Integrated automation system with PSO based scheduling for PCB remanufacturing plants

Alessandro Brusaferri[1], Egidio Leo[2], Leonardo Nicolosi[3], Danial Ramin[1], Stefano Spinelli[1]
[1]CNR - Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing, Milan, Italy
[2]Technische Universität Dortmund, Process Dynamics and Operations Group, Dortmund, Germany
[3]PARVIS systems and services s.p.a., Milan, Italy
{alessandro.brusaferri,danial.ramin,stefano.spinelli}@stiima.cnr.it, egidio.leo@tu-dortmund.de, leonardo.nicolosi@parvis.it

Abstract – **Flexibility and reconfigurability represent crucial challenges to be addressed to achieve sustainable remanufacturing processes, supporting efficient End-Of-Life treatment of heterogeneous mixes of post-use products. In this context, Printed Circuit Boards (PCB) represent a key target to be addressed. In this work, we propose an integrated automation system fostering the implementation of flexible PCB remanufacturing plants. In particular, we extend previous works by introducing on-line scheduling facilities. To this end, we developed a predictive-reactive algorithm by a Unified Particle Swarm Optimization form. The objective function combines the total weighted tardiness and the idle time of the rework machine to minimize energy consumption due to dissipations. We show the effectiveness of the proposed approach by application to the STIIMA-CNR De/Re-Manufacturing pilot plant, reporting improved performances compared to previously implemented scheduling rules.**

***Index Terms* — Automation, Scheduling, Particle Swarm Optimization, Remanufacturing plants.**

I. Introduction

Remanufacturing represents a widely recognized enabler for an efficient and systematic implementation of Circular Economy. However, major uncertainties and variability in the conditions of post-use products with respect to manufacturing systems pose several challenges. According to the recent review performed in [1], modular plants with reconfigurable automation solutions represent research priorities to foster enhanced adaptability and operational costs minimization, enabling development of sustainable remanufacturing processes. Moreover, the increased adoption of new consumer centered manufacturing paradigms, as mass customization, is posing new challenges to remanufacturing processes, which must be conceived to be flexible and adaptive, supporting cost effective and responsive treatment of continuously changing volumes and mix of post-use products. In addition, due the increased diffusion of intelligent mechatronic devices, the volume of mechatronics components to be remanufactured is increasing more and more. Since an important component of such products is the Printed Circuit Board (PCB), it represents a critical issue for remanufacturers and for the environment as well [2]. Indeed, PCBs include a significant content of precious materials (e.g. key-metals, rare earths) whose availability is decreasing. Besides, PCBs are a major source of Waste Electrical and Electronic Equipment (i.e. WEEE).

Present work addresses reported challenges by proposing an integrated automation system fostering the implementation of flexible PCB remanufacturing plants. Despite the wide research literature developed within manufacturing system community, such concepts have been only partially addressed in the remanufacturing field [1]. In particular, we complemented previous works (see e.g. [12][16]) targeting the IEC61499 based distributed control and the flexible routing strategy for PCB remanufacturing by introducing a further component within the overall automation architecture providing on-line scheduling facilities. Hence, we extended and compared previously implemented strategies within the plant, based on the Earliest Due Date first rule and the Shortest Processing Time first. Several requirements have to be properly considered for such purpose. First of all, the task list to be performed on each PCB is not defined a-priori, since it depends on the result of the post-use product test. Then, an on-line re-scheduling system is needed, computing the solution in due time while reacting to deviations occurred. Besides, different optimization criteria have to be tackled, including production related objectives (i.e. tardiness) and energy consumption minimization. We address the solution of the remanufacturing scheduling problem, representing a class of dynamic job shop scheduling problem (JSSP), which is NP-hard in the strong sense [3]. Over the last decades, several approaches have been proposed to deal with instances of JSSP. According to the exhaustive overview in [4], meta-heuristics represent valuable tools mainly due to their scalable problem-independent nature, limited parameters configuration effort and throughout problem exploration by global treatment. Indeed, despite the achievement of a global optimum is not guaranteed, the capability to find good sub-optimal solutions with minor computation time have been reported over a broad range of industrial applications (see e.g. [5]-[9]). Leveraging on such results, in this work we focused on Particle Swarm Optimization (PSO) to develop the solution algorithm in the scheduler [10]. In particular, we developed a predictive-reactive

