A Unified PSO-based method for multi-hoist scheduling in advanced Galvanic plants

Danial Ramin[1], Egidio Leo[2], Leonardo Nicolosi[3], Stefano Spinelli[1] and Alessandro Brusaferri[1]

***Abstract*— In this work, we propose a method based on the unified particle swarm optimization (UPSO) for no-wait multi-hoist scheduling, including a collision avoidance heuristic. Conflicts due to track sharing between hoists and no-wait constraints represent major issues to be addressed. Consequently, a complex optimization problem has to be solved dynamically, to identify the best operating strategy to be executed depending on the characteristics of the current job list. A decomposition procedure has been developed to speed up the solution of the large-scale optimization problem at hand. The proposed approach is demonstrated on a real galvanic process layout, showing the improved performances achieved by the proposed heuristic compared to the monolithic approach.**

***Index Terms*—Particle Swarm Optimization, No-wait job shop scheduling, Multi-hoist scheduling, Advanced Galvanic plants**

I. Context and objectives

In manufacturing facilities, the movement and handling of process material or equipment across the plant is usually carried out by track-mounted hoists (or robots). In this work, we consider a standard configuration, which consists of one or more hoists moving along a single horizontal track mounted on the ceiling. The production schedule for the plant, in general, assumes hoists to be available to perform the movement of the material form one processing unit to another at the desired times, in order to not compromise the defined schedule. However, as the hoists operate on a single track, they must be scheduled to avoid colliding with each other. Thus production schedule and hoist schedule problems are strictly connected. However, because of the complexity of the monolithic problem, these schedules are usually planned separately, considering the production decisions as the input of the hoist scheduling. This simplification could generate infeasibilities in the hoist schedule, as the production schedule is defined without modeling effective hoist availability.

Widely known industrial application where the simultaneous schedule of jobs and hoists assumes a central relevance is the galvanic plants for surface treatments. In these processes, part lots must follow a specific sequence of chemical and water baths, with strict exposure times and demanding storage policies. Hence, highly automated production lines, where material handling is executed by computer-controlled hoists or robots, have become common practice in modern advanced manufacturing systems. Consequently, a strong need for enhanced hoist scheduling systems emerged, representing a key enabler to maximize productivity and product quality.

The first seminal study in this field was developed in the early seventies by [1]. Subsequently, several mathematical models and meta-heuristic algorithms have been investigated (see e.g., review in [2]). Most of them deal with the cyclic case, whereas a minor number deal with advanced multi-hoists lines. The schedule of multiple hoists on a single track is an extremely complex problem to be solved. A major number of decisions have to be optimized and collision among hoists has to be taken into account. Despite the complexity of the problem, little research work has been performed in this area. In [3] a first mixed-integer programming (MIP) formulation is proposed to find an optimal cyclic hoist schedule for the single-track multi-hoist system with unidirectional part flow, with the strong assumption that the part processing sequence should be the same as the tank arrangement. This work was improved by [4], proposing the first MIP approach considering a bidirectional cyclic part flow, and removing the assumption of identical the part processing sequence and tank arrangement. [2] developed a MIP model for a cyclic jobshop single hoist scheduling problem with multi-capacity reentrant tanks and time-window constraints. In a hybrid solution along with a metaheuristic approach [5] proposed a mixed-integer model for the cyclic scheduling of multiple hoists considering collision. However they found the MIP to be computationally expensive when dealing with large-scale problems.

