optimization problem

A. Background

Consider a product made up of N parts, and let $d_1, \ldots, d_D$ be the tasks to dismount it. Each task d_i removes one or more parts from the product (or separates it into subassemblies) and takes t_i minutes. No task can be carried out until its *prior tasks* are performed, which then determine the *technological precedence constraints (TPCs)*. Each task d_i gets a profit $\pi_i \in \mathbb{R}$ defined as

$$\pi_i = r_i + s_i - (c_i^{DIS} + c_i^{DISP}), \tag{1}$$

where r_i is the revenue that comes from selling the recyclable materials, s_i is the income of selling the recoverable

components, c_i^{DIS} is the disassembly cost which stems from multiplying the hourly cost of labor by $(t_i/60)$, and c_i^{DISP} is the disposal cost for nonrecoverable parts.

A *disassembly sequence* is made of part (all) of the tasks $d_1, \ldots, d_D$ when performing partial (complete) disassembly, and must satisfy the TPCs.

Let $w_1, \ldots, w_W$ be the W workstations of a disassembly line. The tasks of a disassembly sequence are assigned to the workstations so that the product spends no more than the *cycle time* T at each station w_k. The cycle time, expressed in minutes, is equal to

$$T = \eta_L (60/R) \tag{2}$$

where $\eta_L \in (0, 1)$ is the line efficiency, i.e. the ratio of working time to total time. The latter considers set up times, faults, etc. Term R is the *production rate*, i.e. the number of products to dismount per hour, defined as

$$R = \frac{D_{year}}{W_{year} \cdot S_{week} \cdot H_{shift}}, \tag{3}$$

where D_{year} is the annual demand for products, W_{year} is the number of working weeks per year, S_{week} is the number of shifts per week, and H_{shift} is the number of hours per shift.

B. Problem formulation

Let $\mathbf{x} \in \{0,1\}^{D \times W}$ be an assignment of the disassembly tasks to the workstations of a line. Let $\mathbf{x}$ contain elements

$$x_{ik} = \begin{cases} 1 & \text{if task } d_i \text{ is assigned to station } w_k \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Let $\mathbf{P} = [p_{ij}]$ be a $D \times D$ matrix where $p_{ij} = 1$ if d_i must be performed before d_j (according to the TPCs), and $p_{ij} = 0$ otherwise. The problem is formulated as follows:

$$\underset{\mathbf{x}}{\text{Maximize}}\ \mathbf{f}(\mathbf{x}) = \tag{5a}$$

$$\left[\sum_{i=1}^{D} \sum_{k=1}^{W} \pi_i x_{ik},\ -\sum_{k=1}^{W} \max_{i=1}^{D} x_{ik},\ \frac{1}{D} \sum_{i=1}^{D} \sum_{k=1}^{W} x_{ik} \right]$$

subject to:

$$\sum_{k=1}^{W} x_{ik} \leq 1, \quad \forall i = 1, \ldots, D \tag{5b}$$

$$\sum_{i=1}^{D} t_i x_{ik} \leq T, \quad \forall k = 1, \ldots, W \tag{5c}$$

$$\sum_{k' \leq k} \sum_{j \neq i} p_{ji} x_{jk'} - \sum_{j \neq i} p_{ji} = 0,\ \forall i = 1, \ldots, D,\ \forall k = 1, \ldots, W \tag{5d}$$

$$\left\lceil \frac{1}{T} \sum_{i=1}^{D} \sum_{k=1}^{W} t_i x_{ki} \right\rceil \leq \sum_{k=1}^{W} \max_{i=1}^{D} x_{ik} \leq D \tag{5e}$$

$$x_{ik} \in \{0,1\},\ \forall i = 1, \ldots, D,\ \forall k = 1, \ldots, W. \tag{5f}$$

Equation (5a) is the objective function $\mathbf{f} : \{0,1\}^{D \times W} \rightarrow \mathbb{R} \times \mathbb{Z}^- \times [0,1]$, whose components are profit, the opposite of the NoW, and disassembly depth of $\mathbf{x}$. Constraints (5b) let each task be performed at one (or no) workstation. Constraints (5c) avoid the cycle time T to be exceeded at each workstation. Equations (5d) are the precedence constraints. Constraint (5e) ensures that the NoW is between the lowest and the highest NoW. Constraints (5f) force binary variables.

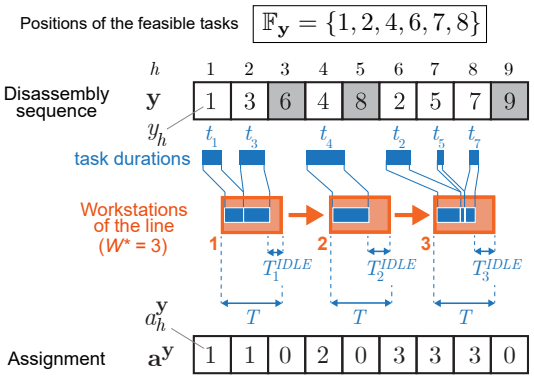

Figure 1. Encoding of a disassembly sequence $\mathbf{y}$ and assignment $\mathbf{a}^{\mathbf{y}}$ of its feasible tasks (white background) to the workstations, given cycle time T.

IV. Resolution Methodology: EMOGA

A. Encoding

A disassembly sequence is encoded as an integer vector $\mathbf{y} \in Sym(D)$, whose elements y_h (also referred to as "genes", "species" or "tasks") are task identifiers, and $Sym(D)$ is the symmetric group of degree D. A sequence $\mathbf{y} \in Sym(D)$ may contain *infeasible tasks*, i.e. tasks that violate some TPC. Let

$$\mathbb{F}_{\mathbf{y}} = \left\{ h : \bigcup_{h'<h} y_{h'} \supseteq \bigcup_{i:p_{ij}=1} y_i,\ j = y_h \right\} \tag{6}$$

contain the positions h of the feasible tasks of $\mathbf{y}$, i.e. each task y_h whose prior tasks y_i are a subset of the ones that come before it in $\mathbf{y}$, i.e. at positions $h' < h$.

The tasks of $\mathbf{y}$ are assigned to the workstations as follows. Let $\mathbf{a}^{\mathbf{y}} \in \{0, \ldots, W\}^D$ be an assignment. Each element $a_h^{\mathbf{y}}$ of $\mathbf{a}^{\mathbf{y}}$ contains the identifier of the workstation to which task y_h is assigned. Infeasible tasks are assigned to a dummy workstation '0'. From $h = 1$ to $h = D$, each task y_h of $\mathbf{y}$ is assigned to

$$a_h^{\mathbf{y}} = \begin{cases} 0 & \text{if } h \notin \mathbb{F}_{\mathbf{y}} \\ \begin{cases} 1 & \text{if } h = 1 \\ a_{h-1}^{\mathbf{y}} & \text{if } T_{h-1}^{IDLE} \geq t_{y_h} \\ 1 + a_{h-1}^{\mathbf{y}} & \text{otherwise} \end{cases} & \text{otherwise} \end{cases} \tag{7}$$

where $T_{h-1}^{IDLE} = T - \sum_{j \in \mathbb{F}_{\mathbf{y}}, a_j^{\mathbf{y}} = h-1} t_{y_j}$ is the idle time of station $a_{h-1}^{\mathbf{y}}$. Encoding and assignment are shown in Fig. 1.