Brusaferri, Alessandro, et al. "Integrated automation system with PSO based scheduling for PCB remanufacturing plants." 2019 IEEE 17th International Conference on Industrial Informatics (INDIN). Vol. 1. IEEE, 2019. DOI: 10.1109/INDIN41052.2019.8972293

scheduling method with regeneration based on a Unified PSO form, balancing the influence of local and global PSO. The scheduling engine has been deployed within the automation system by means of the interfaces provided by the IEC61499 based environment. We compared the proposed scheduling system to previously implemented scheduling rules on a real industrial scale pilot plant, reporting improved performances as well as the capability to easily manage reconfigurations of objective function criteria. The paper is organized as follows: Section II defines the requirements to be considered to develop the scheduling system for the PCB remanufacturing process. Then, the designed scheduling problem is detailed. Section III reports the deployed solution method, based on Unified PSO. Section IV reports the performed integration of the scheduling component within the overall automation system. Section V reports the application to the STIIMA-CNR De/Re-Manufacturing pilot plant and the results achieved.

II. REMANUFACTURING SCHEDULING FRAMEWORK

An integrated remanufacturing process must support three major production scenarios. The highest target is to give a second life to the post-use product. If not feasible, due to technical or economic reasons, sub-components re-use is considered. The final option is to recover the raw material, providing an alternative to End-Of-Life policies based on incineration and landfill [11], [2]. The technological processes start from a robotized station, aimed to disassemble the mechatronic product and load the electronic PCB on a flexible pallet, moved on the line by a modular conveyor (see e.g. [12] for details). If the PCB has an economic value justifying remanufacturing, it undergoes to the PCB analysis process for identifying eventual corrupted components or board damages. Positively tested PCBs exit the system as reusable products. If the analysis identifies an unrepairable failure on the board, the PCB is moved towards the material recovery cell. Otherwise, in case of failure on some sub-components, the PCB is sent to the rework station for substitution. Afterwards, a new PCB analysis is performed to verify the result of the rework operation. If the test is passed, the PCB is moved to the first station and exit the process, otherwise the loop is repeated (with a configurable number of iterations). As soon as the maximum number of iterations is reached, the PCB is sent to the material recovery process, likewise PCBs entering the plant with no remanufacturing feasibility. Before this stage, if the PCB mounts valuable or critical components, it is sent to the rework station for disassembly and recovery before shredding. An integrated remanufacturing plant can include from single up to several instances of each operating station depending on the production needs. Then, based on evolving requirements (e.g. input flow), the process layout might be reconfigured (e.g. by integrating a further station) thus requiring control and scheduling strategy to adapt accordingly. To address the reported requirements, several issues must be properly tackled within the scheduling system. First of all, compared to conventional manufacturing plants, the task list to be accomplished is not known a-priori before entering into production execution. In fact, the number of repair and consequent test loops depends on the specific results of rework operations on the PCBs. Commonly, production related constraints must be respected as well. In particular, since PCB remanufacturing plants have to deliver products by a certain deadline contracted with the customers, tardiness of the PCBs with respect to the due dates has to be considered. Product-specific priorities have to be managed as well, since related to orders from different customers. Moreover, as remarked within the introduction, minimization of energy consumption must be addressed. Considering PCB remanufacturing, the rework machine represents the foremost energy consumer of the process due to the high temperature required for component de-soldering and re-soldering. In particular, machine idle times between operations critically impact energy consumption due to heat dissipation. Therefore, the minimization of the idle time of the rework station must be considered within the strategy.

A. Remanufacturing scheduling problem formulation

Considering the requirements reported, we characterize the PCB remanufacturing scheduling problem as follows:

- a PCB represents a *job* and each PCB processing step constitute a *task*; operation classes include: robotic disassembly station (R), PCB test (T), rework station (RW) and PCB shredding and material recovery cell (W);
- jobs are processed by one machine at each time sample;
- each task is characterized by a processing time p_{jm}, (i.e. the processing time of the job j on the machine m), expressed as an integer number of seconds;
- each job is characterized by a priority factor w_j and a deadline d_j. Possible job scenarios are defined in Table I.
- job types 1-4 represent new PCBs incoming into the process, whereas types 5-8 PCBs exiting the testing machine. The testing cell defines the next tasks required by job type 5-8, according to the test results. Then, the sequence of machines visited is specified by precedence.
- each job must be processed on each machine once. There is no recirculation in the formalized problem since jobs entering in R, T, and W regenerate new jobs within the rolling-horizon problem formulation.
- machines set-up times are set to zero. Such assumption is reasonable due to the availability of a control layer responsible to start up the machine, detailed in section IV.
- pre-emption is not allowed since operations within the line cannot be stopped before completion.
- PCB transportation times are negligible compared to the processing time of the jobs in the machines.