Most of the previous works on the hoist scheduling problem are aimed at identifying an optimal cyclic schedule, maximizing the number of parts produced in each cycle. The major limitation of the cyclic hoist scheduling is the assumption of identical jobs in the system. Besides, the definition of a cycle is rather vague when there are multiple job types and multiple robots. Although the scheduling of multi-item production lines is of high practical relevance, it is rarely dealt with in the literature. [6] studied the two-hoist scheduling problem, pre-assigning to each hoist a precise set of tanks. Adopting this simplification, the collision constraints are not needed and the problem becomes more tractable. However, this approach restricts the sequence of treatment of the parts to follow the layout of the tanks. Hence, the authors consider alternative partition problems. Adopting a less restrictive simplification, [7] studied a layout where the assigned zones can overlap. The algorithm requires the identification of all the collision configurations in the

[1] Danial Ramin, Stefano Spinelli and Alessandro Brusaferri are with the Institute of Intelligent Industrial Systems and Technologies for Advanced Manufacturing (STIIMA), National Research Council (CNR), Milan, Italy, {name.surname}@stiima.cnr.it

[2] Egidio Leo is with the Technische Universitat Dortmund, Process Dynamics and Operations Group, Dortmund, Germany, egidio.leo@tu-dortmund.de

[3] Leonardo Nicolosi is with the PARVIS systems and services s.p.a., Milan, Italy, leonardo.nicolosi@parvis.it

shared common zones and gives priority to one of the hoists. To the best of our knowledge, integrated methods addressing non-cyclic scheduling of multi-hoist, multi-product galvanic processes including collisions management, are still lacking in the literature.

In this paper, we propose a Unified Particle Swarm Optimization (UPSO)-based method for the multi-hoist, multi-stage batch Galvanic process, taking into account robot collisions explicitly. Besides, we developed a decomposition strategy to tackle the intractable solution of the monolithic scheduling problem while addressing applications of realistic size. The paper is organized as follows: Section 2 details the characteristic of the no-wait multi-hoist scheduling problem. Section 3 presents the proposed approach, including the UPSO-based algorithm, the collision management procedure, and the decomposition strategy. Section 4 provides quantitative results from an industrial use case.

II. Scheduling multi-hoists in Galvanic plants

Galvanic plants represent a key component of the manufacturing chain in various application fields, from consumer goods to the aerospace industry. A major challenge of galvanic plants resides in the management of transportation system aimed at moving processed materials through different baths, depending on the specific product needs. Indeed, several specific requirements and constraints have to be properly tackled to guarantee production quality, maximize productivity and minimize operating costs.

In advanced galvanic plants, the hoists, or robots, transport the lots from one tank to another for the next operation. The transfer of parts can be multi-directional. A critical process constraint is that each hoist can only move a single part at a time without holding the job more than the exact transfer time. No intermediate storage is present between consecutive baths, thus when the part finishes its processing time, a hoist must be available to transfer it to the next processing bath. Therefore a non-intermediate storage (NIS) policy is assumed. An additional constraint, which increases the complexity of the problem, is that the baths must process part lots singularly, with a fixed and strict duration, avoiding the overexposure, which can seriously damage the part-lot. These process constraints are known as "zero-wait" (ZW) policy. Because of these constraints, usually two or more hoists are in place in order to improve plant productivity.

It is evident that the optimal utilization of this critical shared resource, i.e., the multi-hoist system, can considerably improve the productivity of these plants, which largely depends on the schedule of the hoist activities, leading to an important reduction in the total processing time. The typical objective of this kind of plant is to find the best production and transportation schedule that minimize the total completion time of all jobs in the system, widely known as the Makespan criterion (MK). Another common objective function is the total weighted tardiness if job deadlines are provided. In this work, we assume that N lots have to be processed following predefined recipes, which indicate the sequence of baths (or units) to be visited, which is common in most industrial setups. Besides, the problem considers that two hoists (or robots) are installed on the same track. Hoists share the track and therefore collisions must be avoided. The hoist schedule assumes a central relevance: the problem concerns the scheduling of N jobs in M units, in a job shop multi-product plant with ZW and NIS policies with shared resources and limited capacity for the job movement. The proposed model can be easily extended to consider multiple robots and different recipes for each part lot. From an optimization point of view, the problem under investigation represents a class of large-scale job shop scheduling problem (JSSP), which is NP-hard in the strong sense.