B. Local fitness

1) Profit: Consider every feasible task y_h of sequence $\mathbf{y}$. The profit local fitness of y_h is deemed as much better the higher the profit of both task y_h and the feasible tasks y_j, $j \in \mathbb{F}_{\mathbf{y}}$, that follow y_h. The profit local fitness is modeled as:

$$\phi_{PROFIT}(y_h) = \pi_{y_h} + \sum_{j \in \mathbb{F}_{\mathbf{y}},\, j > h} \pi_{y_j}. \tag{8}$$

2) Number of workstations: This local fitness is calculated by first assigning the tasks of $\mathbf{y}$ to the workstations (see Section IV-A), so as to get the number $W^\star$ of workstations required and measure the idle time T_k^{IDLE} at each station $k = 1, \ldots, W^\star$. The shorter T_k^{IDLE} the less likely a task will be able to replace one of those currently assigned to station k so as to diminish T_k^{IDLE} and maybe reduce the NoW, while meeting the same number of TPCs. Each feasible task y_h s.t. $h \in \mathbb{F}_{\mathbf{y}}$ is thus assigned a local fitness whose goodness is in proportion to the idle time of the station that hosts y_h. The local fitness related to the NoW is

$$\phi_{STATIONS}(y_h) = T \; - \sum_{j \in \mathbb{F}_{\mathbf{y}},\, a_j^{\mathbf{y}} = a_h^{\mathbf{y}}} t_{y_j}, \tag{9}$$

where the right-hand side is the idle time of workstation $a_h^{\mathbf{y}}$, where task y_h is performed.

3) Depth: Every task y_h of $\mathbf{y}$ is assigned a depth local fitness which is lower, the more tasks y_h it requires to be feasible in $\mathbf{y}$. The local fitness is therefore

$$\phi_{DEPTH}(y_h) = \sum_{q \in \mathbb{F}_{\mathbf{y}},\, q<h} p_{y_q y_h} - \sum_{i=1}^{D} p_{i y_h}, \tag{10}$$

where the first summation counts how many prior tasks y_q of y_h come before it in $\mathbf{y}$, and are feasible. The second summation is the number of prior tasks of y_h.

C. Global fitness

The vector-valued function that measures the global fitness is $\mathbf{\Phi}(\mathbf{y}) = [\Phi_{PROFIT}(\mathbf{y}), -\Phi_{STATIONS}(\mathbf{y}), \Phi_{DEPTH}(\mathbf{y})]$. Its components are calculated using the feasible genes of $\mathbf{y}$ as explained in the following subsections.

1) Profit: Each feasible gene y_h of $\mathbf{y}$ determines a profit $\pi_{y_h} \in \mathbb{R}$ defined in (1). The profit of sequence $\mathbf{y}$ is therefore

$$\Phi_{PROFIT}(\mathbf{y}) \; = \sum_{h \in \mathbb{F}_{\mathbf{y}}} \pi_{y_h}, \tag{11}$$

where $\Phi_{PROFIT} : \bigcup_{H=1}^{D} Sym(H) \to \mathbb{R}$.

2) Number of workstations: This global fitness is equal to the NoW required by sequence $\mathbf{y}$, that is

$$\Phi_{STATIONS}(\mathbf{y}) \; = \max_{i=1}^{D} \; a_i^{\mathbf{y}} \tag{12}$$

where $\Phi_{STATIONS} : \bigcup_{H=1}^{D} Sym(H) \to \mathbb{Z}^+$.

3) Disassembly depth: The disassembly depth global fitness of sequence $\mathbf{y}$ is the ratio of the number of feasible tasks in $\mathbf{y}$ to the number of all disassembly tasks, i.e.

$$\Phi_{DEPTH}(\mathbf{y}) = \frac{|\mathbb{F}_{\mathbf{y}}|}{D}, \tag{13}$$

where $\Phi_{DEPTH} : \bigcup_{H=1}^{D} Sym(H) \to [0, 1]$, and $|\cdot|$ denotes the cardinality of a set.

D. Initial population

The initial population $\mathbb{Y}_0 = \mathbb{Y}_0^{seed} \cup \mathbb{Y}_0^{rand}$ is made up of n individuals, where m individuals ($\mathbb{Y}_0^{seed}$) come from seeding, and the remaining $n - m$ individuals ($\mathbb{Y}_0^{rand}$) are randomly generated permutations of the disassembly tasks $d_1, \ldots, d_D$.

E. Seeding technique

Seeding solutions are obtained by solving m continuous relaxations of the problem (5a)-(5f), each maximizing function

$$\begin{aligned} z^j(\mathbf{x}) = {} & \omega_1^j \textstyle\sum_i \sum_k \pi_i^N x_{ik} - \omega_2^j (D \cdot T - \sum_i \sum_k x_{ik} t_i^N) \\ & + \omega_3^j (\tfrac{1}{D} \textstyle\sum_i \sum_k x_{ik}), \end{aligned} \tag{14}$$

where $j = 1, \ldots, m$, (5d) are relaxed, (5f) becomes $0 \leq x_{ik} \leq 1 \; \forall i, k$, terms π_i^N and t_i^N are the normalized profit and duration of task d_i respectively, $\boldsymbol{\omega}^{\mathbf{j}} = (\omega_1^j, \omega_2^j, \omega_3^j) \in \mathbb{R}_+^3$, and $\omega_1^j + \omega_2^j + \omega_3^j = 1$. Term $(D \cdot T - \sum_i \sum_k x_{ik} t_i^N)$ is the total idle time, which is considered as it is linear and because its minimization implies that the NoW is minimized [3].

Let $\hat{\mathbf{x}}^{\mathbf{j}}$ be the optimal solution of the continuous relaxation of (5a)-(5f) using $\boldsymbol{\omega}^{\mathbf{j}}$. Seeding solutions are obtained as:

1) generate a set $\mathbf{\Omega}$ of $m/2$ weight vectors by uniform sampling, so $\omega_1^j = (\frac{2}{m})3h$, $\omega_2^j = (\frac{2}{m})3k$ where $k \leq h$ and $h, k = 1, \ldots, \frac{1}{3}(\frac{m}{2})$, with $\omega_3^j = 1 - (\omega_1^j + \omega_2^j)$;
2) solve the continuous relaxations and consider solutions $\lfloor \hat{\mathbf{x}}^{\mathbf{j}} \rfloor$, $\forall \boldsymbol{\omega}^{\mathbf{j}} \in \mathbf{\Omega}$, then push infeasible tasks forward along the sequence trying to increase the feasibility;
3) for each $\lfloor \hat{\mathbf{x}}^{\mathbf{j}} \rfloor$, calculate its Euclidean norm $e_j = ||\lfloor \hat{\mathbf{x}}^{\mathbf{j}} \rfloor||$;
4) for each $\lfloor \hat{\mathbf{x}}^{\mathbf{j}} \rfloor$, measure the distance d_{ju} from its nearest neighbor $\lfloor \hat{\mathbf{x}}^{\mathbf{u}} \rfloor$, i.e. $d_{ju} = e_j - e_u$, $u = \arg\min_{v \neq u} d_{jv}$;
5) for each $\lfloor \hat{\mathbf{x}}^{\mathbf{j}} \rfloor$ whose nearest neighbor $\lfloor \hat{\mathbf{x}}^{\mathbf{u}} \rfloor$ is considered too far, i.e. $d_{ju} > \frac{2\lambda}{m} \sum_{q=1}^{m/4} d_{qu}$, try to reduce the gap by repeatedly generating $\boldsymbol{\omega}_\ell = (\omega_1^\ell, \omega_2^\ell, \omega_3^\ell)$, where $\omega_1^\ell \sim \mathcal{U}(0, 1)$, $\omega_2^\ell \sim \mathcal{U}(0, \omega_1^\ell)$ (where $\mathcal{U}(a, b)$ is the uniform probability distribution in $[a, b]$), and $\omega_3^\ell = 1 - (\omega_1^\ell + \omega_2^\ell)$, until $e_j < ||\lfloor \hat{\mathbf{x}}^\ell \rfloor|| < e_u$ or a max time period runs out.

Solutions $\lfloor \hat{\mathbf{x}}^{\mathbf{j}} \rfloor$ are encoded as explained in Section IV-A before injection.

F. The push-swap mutation operator

The push-swap mutation (PSM) is a tailored operator with a twofold behavior that depends on the feasibility of the species undergoing mutation. When feasible, PSM performs in *push* mode, otherwise in *swap* mode.

This behavior was designed to get a perfect fit between the EO and GA modules of EMOGA, in order to obtain better solutions and increase their diversity in the EO loop, without excessively undoing the exploitation made by the crossover in the genetic loop (see Section IV-G).

1) Swap mode: Consider a disassembly sequence $\mathbf{y}$ and let the species $\tilde{y}_h$ to mutate be infeasible. Each y_j such that $j \neq h$ is assigned a random number $\rho_j \sim \mathcal{U}(0, 1)$. Species $\tilde{y}_h$ is then exchanged with the species y_k s.t. $k = \arg\max_{j \in \{1, \ldots, D\} \setminus h} \rho_j$. The swap mode is shown on the left-hand side in Fig. 2.

2) Push mode: Consider a disassembly sequence $\mathbf{y}$ and let the species $\tilde{y}_h$ to mutate be feasible. Let $\tilde{\mathbf{n}}_{\mathbf{h}}$ be the *feasible neighborhood* of $\tilde{y}_h$, which contains $\tilde{y}_h$ and all the feasible species of $\mathbf{y}$ encountered starting from $\tilde{y}_h$ and moving first backward and then forward along $\mathbf{y}$, without going through infeasible species. Push moves all species in $\tilde{\mathbf{n}}_{\mathbf{h}}$ one position back. An example is shown on the right-hand side in Fig. 2.

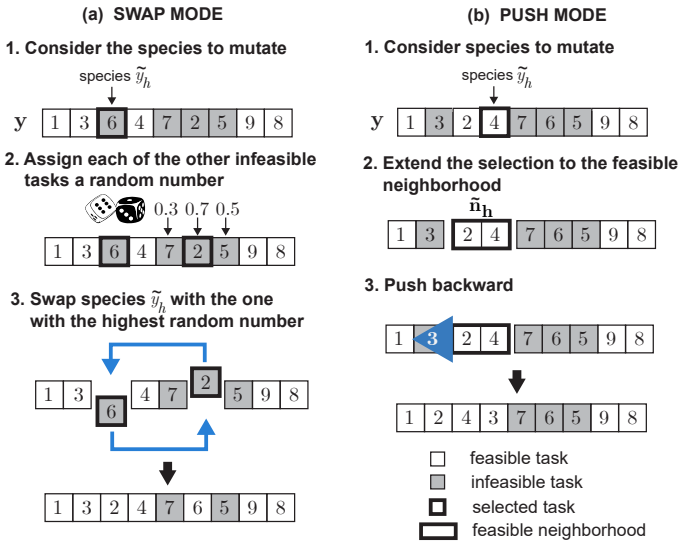

Figure 2. PSM for a product requiring nine tasks to be completely disassembled. Parts (a) and (b) show the swap and push modes, respectively.

G. Description of EMOGA

EMOGA generates an initial population $\mathbb{Y}_0$ of n individuals (see Section IV-D) which becomes the current population $\mathbb{Y}$.

The genetic loop starts assigning individuals their global fitness. Each individual then gets a non-domination rank (equal to the number of solutions that dominate it), based on which individuals are sorted. EMOGA selects pairs of individuals using a binary tournament with a crossover rate α, and crosses them over using the best-order crossover (BOX). BOX is used as it is more efficient than many others for combinatorial problems [22]. BOX gets two offspring $\mathbf{s^1}$ and $\mathbf{s^2}$ which enter the EO loop according to a probability β.

Let us suppose that $\mathbf{s^i}$ enters the EO loop. EMOGA assigns its species s^i_j a local fitness $\phi(s^{i,o}_j)$ for each objective $o \in \mathbb{O}$, with $\mathbb{O} \triangleq \{PROFIT, STATIONS, DEPTH\}$, and initializes an empty archive $\mathbb{A}_i$. The species of $\mathbf{s^i}$ are sorted w.r.t. decreasing local fitness to get a ranking $\mathbf{r^{i,o}}$ $\forall o \in \mathbb{O}$. All species $r^{i,o}_j$ of each ranking are assigned a mutation probability $(r^{i,o}_j)^{-\tau}$, $\tau > 0$, which is proportionally higher, the worse the species. EMOGA selects a species $\tilde{r}^{i,o}$ to mutate $\forall o \in \mathbb{O}$, and then one or more of these species undergo the PSM to get a new solution $\tilde{\mathbf{s}}^\mathbf{i}$. The species that actually mutate(s) are chosen depending on the iteration. In the first iterations, EMOGA chooses species $\tilde{r}^{i,PROFIT}$, then $\tilde{r}^{i,STATIONS}$, and finally $\tilde{r}^{i,DEPTH}$. The remaining iterations are equally divided into a number of groups corresponding to the number of possible combinations of the objectives. Each group mutates a combination of more than one $\tilde{r}^{i,o}$. Each time, solution $\tilde{\mathbf{s}}^\mathbf{i}$ replaces the previous one, and if $\tilde{\mathbf{s}}^\mathbf{i}$ dominates some solutions in archive $\mathbb{A}_i$, it takes their place. If $\mathbb{A}_i = \emptyset$, solution $\tilde{\mathbf{s}}^\mathbf{i}$ is put into $\mathbb{A}_i$. The EO loop performs *max_eo* iterations, then merges $\mathbb{A}_i$ with $\mathbb{Y}$. Individuals are sorted w.r.t. increasing non-domination rank. The first n individuals pass into the new population. The genetic loop thus continues up to *max_epochs*.