The objective function has been defined as the weighted combination of the total weighted tardiness (TWT) and the idle time of the RW machine (IT):

$$F = \sigma \cdot TWT + (1 - \sigma) \cdot IT \quad (1)$$

with weight factor $\sigma \in \mathbb{R}$. Then, the scheduling objective can be easily reconfigured by acting on the weight factor coefficient, thus stating specific preferences over the two terms. Such facility was not supported by previously implemented scheduling rules. A further aspect to be considered within the scheduling system is the capability to react quickly to

unexpected events by revising schedules in a cost-effective manner. To such an aim, we deployed a predictive-reactive scheduling strategy by a hybrid policy. In particular, re-scheduling is performed periodically by checking for deviations from the previous schedule (e.g. depending on the result of the circuit test or rework operation) or occurrences of events (e.g. machine breakdown, high-priority job arrival). Besides, we adopted a regeneration approach considering, within the optimization problem, the entire set of tasks which have not been processed yet at the re-scheduling point [13]. Then, optimizations are executed until a configurable deadline (e.g. maximum time, number of iterations). Furthermore, in order to consider the plant behavior during the re-scheduling process, the optimization algorithm provides a schedule with initial time delayed by the optimization deadline. During the optimization process, the plant follows the schedule previously generated. Investigation of further approaches (e.g. considering robustness with respect to disruptions, bounded stability, etc.) are foreseen as future extensions of the present work.

TABLE I
POSSIBLE JOB SEQUENCES: JOB TYPES (JT) AND THE CORRESPONDING MACHINE SEQUENCE (MS)

JT	1	2	3	4	5	6	7	8
MS	R-T	R-RW-T	R-RW-T	R-W	R	RW-T	RW-W	W

III. UPSO-BASED SOLUTION METHOD

As introduced above, we deploy a scheduling solution method based on Particle Swarm Optimization, proceeding as follows. At each iteration k of the algorithm, particle positions $x_i(k)$ are adjusted according to their velocity $v_i(k)$, where $i = 1, \dots, P$ and P is the total number of particles. Positions $x_i = (x_{i1}, x_{i2} \dots x_{in})$ and velocities $v_i = (v_{i1}, v_{i2} \dots v_{in})$ are vectors of n items, where n is the problem dimension. The position of any particle of the swarm represents a possible solution, composed of a list of float numbers with length equal to the number of tasks to be scheduled (i.e. $taskNum$).

Afterwards, the solution at each iteration is evaluated on the fitness function, by performing the following operations:

1. adapt particle position x_i by sorting the vector of float numbers from the smallest to the highest value, as e.g.:

Original particle	1.45	2.34	0.23	-1.41	7.52
Adapted particle	3	4	2	1	5

   The new vector of integers represents the list of task IDs within 1 and *taskNum*. In particular, it indicates the priority by which the tasks will be considered for allocation in the scheduling time line;
2. allocate each task in the scheduling time line, by avoiding that multiple tasks are executed concurrently on the same machine while respecting precedencies of tasks belonging to the same job;
3. evaluate the fitness value of solution x_i, namely F_i.

During algorithm execution, velocities are iteratively adapted by considering three contributions:

- the particle velocity at the previous iteration $v_i(k)$, giving proper momentum to roam across the search space.
- the distance between the current position and the best position p_i ever visited by the particle, encouraging movement toward its own best position found so far.
- the distance between the current position and the best position p_g ever visited by either the whole swarm (namely global PSO - GPSO), or a sub-group of particles (namely local PSO - LPSO), introducing a collaborative effect of the particles in finding the global optimal position

The optimization algorithm proceeds until a user-defined stop criterion is reached, e.g. a maximum number of iterations or a maximum execution time. Once the optimization procedure is completed, the particle associated to the best fitness value is taken as the solution.

Different connections in the contributions reported above relate to specific PSO forms. In this work, we deployed the Unified Particle Swarm Optimization (UPSO) approach, balancing the influence of the LPSO and the GPSO on the position shifts by introducing the unification factor u [10]. Specifically, the UPSO has been formulated as follows:

$$vg_i(k+1) = \xi \cdot v_i(k) + c_1 R_1 (p_i - x_i(k)) + c_2 R_2 (p_{gg} - x_i(k)) \quad (2)$$

$$vl_i(k+1) = \xi \cdot v_i(k) + c_1 R_1 \left(p_i - x_i(k)\right) + c_2 R_2 \left(p_{gl} - x_i(k)\right) \quad (3)$$

where R_1 and R_2 are random numbers uniformly distributed within $[0,1]$, acceleration coefficients c_1 and c_2 can be either constant or variable throughout the optimization, and the constriction factor ξ controls the particle velocity preventing explosion and ensuring convergence [14]. $vg(vl)$ and $p_{gg}(p_{gl})$ represent respectively the velocity and the best position of the particle for GPSO (LPSO). The constriction factor has been calculated by the following expression with $\kappa \in (0,1)$:

$$\xi = \begin{cases} \frac{2\kappa}{\phi - 2 + \sqrt{\phi^2 - 4\phi}}, & for\ \phi = c_1 + c_2 > 4, \\ \kappa, & otherwise \end{cases} \quad (4)$$