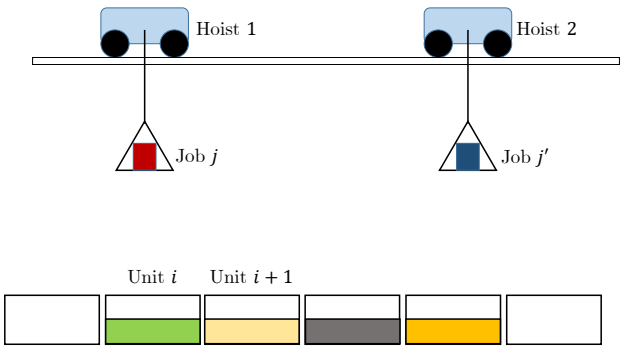

Fig. 1. Galvanic plant schema

A. Case study: advanced galvanic plant for metal components

To validate the proposed scheduling method, an advanced automatic industrial process for surface-treatment of metal components is presented. The name of the company is not reported due to confidentiality reasons. The galvanic plant, Figure 1, is composed of two lines of chemical baths with 39 baths per line, where the jobs have to be scheduled following their own production recipes. To move the jobs from one bath to another, two hoists for each line are mounted on the ceiling. Since each couple of hoists operates on a single track they could collide or interfere with each other. The job/task list employed for testing has been identified from real production scenarios executed on the plant. The specific details regarding processing times of each task and the sequence of baths followed by each job recipe are reported in Table I. For the sake of readability the data related to the first two jobs is provided. Different baths (10-11, 15-16, 24-25-26-27) can be used as parallel units. The input/output buffer is defined by the baths 79-74: each job starts and finishes its production recipe in one of these baths. The transfer of jobs from one line to the other one is performed by two transfer hoists: the first one transfers the jobs from bath 72 to 1; the second one from bath 31 to 39. Transport time refers to the time required to reach the next unit. A lower bound and an upper bound are provided for each hoist transport task (see I). In addition, recycle flows can occur.

TABLE I

TASK LIST

IDjob	Unit	Transport Time [s]	Min Processing Time [s]	Max Processing Time [s]
1	1	25	10	10
1	3	29	300	330
1	8	23	2	4
1	9	26	10	20
1	12	23	300	330
1	13	32	20	40
1	6	23	60	66
1	7	23	120	132
1	8	23	2	4
1	9	25	10	20
1	11	25	300	330
1	9	38	10	20
1	20	23	2	4
1	21	26	300	330
1	24	29	300	330
1	29	23	4	8
1	30	36	4	8
1	20	29	2	4
1	15	28	2400	2640
1	19	39	2	4
1	31	33	10	10
1	39	26	10	10
1	42	23	210	231
1	41	32	6	12
1	48	49	4	4
1	67	32	3	6
1	74	128	1	1
2	1	23	10	10
2	2	28	300	330
2	5	25	60	66
2	7	26	5	5
2	8	29	2	4
2	9	29	10	20
2	10	30	60	66
2	14	25	5	10
2	15	28	2401	2641
2	18	27	2	2
2	20	28	2	4
2	23	25	2430	2673
2	24	30	1830	2013
2	28	30	240	264
2	29	29	4	8
2	30	31	4	8
2	31	33	10	10
2	39	28	10	10
2	43	27	66	73
2	45	57	15	30
2	64	28	10	11
2	63	30	353	388
2	64	28	10	11
2	65	28	4	8
2	66	28	4	8
2	67	28	3	6
2	68	28	20	22
2	69	28	1	1
2	70	28	120	132
2	74	128	1	1

III. HOISTS SCHEDULING METHOD

A. Unified Particle Swarm Optiomization approach

In this work, we exploited Particle Swarm Optimization (PSO) as the backbone optimization framework. The PSO represents a well-established meta-heuristic approach, due to its ease of implementation, efficiency, and robustness [8]. Originally conceived for optimization with continuous variables, the PSO has been recently extended and adapted also to deal with binary, discrete, and mixed-integer variable problems [9], [10]. Widespread adoption of PSO can be found in industry for the solution of complex optimization problems thanks to its speed in finding a global optimum, little effort required for parameter configuration and reduced risk to fall into a local minimum. Complex production scheduling problems [11], energy-driven factory optimization [11], assembly sequence planning and model predictive control [12] are just some examples of the wide range of applications available. The problem at hand represents a purely combinatorial problem, for which several discrete optimization procedures can be found in the literature. Nevertheless, the discrete version of the PSO has been considered in this work, since planned developments will involve also the optimization of continuous variables, e.g., task processing time and additional energy-related aspects.