H. Pseudocode

The pseudocode of EMOGA is in Algorithm 1, where:

- `crossover`: performs the BOX of the individuals as arguments, and returns two offspring;
- `mutate`: performs the PSM on the solution passed as the first argument, mutating one or more of its species (whose positions are the next arguments) based on *eo_step*.

Algorithm 1 EMOGA

INPUT: $\boldsymbol{\pi}, \mathbf{t}, D, W, \mathbf{P}, max\_epochs, max\_eo, n, \tau, \alpha, \beta$
OUTPUT: $\mathbb{Y}$
generate $\mathbb{Y}_0^{rand}$ and $\mathbb{Y}_0^{seed}$ and initialize $\mathbb{Y}_0$ with them
epoch $\leftarrow 1$
——— *GENETIC LOOP* ———
$\mathbb{Y} \leftarrow \mathbb{Y}_0$
while *current_epoch* $\leq$ *max_epochs* **do**
  evaluate global fitness $\Phi_{PROFIT}(\mathbf{y})$, $\Phi_{STATIONS}(\mathbf{y})$, $\Phi_{DEPTH}(\mathbf{y})$ of each solution $\mathbf{y} \in \mathbb{Y}$
  sort the individuals based on non-domination rank
  choose $\mathbf{y^1}, \mathbf{y^2} \in \mathbb{Y}$ with binary tournament, using α
  $(\mathbf{s^1}, \mathbf{s^2}) \leftarrow$ `crossover`$(\mathbf{y^1}, \mathbf{y^2})$
  for each $\mathbf{s^i}$ selected with probability β **do**
    eo_step $\leftarrow 1$
    $\mathbb{A}_i \leftarrow \emptyset$, $\mathbb{O} \leftarrow \{PROFIT, STATIONS, DEPTH\}$
    ——— *EXTREMAL OPTIMIZATION LOOP* ———
    while *eo_step* $\leq$ *max_eo* **do**
      $\forall\ s^i_j$ of $\mathbf{s^i}$, compute local fitnesses $\phi_{PROFIT}(s^i_j)$, $\phi_{STATIONS}(s^i_j)$, $\phi_{DEPTH}(s^i_j)$
      compute $\mathbf{r^{i,PROFIT}}$, $\mathbf{r^{i,STATIONS}}$, $\mathbf{r^{i,DEPTH}}$
      assign each $r^{i,o}_j$ of $\mathbf{r^{i,o}}$ probability $(r^{i,o}_j)^{-\tau}, \forall o \in \mathbb{O}$
      select the worst species $\tilde{r}^{i,o}, \forall o \in \mathbb{O}$
      $\tilde{\mathbf{s}}^\mathbf{i} \leftarrow$ `mutate`$(\mathbf{s^i}, \tilde{r}^{i,PROFIT}, \tilde{r}^{i,STATIONS}, \tilde{r}^{i,DEPTH}, eo\_step)$
      replace with $\tilde{\mathbf{s}}^\mathbf{i}$ the solutions of $\mathbb{A}_i$ it dominates
      $\mathbf{s^i} \leftarrow \tilde{\mathbf{s}}^\mathbf{i}$
      eo_step $\leftarrow$ *eo_step* $+ 1$
    end while
    put the solutions of $\mathbb{A}_i$ into $\mathbb{Y}$
  end for
  cut $\mathbb{Y}$ to n individuals by non-dominated sorting
  epoch $\leftarrow$ *epoch* $+ 1$
end while
return $\mathbb{Y}$

V. Results and Discussion

EMOGA was implemented in Java and MATLAB and was tested on two real-world case studies related to the disassembly of a hammer drill and a microwave oven, respectively. These products were chosen as they are fairly complex, widespread, in high-demand and highly subject to disassembly for recycling. The experiments were carried out on a workstation with an Intel i7 Quad-Core CPU at 3.4 GHz and 16 GB of RAM.

A. Parameters of the algorithm

EMOGA requires population size n, crossover rate α, mutation probability β, and τ (see Section II-C). The value

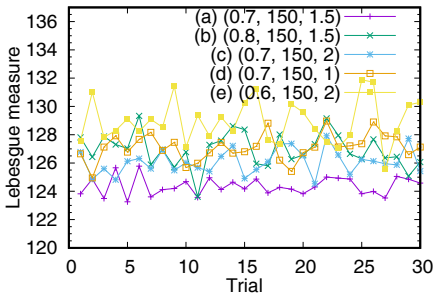

Figure 3. Performance of EMOGA with various configurations of parameters.

of β was set to $\beta = 0.1\alpha$ considering that in GAs $\beta \ll \alpha$, and good mutation probabilities are typically one or two orders of magnitude lower than the crossover probability [14].

To find the best configuration $(\alpha^\star, n^\star, \tau^\star)$, three values per parameter were first chosen on the basis of a trial and error analysis: $\alpha \in \{0.6, 0.7, 0.8\}$, $n \in \{150, 200, 250\}$ and $\tau \in \{1, 1.5, 2\}$. These values were combined in all possible ways thus obtaining 27 configurations. For each configuration, a total of 30 trials were run. The max number of generations was 1000. For each trial, the Lebesgue measure (or hypervolume) of the Pareto front was determined and then the mean was calculated. Means were compared to each other, and Student's t-test with 95% confidence was used for validation.

Fig. 3 shows the performance of EMOGA over 30 trials when using the five best configurations tested. The performance of each configuration (α, n, τ) is measured in terms of the Lebesgue measure. The scatter plot refers to the microwave oven disassembly, i.e. the more complex of the two case studies. The best results were obtained with configuration (a), which is thus used in the following as a baseline.

Configuration (b) — which has a higher crossover rate w.r.t. the baseline ($\alpha = 0.8$ instead of $\alpha = 0.7$) — produces quite higher (i.e. worse) values of the Lebesgue measure. This is most likely the effect of a crossover rate that is too high.

A similar performance was obtained with configuration (d), which differs from the baseline because of the lower value of τ ($\tau = 1$ instead of $\tau = 1.5$). Decreasing τ thus results in a performance loss that is proportional to the one obtained with configuration (b).

Even configuration (c) — which is characterized by a higher value of τ w.r.t. the baseline ($\tau = 2$ instead of $\tau = 1.5$) — produces a worse performance w.r.t. the baseline. When τ is increased, the EO core is less likely to replace species different from the worst one, and this may lead to worse solutions. Finally, the scatter plot of configuration (e) shows that decreasing the crossover rate while increasing τ — thus giving less freedom to the EO core — leads to unstable results.

B. Case study I: hammer drill disassembly

1) Description of the product: This case study relates to the disassembly of the hammer drill shown in Fig. 4. The precedence graph is in Fig. 5. The 22 tasks for complete disassembly are described in Table I, where part identifiers are preceded by '#' and refer to Fig. 4. Subassemblies are denoted with the list of their part identifiers in curly brackets.