Global and local contributions have been combined by:

$$v_i(k+1) = u \cdot vg_i(k+1) + (1-u) \cdot vl_i(k+1) \quad (5)$$

in which values of u approaching zero favor LPSO facilitating the exploration phase, whereas values of u close to 1 give priority to GPSO exploitation search. u can be either constant or variable throughout the optimization procedure. At each iteration, as soon as the new particle velocities v_i have been estimated, the particle positions are updated accordingly by:

$$x_i(k+1) = x_i(k) + v_i(k+1) \quad (6)$$

The new positions are evaluated by the objective function. In the monotonically decreasing case, the best particle positions can be updated by:

$$p_i(k+1) = \begin{cases} x_i(k+1), & if\ f(x_i(k+1)) < f(p_i(k)) \\ p_i(k), & otherwise \end{cases} \quad (7)$$

Finally, the global best position p_{gg} and p_{gl} are similarly adjusted as follows.

$$p_{gg}(k+1) = \begin{cases} x_i(k+1), & if\ f(x_i(k+1)) < f(p_{gg}(k)) \\ p_{gg}(k), & otherwise \end{cases} \quad (8)$$

$$p_{gl}(k+1) = \begin{cases} x_i(k+1), & if\ f(x_i(k+1)) < f(p_{gl}(k)) \\ p_{gl}(k), & otherwise \end{cases} \quad (9)$$

In the LPSO, particles have been connected in a ring network topology with a neighborhood radius equals to 2. Thus, any

sub-group is composed of five particles and each particle is allowed to exchange information with the previous and the next two neighbors. As it concerns the unification factor, in order to favor exploitation at a very early phase and increase exploration at the end of the optimization, it has been considered variable within 0 and 1 according to an exponential relation with the maximum execution time. The choice of the swarm size often depends on the version of PSO used, the number of variables, and the complexity of the search space. Generally, increasing the number of particles decreases the number of required algorithm iterations, and the success rate of the algorithm could be increased significantly since more particles sample the solution space. Nevertheless, more particles require more function evaluations. To this concern, some authors have considered a swarm size as twice up to five times the number of variables [14], whereas others have used a dynamic population size [15]. Since no methods outclasses the others in general, in this work we adopt a fixed swarm size. Furthermore, [14] suggested to limit particle velocity to the maximum value of positions in order to avoid swarm explosion. In our case, the maximum value of particle positions is represented by the total number of tasks ($taskNum$) to be scheduled. Hence, particle velocity is limited within $[-taskNum, taskNum]$. Besides, we adopted the set $\phi = c_1 + c_2 > 4$ suggested in [15] to prevent explosion and support convergence. We might remark that, for pure combinatorial problem, further discrete formulations can be found in literature, e.g. the DPSO. Nevertheless, we adopt the original formulation of the PSO since we plan to integrate further continuous variables in future extensions of present work, e.g. task processing time, power allocated to rework.

IV. Integrated Automation Architecture

The integrated automation system for the PCB remanufacturing process including the scheduling function, is depicted in Figure 1. The framework is composed of three major components connected by an Industrial Ethernet communication infrastructure. The first is the distributed control application including the remanufacturing stations and the flexible conveyor modules (see e.g. [12]) deployed within the NXTStudio environment, implemented on dedicated Industrial PCs with Linux OS. Each unit has been coded as a composite IEC61499 Function block type, exposing the supported automation tasks (e.g. PCB rework, etc.) as inputs and the station run-time state (e.g. idle, running task, task completed, failure, etc.) as outputs. Moreover, each unit exposes a LOAD_PCB (load board in the machine) event input linked to an input data PCB_ID. When called, the identification code of the pallet is passed to the machine by PCB_ID. This code is used by the control logic of the machine to upload from the production database the operation program to be performed on the specific PCB (i.e. reflow temperature curves, current/voltage levels for components testing, etc.). When the operation is completed, the operating unit function block generates an event output asking for board download to the conveyor. Besides, operating units include the control logic aimed at sending a switch-on event, depending on the scheduled operations and the time required to activate from the idle mode.