The PSO is a population-based algorithm, originally inspired by the swarming behavior of bird flocks searching for food, which allows to probe the whole search space concurrently. The population, i.e. the swarm, is formed by P particles, i.e. the potential solutions of the optimization problem [13], each one is characterized by a position x_i in the optimization space and a velocity v_i. The positions and velocities are vectors, i.e. $x_i\, v_i \in \ltimes$, where n is the problem dimension. At each iteration k of the algorithm, each particle position $x_i(k)$ is adjusted according to its velocity $v_i(k)$; subsequently, also the velocity of each particle is updated, for the next iteration, by considering three contributions [14]:

- The particle velocity at the current iteration $v_{ij}(k)$;
- The distance between the current position and the best position p_{ij} ever visited by the particle;
- The distance between the current position and the best position ever visited by either the whole swarm (p_{gg}, namely global PSO) or a sub-group of particles (p_{gl}, namely local PSO).

Specific combinations of such contributions define a particular PSO algorithm. The original PSO model is described by (1). In this work, the so-called Unified Particle Swarm Optimization (UPSO) method is adopted, since it allows a balanced influence of the local PSO (LPSO) and the global PSO (GPSO) on the position shifts, through a convex combination determined by the unification factor u. The UPSO is mathematically defined by (1) to (3), where R_1 and R_2 are random numbers uniformly distributed within [0,1], c_1 and c_2 are called acceleration coefficients and can be either constant or variable throughout the optimization. The factor ξ, defined in (4), is the so-called constriction factor that allows controlling particle velocity preventing explosion and ensuring convergence. In particular, in (1) v_g and p_{gg} (respectively, v_l and p_{gl} in (2)) represent the velocity and the best position of the particle in the case of GPSO (LPSO, respectively). The two contributions are opportunely weighted by the combination in (3): values of u approaching zero favor LPSO facilitating the exploration phase, whereas values of u close to 1 give priority to GPSO and, therefore, to exploitation search. The unification factor can be kept constant or it can be varied during the optimization procedure.

$$v_{g_{ij}}(k+1) = \xi v_{ij}(k) + c_1 R_1 (p_{ij} - x_{ij}(k)) + c_2 R_2 (p_{gg} - x_{ij}(k)) \quad (1)$$

$$v_{l_{ij}}(k+1) = \xi v_{ij}(k) + c_1 R_1 (p_{ij} - x_{ij}(k)) + c_2 R_2 (p_{gl} - x_{ij}(k)) \quad (2)$$

$$v_{ij}(k+1) = u v_{g_{ij}}(k+1) + (1-u)(v_{l_{ij}}(k+1)) \quad (3)$$

The particle positions can be updated accordingly by:

$$x_{ij}(k+1) = x_{ij}(k) + v_{ij}(k+1) \quad (4)$$

The new positions are evaluated by a problem specific objective function. The best particle positions are updated