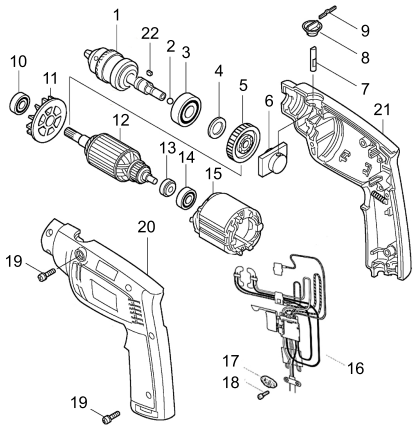

Figure 4. Exploded view of the hammer drill considered in case study I.

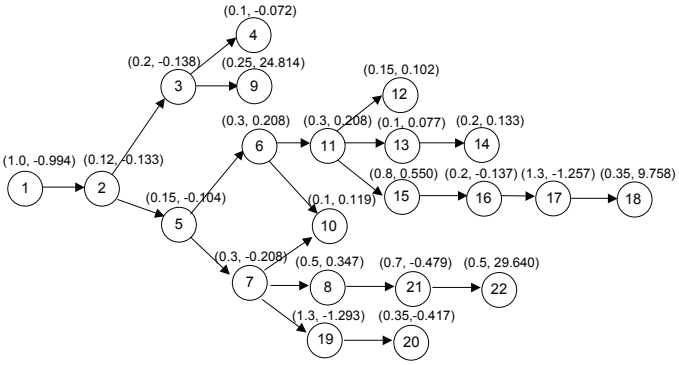

Figure 5. Precedence graph of the hammer drill. Nodes contain task identifiers and the pairs above contain task duration [min] and profit [€], in the order.

Table I
PARTS REMOVED BY EACH TASK FOR HAMMER DRILL DISASSEMBLY

ID	Removed parts	ID	Removed parts
1	Screws #19 from housing set #20	12	Steel ball #2 from #1
2	Housing set #20	13	Knob #8 from {#6,#7}
3	Screws #18 from #17 and #21	14	Shifter pin #7 from cam #6
4	Strain relief #17	15	Helical gear #5 from #1
5	Leaf spring #9 from housing set #21	16	Flat washers #4 from #1
6	{#1,...,#8,#22} from housing set #21	17	Ball bearing #3 from #1
7	{#10,...,#16} from housing set #21	18	Key #22 from spindle #1
8	{#10,...,#14} from rotor housing #15	19	Ball bearing #10 from #12
9	#16 from rotor housing #15	20	Fan #11 from #12
10	Housing set #21	21	Ball bearing #14 from #12
11	{#6,...,#8} from {#1,...,#8,#22}	22	Seal #13 from #12

2) Setup and parameters: The company involved dismounts $D_{year} = 45,000$ drills per year, in a disassembly line that works $W_{year} = 42$ weeks/year with $S_{week} = 5$ shifts/week and $H_{shift} = 8$ hours/shift. The line efficiency is $\eta_L = 0.9$. Thus, $T = 2$ min (see (2) and 3)).

EMOGA was set up with $\alpha = 0.7$, $n = 150$, $\tau = 1.5$, $m = 30$ and $\lambda = 2$.

3) Discussion: Fig. 9 shows one of the best Pareto fronts obtained. The best solution is typically selected by the managers on the basis of both the availability of production resources and the situation of the market of recycled materials

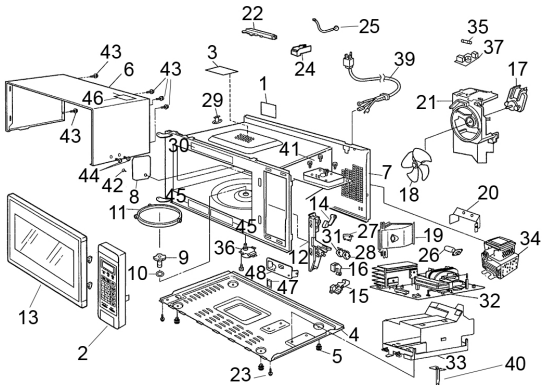

Figure 6. Exploded view of the microwave oven considered in case study II.

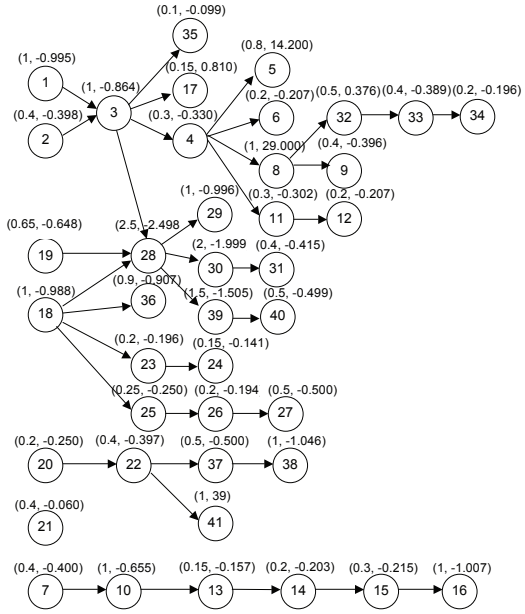

Figure 7. Precedence graph of the microwave oven. Nodes are task identifiers and the pairs above contain task duration [min] and profit [€], in the order.

Table II
PARTS REMOVED BY EACH TASK FOR MICROWAVE OVEN DISASSEMBLY

ID	Removed parts	ID	Removed parts
1	Screws #43	22	Base #4
2	Screws #44	23	Cover #8
3	Cabinet body #6	24	Clip #42 from cover #8
4	Orifice #21	25	Roller #11
5	Fan motor #17	26	Component {#9,#10}
6	Fan blade #18	27	Washer #10 from turntable coupling #9
7	Accordion #39	28	Block {#12,...,#16,#27,#28,#31}
8	Magnetron #34	29	Microwave micro switch {#28,#31}
9	Screws #41	30	Hook levers {#14,...,#16}
10	Oven #7	31	Micro switch #27 from door hook #12
11	Fuse #35	32	Inverter #32
12	Noise filter #37	33	Airguides {#19,#20}
13	Cover #1	34	Lamp #26
14	Cover #22	35	Cutout #29
15	Sensor cover #24	36	Labels {#30,#45,#46}
16	Sensor #25	37	Block {#33,#40}
17	Cover #3	38	Ground plat #40 from bracket #33
18	Frontal panel #13	39	Door bracket #36
19	Control panel #2	40	Stopper # 47 from #36
20	Screws #23	41	Motor #38
21	Foots #5		

and spare parts. Two representative situations were considered: i) the existing disassembly line with limited NoW with amortized equipment; ii) the need for a reduction in personnel.

In the first situation, the company involved has a disassembly line with five workstations. Managers thus chose the solution with the highest profit and NoW$\leq$ 5, i.e. $\mathbf{y^1} = (1, 2, 3, 9, 5, 7, 8, 21, 22)$, whose assignment is $\mathbf{a^{y^1}} = (1, 1, 1, 1, 1, 2, 2, 2, 2)$. Infeasible tasks are neglected. Solution $\mathbf{y^1}$ is circled in the relative front in Fig. 9, and its global fitness is $\mathbf{\Phi}(\mathbf{y^1}) = (44.40, 2, 0.59)$.