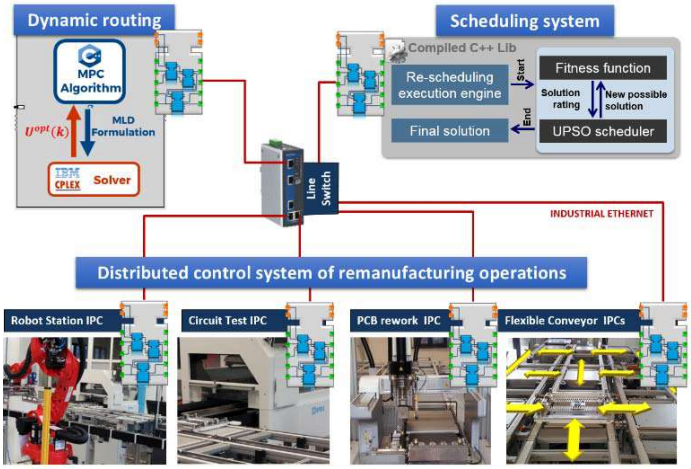


Fig. 1. Overall automation system

The second component of the automation system is dedicated to the dynamic optimization of PCBs routing throughout the flexible conveyor. In that respect, a receding horizon optimization approach based on Model Predictive Control have been deployed (see [16] for further details on the algorithm). The algorithm has been developed in C++ and encapsulated within an IEC61499 function block, by exploiting the interface provided by the NXT Control environment.

The scheduling system, representing the third component of the architecture, has been implemented in C++ and interfaced to NXTStudio by the same mechanism. The scheduling function starts from idle mode, waiting for incoming execution events. Once arrived, the algorithm checks the list of jobs/tasks to be processed (with related due date/priority) as well as the availability of operations. Once the solution is computed, an execution message (including PCB_ID data and timings) is sent to the dynamic routing module and to the distributed control system. Then, the scheduler returns to idle state, waiting for subsequent cyclic or event-related computation.

V. Experiments and Results

We compare the proposed scheduling approach with previously implemented strategies, based on the Earliest Due Date first (EDD) rule and the Shortest Processing Time first (SPT), within the STIIMA-CNR De/Re-Manufacturing pilot plant. As shown in Table III, tests have been performed on four different scenarios, targeting the current process layout with increasing problem dimension, where job, task lists, and job due dates were created randomly. The current layout includes a disassembly cell, a test cell, a rework station, and a material recovery cell, connected by the automatic conveyor.

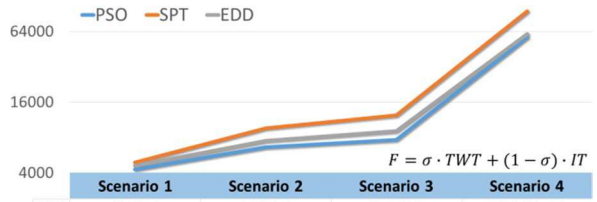
TABLE III
Testing Scenarios

Scenarios	1	2	3	4
Job number	9	15	23	49
Task number	20	32	51	96

To ensure comparison of the PSO-based strategy and the priority rules, the scheduling methods are tested over the same job shop conditions: the new jobs created by the test machine

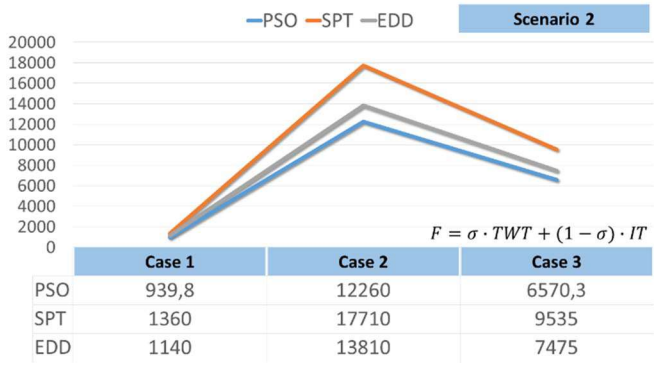
are deterministic but not known a priori to the scheduler. Each scenario has been designed taking into account real plant requirements and constraints, thus reflecting realistic operating situations. Considering the operating machines processing time and the computational cost, we adopted a re-scheduling period equal to 300 sec and a maximum optimization time of 120 sec. The weighting factor $\sigma = 0.5$, thus giving the same importance to TWT and IT. c_1 and c_2 are varied respectively from 3.9 to 0.11 and from 0.11 to 3.9, according to an exponential relation with the maximum optimization time, with $\phi = 4.01$ and $\xi = 1.1$.