Algorithm 1 The PSO algorithm

procedure PSO
for each particle $i = 1, \ldots, P$ **do**
    Set the initial position of the particle $x_i(0)$ with a uniformly distributed random vector
    Set the initial best known position of the particle to its initial position: $p_i(0) \longleftarrow x_i(0)$
  if $f(p_i) < f(p_{gg})$ (respectively, $f(p_i) < f(p_{gl})$) **then**
    Update the best known position of the swarm globally (locally): $p_{gg} \longleftarrow p_i$ (respectively, $p_{gl} \longleftarrow p_i$)
  Initialize the velocity of the particle
while a termination criterion is not met **do**:
  for each particle $i = 1, \ldots, P$ **do**
    for each dimension $j = 1, \ldots, n$ **do**
      Pick random numbers: $R_1, R_2 \in U(0, 1)$
      Update the velocity of the particle as in (1)-(3)
      Update the particle position as in (4)
    if $f(x_i) < f(p_i)$ **then**
      Update the particle best-known position: $p_i \longleftarrow x_i$
    if $f(p_i) < f(p_{gl})$ **then** (respectively, **if** $f(p_i) < f(p_{gg})$ **then**)
      Update the best known position of the swarm globally (locally): $p_{gl} \longleftarrow p_i$ (respectively, $p_{gg} \longleftarrow p_i$)

by:

$$p_i(k+1) = \begin{cases} x_i(k+1) & \text{if } f(x_i(k+1)) < f(p_i(k)) \\ p_i(k) & \text{otherwise} \end{cases} \quad (5)$$

Finally, the global best position are similarly adjusted as follows:

$$p_{gg}(k+1) = \begin{cases} x_i(k+1) & \text{if } f(x_i(k+1)) < f(p_{gg}(k)) \\ p_{gg}(k) & \text{otherwise} \end{cases} \quad (6)$$

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

The optimization algorithm, described in Algorithm 1, terminates when a user-defined stop criterion is reached, which is typically the maximum number of iterations or the maximum execution time.

B. Developed Fitness function

As any meta-heuristic algorithm, the PSO uses a fitness function to quantify the optimality of a particle (sequence). The fitness function, incorporating the optimization constraints, is also responsible for the definition of the feasible region and therefore it has to determine whether a solution is feasible or not. When a proposed solution violates the problem constraints, the common practice is to assign an infinite value to the fitness function. However, this approach is not reliable when dealing with a large-scale optimization problem. Indeed, the algorithm might terminate exceeding the maximum execution time without finding a feasible solution. In the proposed work, this limitation has been overcome by developing a procedure that modifies the infeasible solution (particles' positions) until it becomes feasible, thus making its fitness function evaluable. The proposed algorithm is described by the pseudo-code in Algorithm 2. The developed fitness function takes implicitly the hoist collisions into account. During each iteration, the scheduling feasibility check is performed by analyzing the eventual presence of hoist conflicts. In the case of hoist conflicts in the current iterate solution, starting times and temporal intervals are shifted to achieve feasibility within the current run. Then, by iterations, the algorithm converges to a feasible local minimum.

Algorithm 2 Fitness Function

procedure FITNESS FUNCTION PROCEDURE
  for each task
    check for collision with algorithm 3
    If no collision
      compute the tardiness of the solution
    If collision
      while (solution not feasible)
        modifiy solution with algorithm 3

C. Hoists collisions avoidance

Each particle describes a solution to the optimization problem. A solution is characterized by:

- Start time of each task of each job in the units defined by the recipe;
- End time of each task of each job in the units defined by the recipe;
- Unit ID (if there are parallel units) to process each task of each job;
- Hoist ID (if there are more than two hoists) to perform each transport task;
- Start time of the transport task;
- End time of the transport task.

It is worth highlighting that each transport task is composed of an empty movement, performed by the hoist to reach the unit where the job is processed, and a loaded movement, to transport the job. The proposed algorithm,

comparing all the transport tasks, detects the hoists' collisions, and modifies the solution to avoid conflicts.

Our aim is to include a slight modification to the scheduling problem, in order to minimize the impact on the computational burden and to maintain a simple treatment. To this end, for each transport task a temporal/spatial interval with start and end time/unit of the task as extreme points is included. Comparing each transport task to all the others, a collision is detected if the intersection between the temporal intervals and the spatial intervals is not an empty set. In the case of the detection of a collision, the algorithm modifies the start and end times of the task under investigation.