In the second situation, managers chose the solution with the highest profit and NoW$=$ 5, i.e. $\mathbf{y^2} = (1, 2, 5, 7, 3, 19, 20, 8, 21, 22, 6, 9, 11, 15, 16, 17, 18)$, whose assignment is $\mathbf{a^{y^2}} = (1, 1, 1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5)$ (see the relative Pareto front in Fig. 9). The global fitness is $\mathbf{\Phi}(\mathbf{y^2}) = (48.39, 5, 0.865)$. The upper part of Fig. 8 shows the allocation of the tasks of $\mathbf{y^1}$ and $\mathbf{y^2}$ to the workstations, in terms of task durations. As can be seen, EMOGA can obtain leveled idle times at the workstations.

C. Case study II: microwave oven disassembly

1) Details on the product and motivation: The second case study refers to the disassembly of a microwave oven, shown in Fig. 6. This product was chosen as they are common items in the home, and because it enabled us to test and investigate both the performance and scalability of EMOGA. The precedence graph is in Fig. 7. The 41 tasks are described in Table II.

2) Setup and parameters: The company involved dismounts $D_{year} = 30,000$ ovens per year. The line has the same characteristics as the one in Section V-B2. Using (2) and (3), $R = 18$ products/hour and $T = 3$ min.

EMOGA was set up with $\alpha = 0.7$, $n = 300$, $\tau = 1.5$, $m = 30$ and $\lambda = 2$.

3) Discussion: One of the best Pareto fronts is in Fig. 9. Still considering the situations in Section V-B3, here managers selected the circled solutions, i.e. $\mathbf{y^1} = (20, 22, 2, 41)$ and $\mathbf{y^2} = (2, 20, 1, 3, 21, 17, 22, 18, 4, 5, 41, 19, 8)$, whose fitnesses are $\mathbf{\Phi}(\mathbf{y^1}) = (28.27, 1, 0.512)$ and $\mathbf{\Phi}(\mathbf{y^2}) = (57.4, 3, 0.61)$. The company has a line with three workstations, managers thus selected the solution with the highest profit and NoW≤ 3, i.e. $\mathbf{y^2}$. In the case of an unexpected lack of personnel with, e.g., just one operator available, managers selected solution $\mathbf{y^1}$, with one workstation and max profit. The bottom part of Fig. 8 shows that the allocation of the tasks of $\mathbf{y^1}$ and $\mathbf{y^2}$ to the workstations achieves leveled idle times.

VI. PERFORMANCE EVALUATION AND VALIDATION

To the best of the authors' knowledge, there is no Pareto-based approach in the literature that considers the same objectives. The authors thus structured the performance evaluation in four parts to show that on average:

1) EMOGA leads to better solutions w.r.t. EO;
2) EMOGA outperforms two variants of the algorithm that handle the TPCs as constraints;
3) EMOGA is better than an efficient Pareto-based GA that handles the TPCs as constraints, or as an objective;

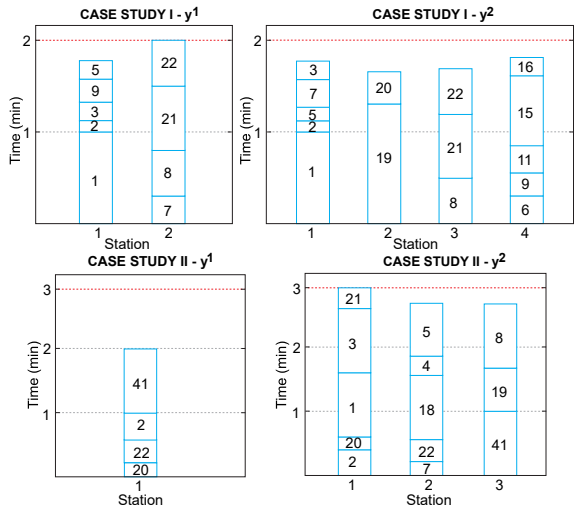

Figure 8. Assignment of the tasks of solutions $\mathbf{y^1}$ and $\mathbf{y^2}$ of case studies I (top) and II (bottom) to the stations, in terms of task durations.

4) EMOGA performs better than other techniques, such as tabu search (TS) and particle swarm optimization (PSO).

Points from 1) to 3) highlight that EMOGA is more efficient than both GAs and EO, so it hybridizes these two techniques in a way that exploits their advantages and avoids their drawbacks. Point 4) shows that EMOGA can also be a valid alternative to other techniques not based on GAs and/or EO, which are efficient when dealing with DLBPs.

The values of the parameters of the compared techniques were found by simulations, using the procedure described in Section V-A. The details are omitted due to lack of space.

A. Statistical validation

The statistical validation was carried out by using Student's t-test, as described in Section V-A. The Lebesgue measure (*Leb*) of the Pareto front and the execution time (*time*) were considered as metrics to evaluate the performance.

B. Performance evaluation

The performance of EMOGA was assessed w.r.t. case study II, i.e. the more complex one. *Leb* was calculated w.r.t. $(0, 0, 0)$, normalizing the objectives in $[0, 1]$ and considering that $\max \mathbf{f} = \min -\mathbf{f}$, so the lower *Leb*, the better.

1) Only EO: EMOGA was compared to τ-EO with Cauchy mutation (CM) and Gaussian mutation (GM). Their merits are: CM easily escapes from local optima and gets close to global optima faster than GM; GM is good in local convergence [23].

Fig. 9 shows the scatter plots of *time* and *Leb* for the trials based on GM (labeled as EO_GM) and CM (labeled as EO_CM), denoted as "1.a" and "1.b", respectively. The average *time*s and *Leb*s can be easily compared to the ones of EMOGA as shown by the box plots on the upper-right in Fig. 9. As highlighted by the scatter plots of *Leb*, EMOGA outperforms the compared algorithms in almost all the trials, but is a bit slower. The increase in *time* was on average +3.17 min. However, the DLBP has no strict time constraint.

2) Precedences as a constraint: EMOGA was compared to two modified versions that handle the TPCs as constraints. Two widely used penalty methods were considered. Repairing techniques were neglected as they are more appropriate when repairing entails low computational costs [19].

EMOGA was first compared to a modified version that uses a static penalty (SP) method. SP methods were considered because they are generally more efficient [19]. EMOGA was then compared to a version based on an efficient parameterless penalty (PP) method [24]. These two methods were chosen as they are efficient and widely used.

The scatter plots comparing EMOGA to the variants based on SP and PP are indicated in Fig. 9 with "2a" and "2b", respectively. Note that here the objectives are profit and NoW. As Fig. 9 shows, the PP-based variant is almost as accurate as EMOGA, but *time* is longer in all trials (+47.4%, on average).

With reference to the SP-based method, the scatter plots in Fig. 9 show that the Pareto front approximation is worse than EMOGA in all trials except one (+15.8% of *Leb*, on average). Note that the SP-based method is slower than EMOGA and requires a time consuming phase to tune the parameters: this time is neglected in the plots.

3) Only Pareto-based genetic algorithm: EMOGA was compared to the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [25], an efficient GA. Inversion mutation and BOX were used as they were shown to be the most efficient for permutation-based problems [22].