Experiments demonstrate that the adoption of the proposed PSO-based strategy leads to a significant improvement of the system performances. Figure Fig. 2 reports a comparison of the overall performance index (i.e. fitness), obtained by the EDD rule, the SPT rule, and the PSO algorithm in all scenarios of Table III, by setting all job priorities equal to 1. Figure 3 shows the comparison of the performances obtained over scenario 2, where the experiments (a) and (b) are executed respectively considering job priorities equal to 1 (only periodic re-scheduling), and different job priorities (hybrid re-scheduling procedures).



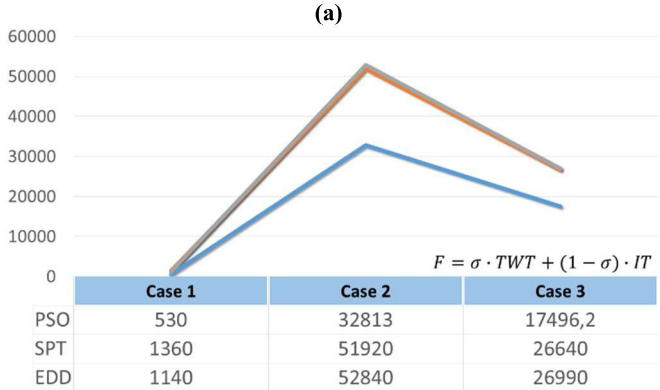
	Scenario 1	Scenario 2	Scenario 3	Scenario 4
PSO	4250,3	6570,3	7640,5	57010,48
SPT	4875	9535	12295	94367,8
EDD	4600	7475	9025	60903,8

Fig. 2. Comparison over scenarios with priorities = 1 and σ = 0.5.



	Case 1	Case 2	Case 3
PSO	939,8	12260	6570,3
SPT	1360	17710	9535
EDD	1140	13810	7475

(a)



	Case 1	Case 2	Case 3
PSO	530	32813	17496,2
SPT	1360	51920	26640
EDD	1140	52840	26990

(b)

Fig. 3. Comparison over scenario 2

In particular, cases 1 and 2 consider respectively the IT and the TWT as single objective function, whereas case 3 takes into account a linear combination of IT and TWT with $\sigma = 0.5$.

Moreover, Figure 4 and Figure 5 depicts the evolution of the tardiness of each job exiting the system for the scenario 2 and scenario 4 with the hybrid policy. As shown, the tardiness remains bounded during the overall rescheduling scenario even for the hybrid policy with frequent reschedules.

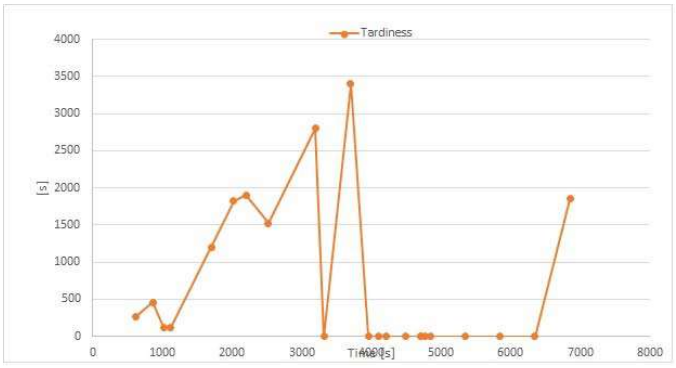


Fig. 4. Tardiness over scenario 2 by the UPSO with hybrid re-scheduling policy.

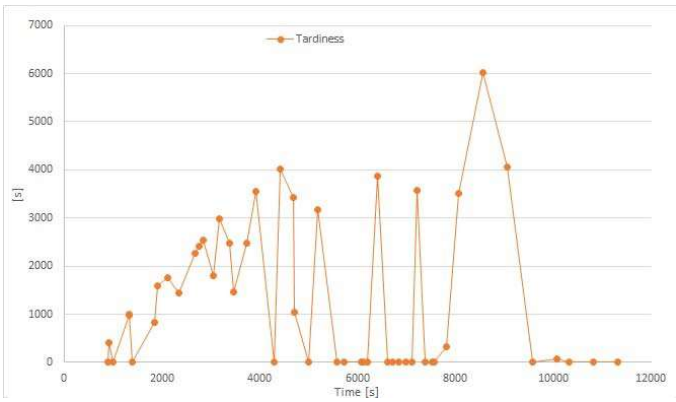


Fig. 5. Tardiness over scenario 4 by the UPSO with hybrid re-scheduling policy

Figure 7-8 show the amount of power and the cumulative energy provided to the Rework machine over testing scenarios 2 and 4 for recovering operating conditions (150°C) due to idle time dissipations. Figure 6 reports the machine temperature evolution during idle time from operating conditions.