The modification is performed delaying the starting time by the end of the task responsible for the collision. Since we are dealing with NIS and ZW constraints, all the tasks of the job responsible for the collision are delayed by the same quantity. Since this modification might violate other constraints, the algorithm checks again the feasibility of the solution. If no constraint violations are found, the algorithm terminates providing a feasible solution, otherwise, the procedure is repeated until a feasible solution is provided. The major steps of the procedure are reported in pseudo-code by Algorithm 3.

Algorithm 3 Collision Procedure

```
1: procedure COLLISION PROCEDURE for each job
2:         Compose temporal/spatial intervals for the tasklist;
3:     for each interval
4:         Evaluate intersection of temporal/spatial intervals
5:         If Collision number != 0
6:             delay starting time and update time intervals
7:             go to 4: check intersection
```

After each modification, the main algorithm evaluates the process constraints and if the solution is feasible, the algorithm proceeds to calculate the consequent job tardiness.

D. Developed solution strategy

The integration of the hoist scheduling and job scheduling leads to a large-scale optimization problem, characterized by high computation time. Even though meta-heuristics optimization approaches are generally capable of finding good quality solutions for such large-scale problems faster than other techniques, the computation time required to terminate the PSO algorithm, for this instance, is greater than 2 hours. To speed up the solution phase, instead of solving an intractable monolithic problem, a decomposed approach, generating smaller subproblems that can be easily solved, is proposed. The original problem is split into subproblems partitioning the original job list into smaller ones. These smaller subproblems are solved in a sequential way, taking into account the state of the plant (tasks pre-assigned to hoists and machines) defined by previously performed optimizations. Since the objective of the original problem is to minimize the tardiness of the jobs, the subproblems are created exploiting one of the most common priority rules: the earliest due date. The jobs are presorted sequentially in order to classify them into ranks according to their due date. The job list of each subproblem is generated by selecting five jobs per time. By adopting this strategy, the first optimized subproblem is composed of the jobs with the earliest due date. Then, the initial position of the PSO particles is initialized to the solution obtained in the previous subproblem optimization.

IV. Computational results

In this section, we report the results obtained by the application of the proposed method to a real production scenario of the Galvanic process. The job-list includes 55 jobs, covering a daily production plan. Computations have been performed on a PC with 8 Core (with parallel processing in 8 threads) 2.5 GHz Processor with 8GB of RAM. Due to the lack of a baseline method addressing the same problem class within the research field, we compared the decomposition approach to the conventional monolithic solution, to investigate the contribution of such a technique. As shown in Table II, the proposed decomposition solution outperforms the monolithic model. Indeed, a better (i.e., lower fitness function) solution is found in less computational time. Even if global optimality is not guaranteed (a characteristic of meta-heuristic approaches in general) the algorithm converged to a feasible local optimum in a computational time compatible with the process requirements. Figures 2-3 depict the related fitness value over the number of iterations. It is worth noting that the increasing envelop of the fitness curves, shown in Figure 3, is due to solution in sequence of the decomposed sub-problems. Each converging curve represents the solution of the related sub-problem. Then, the succeeding sub-problem is solved by adding to its fitness function the converged cost of the previous subproblems.

The resulting schedules for the hoists obtained by the decomposition algorithm, considering the conflicts on the shared track, is reported in Figure 4. Due to the large number of tasks, for the sake of readability only the first two hours of the scheduling horizon is shown here. The width of each block (with arbitrary height) represents the transfer time for