Two comparisons were made, first considering TPCs as one of the objectives (i.e., 'depth'), and then as constraints using the PP method. In Fig. 9, the related plots are referred to as "3a" and "3b", respectively. In "3a", EMOGA is faster in almost all trials (+10.3% on average) and makes a better approximation of the Pareto front, cutting *Leb* down from 128.17 to 124.72. In "3b", EMOGA is slower than NSGA-II (+1.84 min on average), but on average gets slightly better solutions w.r.t. those of NSGA-II: *Leb* is 124.42 vs. 125.01 (see the box plots in Fig. 9). EMOGA is thus preferable. This is confirmed by Fig. 9, which shows one of the best fronts obtained by NSGA-II. Considering the solutions chosen in case study II, here managers should choose those indicated by the arrows in Fig. 9, as they are the best alternatives w.r.t. $\mathbf{y^1}$ and $\mathbf{y^2}$: same NoW and closest disassembly depth. However, the profit is lower, i.e. 25.34 € (-10.4%) and 53.81€ (-6.25%), respectively. Considering $D_{year} = 30,000$, with EMOGA managers save 87,900 € and 308,700 € a year, respectively, with an average increase in annual profit of up to 16.7%.

4) Tabu search and Particle Swarm Optimization: Finally, EMOGA was compared to TS and PSO, as they perform better than others when dealing with DLBPs [26]. The scatter plots and box plots of *Leb* and *time* of TS are shown in Fig. 9, and are referred to as "4a". As the figure shows, EMOGA is faster than TS and achieves better solutions. The corresponding box plots in Fig. 9 show that TS converges in 14.68 min on average, so it is 15.6% slower than EMOGA. TS obtains definitely higher values of *Leb* w.r.t. EMOGA (see the box plots in Fig. 9), and thus produces a worse approximation of the Pareto front.

The plots that compare EMOGA to PSO are shown in Fig.

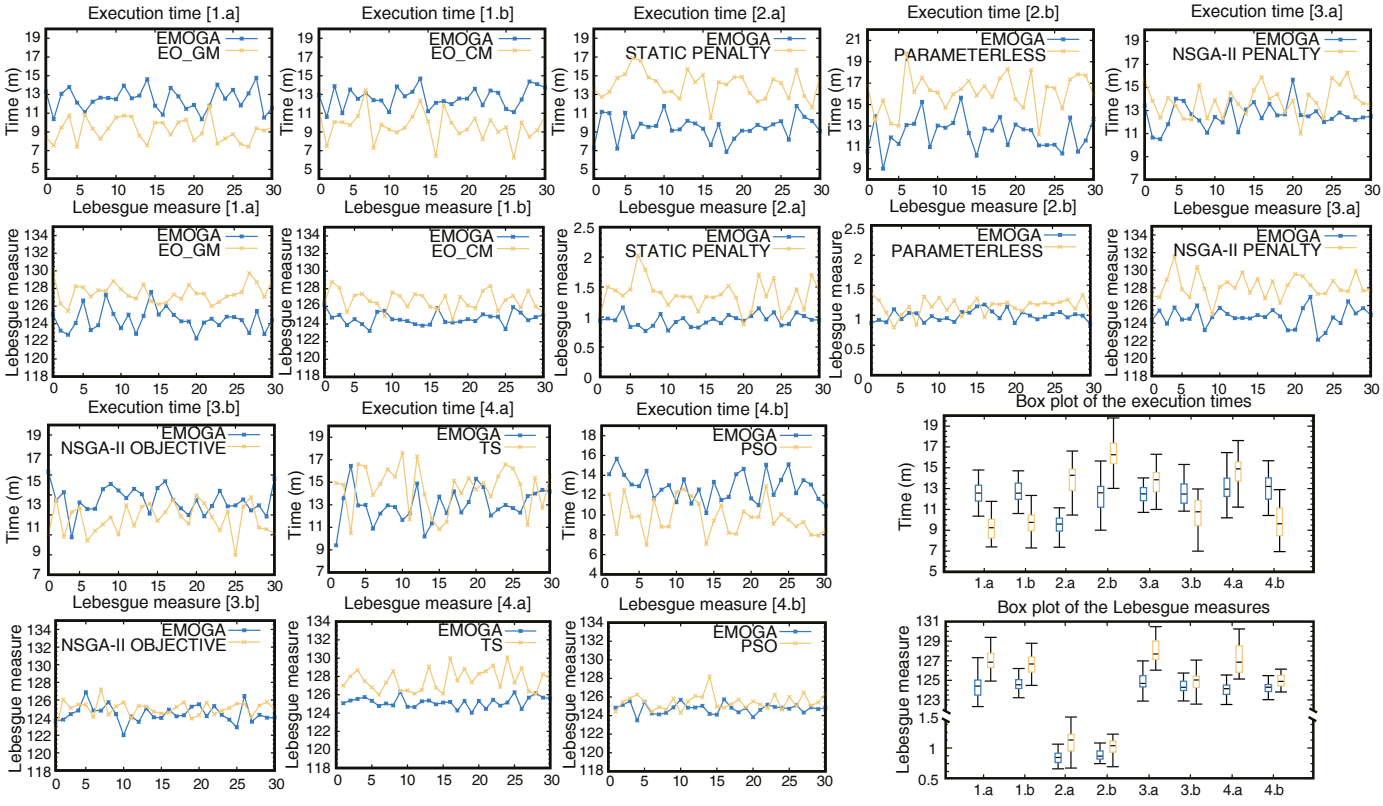

Figure 9. Scatter plots comparing $time$ and Leb for all the scenarios considered in the performance evaluation (i.e. [1.a], [1.b], [2.a], [2.b], [3.a], [3.b], [4.a], [4.b]) over 30 independent trials. Labels on the x-axis relate to the trial identifiers. The bottom-right corner contains the box plots comparing the same quantities for each scenario. The two boxes of each pair (1.a, 1.b, etc.) relate to EMOGA and the compared technique, from left to right.

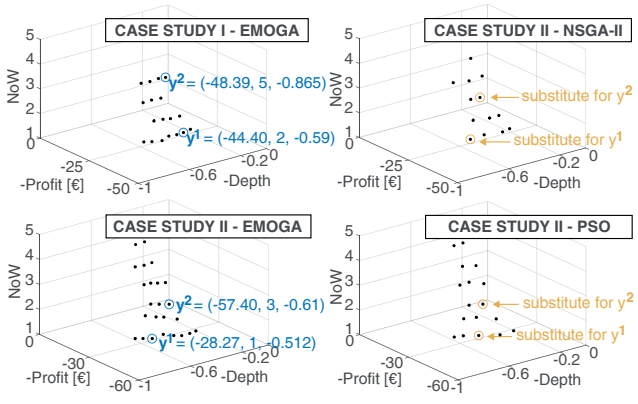

Figure 10. Pareto fronts obtained by EMOGA (left-hand side) for the two case studies, and Pareto fronts obtained for case study II by the techniques used for comparison (right-hand side).