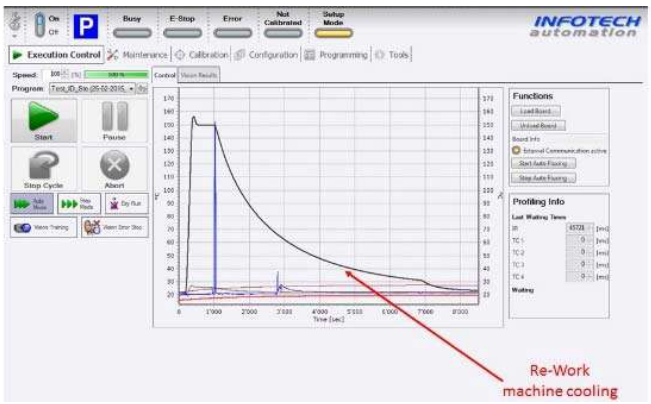


Fig. 6. RW temperature evolution from working temperature.

As shown in Figure 7, for the scenario 2, the cumulative energy consumption provided to the rework station results equal to 660 KJ for the proposed PSO-based optimization and to 1180 KJ for the SPT and the EDD rules. At the same time, the proposed

PSO-based method achieves an improved plant tardiness compared to the priority rules. On scenario 4, the cumulative energy consumption are respectively 480 KJ for the PSO-based method, 1700 KJ for the EDD rule and 2200 KJ for the SPT rule. As reported above, the PSO based scheduler outperforms the dispatching rules on the tardiness objective as well.

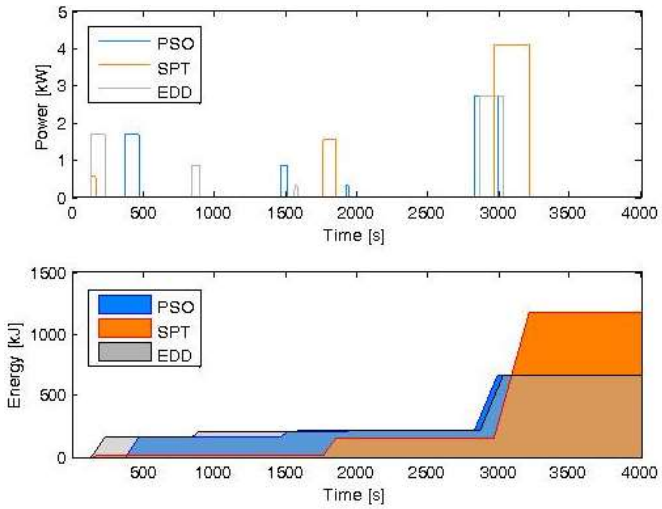


Fig. 7. Power and energy provided to the RW machine over scenario 2.

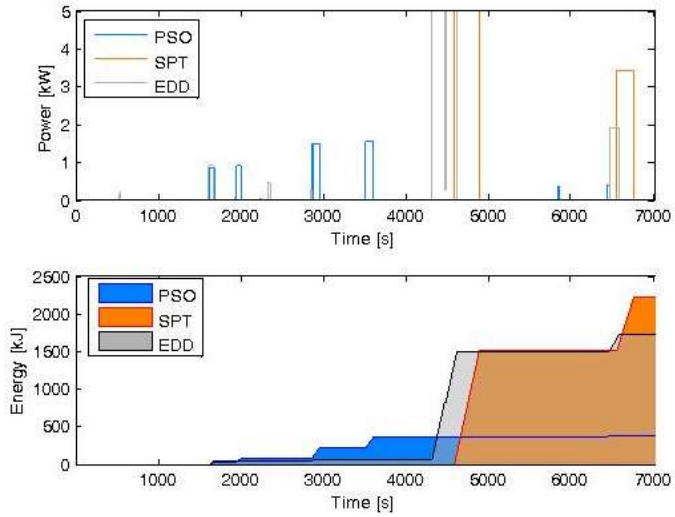


Fig. 8. Power and energy provided to the RW machine for scenario 4.

VI. Conclusions

This work investigated an integrated automation system for PCB remanufacturing processes, extending previous studies by addressing the scheduling component. For this purpose, an UPSO based algorithm has been designed, considering the plant conditions while performing minimization of the total weighted tardiness of the PCBs and the idle time of rework station, in order to reduce energy consumption. The implemented system has been tested on the STIIMA-CNR De/Re-Manufacturing pilot plant, covering different production scenarios and compared to previously adopted approaches (based on EDD and SPT rules). Results display the capability of developed method to outperform previously adopted priority rules over different configurations of the objective function weight. This study represents a first step towards the full exploration of enhanced automation and scheduling systems fostering sustainable remanufacturing. Several future extensions are foreseen: we plan to investigate alternative optimization approaches, including techniques to reduce computation time and to support scheduling stability, as well as formal uncertainty characterization. Moreover, we foresee the application to further remanufacturing processes covering different application fields.

Acknowledgement

The present work has been supported by the DAEDALUS project, funded by the EUs Horizon 2020 research and innovation program under grant agreement N 723248.

References

[1] T. Tolio, A. Bernard, M. Colledani, S. Kara, G. Seliger, J. Duflou, O. Battaïa, S. Takata (2017). Design, management and control of demanufacturing and remanufacturing systems. CIRP Annals - Manufacturing Technology. 66. 10.1016/j.cirp.2017.05.001.

[2] G. Copani, A. Brusaferri, M. Colledani, N. Pedrocchi, M. Sacco, and T. Tolio, "Integrated de-manufacturing systems as new approach to end-of-life management of mechatronic devices", Proc. of 10th Global Conf. on Sustainable Manufacturing, Istanbul, Turkey, pp. 332 – 339, 2012.

[3] V. Sels, M. Vanhoucke, and N. Gheysen, "A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions", International Journal of Production Research, vol. 50, no. 15, pp. 4255–4270, 2011.

[4] A. Arisha, P. Young, and M. El Baradie, "Job shop scheduling problem: an overview", Int. Conf. for Flexible Automation and Intelligent Manufacturing (FAIM), Dublin, Ireland, pp. 682–693, 2001.

[5] M. Karimi-Nasab, M. Modarres, and S.M. Seyedhoseini, "A self-adaptive PSO for joint lot sizing and job shop scheduling with compressible process times", Applied Soft Computing, vol. 27, pp. 137–147, 2015.

[6] S. Kitamura, K. Mori, S. Shindo, and Y. Izui, "Modified multi-objective particle swarm optimization method and its application to energy management system for factories", Electrical Engineering in Japan, Wiley Periodicals Inc., vol. 156, no. 4, pp. 33–42, 2006.

[7] M. Li, B. Wu, Y. Hu, C. Jin, and T. Shi, "A hybrid assembly sequence planning approach based on discrete particle swarm optimization and evolutionary direction operation", Int. J. Adv. Manuf. Technol, vol. 68, pp. 617–630, 2013.

[8] J.J. Kim and J.J. Lee, "Trajectory Optimization With Particle Swarm Optimization for Manipulator Motion Planning", IEEE Transactions on Industrial Informatics, vol. 11, no. 3, pp. 620–631, 2015.

[9] J. Mercieca and S.G. Fabri, "Particle swarm optimization for nonlinear model predictive control", in Proc. Of the 5th Int. Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP), pp. 88–93, 2011.

[10] K.E. Parsopoulos and M.N. Vrahatis, "UPSO: A unified particle swarm optimization scheme", Lecture Series on Computer and Computational Sciences, Vol. 1, Proc. of the International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2004), VSP International Science Publishers, Zeist, pp. 868–873, 2004.

[11] J.R. Duflou, G. Seliger, S. Kara, Y. Umeda, A. Ometto, and B. Willems, "Efficiency and feasibility of product disassembly: A case-based study", CIRP Annals, 2009.

[12] S. Spinelli, A. Cataldo, G. Pallucca and A. Brusaferri, "A distributed control architecture for a reconfigurable manufacturing plant," 2018 IEEE Industrial Cyber-Physical Systems (ICPS), St. Petersburg, 2018, pp. 673-678.doi: 10.1109/ICPHYS.2018.8390788

[13] G.E. Vieira, J.W. Herrmann, and E. Lin, "Rescheduling manufacturing systems: a framework of strategies, policies, and methods", Journal of Scheduling, vol. 6, pp. 39–62, 2003.

[14] I.C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection", Information Processing Letters, vol. 85, pp. 317–325, 2003.

[15] W.F. Leong and G.G. Yen, "PSO-based multi-objective optimization with dynamic population size and adaptive local archives", IEEE Transactions on systems, Man and Cybernetics – B,vol. 38, no. 5, pp. 1270–1293, 2008.

[16] A. Cataldo and R. Scattolini, "Dynamic Pallet Routing in a Manufacturing Transport Line With Model Predictive Control," in IEEE Transactions on Control Systems Technology, vol. 24, no. 5, pp. 1812-1819, Sept. 2016.