9, and are referred to as "4b". The scatter plots and the box plots of *time* show that PSO is quite faster than EMOGA, ∼3 min on average. However, the scatter plots of *Leb* show that EMOGA obtains better solutions (-2.73% of *Leb*, on average). As PSO may seem no worse than EMOGA, Fig. 10 shows one of the best Pareto fronts obtained by PSO. Managers chose the solutions indicated by the arrows as substitutes for the ones chosen when using EMOGA. These solutions have quite a lower depth (-8.41%) and achieve a lower profit: 26.15 € (-7.5%) and 53.88 € (-6.13%), respectively. With $D_{year} = 30,000$, EMOGA is preferable, as lets the company save 63,600 € and 105,600 €, respectively, with up to 7.32% average increase in annual profit.

VII. Conclusions

This paper has presented EMOGA, a hybrid nature-inspired technique for a multiobjective DLBP. EMOGA is a Pareto-based GA with an EO core that exploits a tailored mutation operator and a continuous relaxation-based seeding technique.

The experiments involved the disassembly of a drill and a microwave oven. A better performance was obtained in all the comparisons using non-hybrid algorithms. EMOGA could thus be a valid alternative to existing techniques as it could lead to more efficient disassembly lines.

References

[1] P. Modak, "Waste: investing in energy and resource efficiency," in *Towards a Green Economy: Pathways to Sustainable Development and Poverty Eradication. United Nations Environment Programme*, 2011.

[2] K. Breivik *et al.*, "Tracking the global generation and exports of e-waste. do existing estimates add up?" *Environmental Sci. & Technology*, vol. 48, pp. 8735–8743, 2014.

[3] S. McGovern and S. Gupta, *The disassembly line: balancing and modeling*. New York, USA: McGraw-Hill, 2011.

[4] S. McGovern and S. M. Gupta, "A balancing method and genetic algorithm for disassembly line balancing," *European J. of Operational Research*, vol. 179, no. 3, pp. 692–708, 2007.
[5] C. Ullerich, *Advanced Disassembly Planning*. Springer, 2014.
[6] G. Tian *et al.*, "A chance constrained programming approach to determine the optimal disassembly sequence," *IEEE Trans. Autom. Sci. and Eng.*, vol. 10, no. 4, pp. 1004–1013, 2013.
[7] 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.
[8] W. Hao and Z. Hongfu, "Using genetic annealing simulated annealing algorithm to solve disassembly sequence planning," *J. of Syst. Eng. and Electron.*, vol. 20, no. 4, pp. 906–912, 2009.
[9] G. Tian *et al.*, "Probability evaluation models of product disassembly cost subject to random removal time and different removal labor cost," *IEEE Trans. Autom. Sci. and Eng.*, vol. 9, no. 2, pp. 288–295, 2012.
[10] W.-C. Yeh, "Optimization of the disassembly sequencing problem on the basis of self-adaptive simplified swarm optimization," *IEEE Trans. Syst. Man Cybern. A., Syst. Humans*, vol. 42, no. 1, pp. 250–261, 2012.
[11] C. B. Kalayci *et al.*, "Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm," *J. of Manuf. Syst.*, vol. 37, no. 3, pp. 672–682, 2015.
[12] O. Ondemir *et al.*, "Optimal end-of-life management in closed-loop supply chains using rfid and sensors," *IEEE Trans. Ind. Informat.*, vol. 8, no. 3, pp. 719–728, 2012.
[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] 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.
[15] S. Boettcher and A. G. Percus, "Extremal optimization: Methods extremal optimization: Methods derived from co-evolution," in *In Proc. of the Genetic and Evol. Comput. Conf.*, 1999.
[16] P. Gómez-Meneses *et al.*, "A hybrid multi-objective extremal optimisation approach for multi-objective combinatorial optimisation problems," in *IEEE Congr. on Evol. Comput.*, 2010, pp. 1–8.
[17] M.-R. Chen *et al.*, "Multiobjective optimization using population-based extremal optimization," *Neural Computing and Applicat.*, vol. 17, no. 2, pp. 101–109, 2008.
[18] Y.-W. Chen *et al.*, "Hybrid evolutionary algorithm with marriage of genetic algorithm and extremal optimization for production scheduling," *The Int. J. of Advanced Manuf. Technol.*, vol. 36, no. 9-10, pp. 959–968, 2008.
[19] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Comput. Methods in Appl. Mechanics and Eng.*, vol. 191, no. 11–12, pp. 1245–1287, 2002.
[20] S. Boettcher and A. G. Percus, "Extremal optimization: methods derived from co-evolution," in *Genetic and Evol. Computation Conf.*, 1999, pp. 825–832.
[21] P. Bak and K. Sneppen, "Punctuated equilibrium and criticality in a simple model of evolution," *Physical Review Lett.*, vol. 71, pp. 4083–4086, 1993.
[22] A. Andreica and C. Chira, "Best-order crossover for permutation-based evolutionary algorithms," *Appl. Intell.*, vol. 42, no. 4, pp. 751–776, 2015.
[23] K.-T. Lan and C.-H. Lan, "Notes on the distinction of gaussian and cauchy mutations," in *Int Conf. on Intell. Syst. Design and Appl.*, 2008, pp. 272–277.
[24] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput. Methods in Appl. Mechanics and Eng.*, vol. 186(2/4), pp. 311–338, 2000.
[25] K. Deb *et al.*, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.
[26] C. B. Kalayci and S. M. Gupta, "A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem," *The Int. J. of Advanced Manuf. Technol.*, vol. 69, no. 1, pp. 197–209, 2013.

Francesco Pistolesi is a Postdoctoral Researcher with the Department of Information Engineering at the University of Pisa, Pisa, Italy. He graduated – *summa cum laude* – from the University of Pisa with a master's degree in computer engineering (*enterprise* curriculum), and got a Ph.D. in information engineering (*computer system architectures* curriculum) from the Leonardo da Vinci doctoral school of engineering of the University of Pisa. His research interests embrace computational intelligence and data mining, with applications to decision support and multiobjective optimization. Francesco is currently working on innovative solutions for the next-generation smart industries, ranging from industrial processes to occupational safety and health.

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.

Michela Dalle Mura is a Ph.D. student in industrial engineering (*mechanical engineering* curriculum) of the Leonardo da Vinci doctoral school of engineering of the University of Pisa, Pisa, Italy. She graduated with top marks in management engineering, and is now working at the Department of Civil and Industrial Engineering of the University of Pisa. Her research interests concern the development of innovative methods and software tools for enhancing the efficiency of assembly and disassembly lines, including the consideration of ergonomic aspects and the implementation of augmented reality applications.

Gino Dini is a Full Professor of production engineering within the Department of Civil and Industrial Engineering, University of Pisa, Pisa, Italy. His recent research interests concern automation of manufacturing processes and development of innovative tools for enhancing efficiency in assembly and disassembly sequences and plants, including implementation of augmented reality methods for supporting workers in performing assembly/disassembly tasks. He is an active member of CIRP (International Academy for Production Engineering). He has contributed as a co-author to more than 150 papers published in international journals and conferences and he was the coordinator in many national and international research projects.