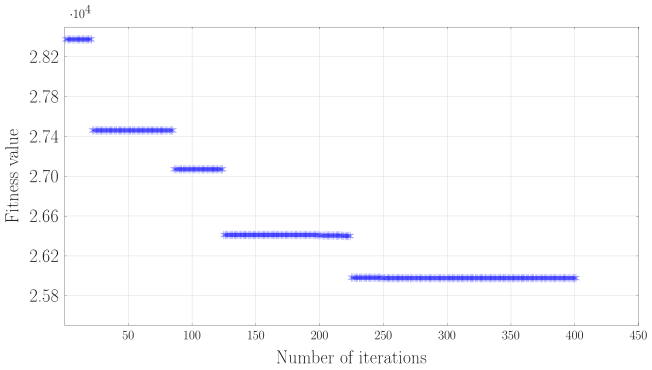

Fig. 2. Fitness function of the monolithic problem

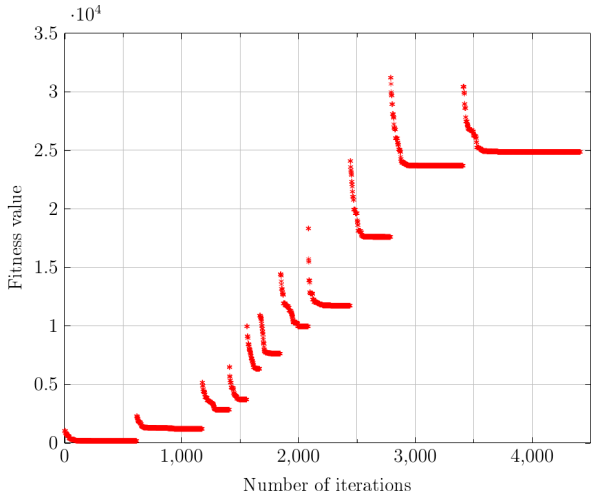

Fig. 3. Accumulative fitness function of the decomposed sub-problems

TABLE II
COMPUTATIONAL RESULTS

	Monolithic problem	Decomposed problem
CPU time [s]	> 6000	1800.9
Fitness function value	2.6e4	2.5e4
Number of iterations	400	4411

various stages of the same job (distinguished by different shades of the same color) as handled by the assigned hoist (shown in the corresponding panel). Here the hoist pairs (1,2) and (3,4) each share the same track.

V. CONCLUSION

In this paper, a non-cyclic scheduling approach for multi-hoist multi-product Galvanic plants with no-wait constraints is proposed. To this end, we adopted a Unified PSO approach. The main target is the inclusion of hoists collision avoidance within the optimization problem as multiple hoists are mounted on the same track in most Galvanic processes. To this end, we embedded a dedicated heuristic into the fitness function computation loop, post-processing the solution found by the UPSO during each iteration to achieve a feasible allocation of the hoists, thus avoiding collisions.

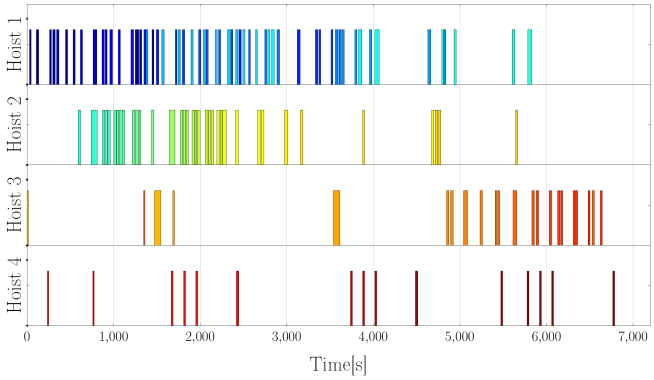

Fig. 4. Hoist schedule

Due to the extensive computation time required to solve the overall problem by a monolithic approach, we developed a dedicated decomposition strategy. By application to a real production scenario from a Galvanic plant, we have shown the capability of the proposed method to achieve feasible solutions. Besides, the decomposition approach provides a strong reduction of the optimization time, which is fundamental to enable the exploitation of the proposed scheduling method within industrial practice. This is particularly crucial for the implementation of reactive schedulers, required to compensate for unforeseen events that could occur during production execution.

Next developments will include the exploration of other decomposition strategies, the integration of tasks processing time variables and baths/hoists energy-related constraints/objectives; and the application of this framework to other use cases.

REFERENCES

[1] L. Phillips and P. Unger, "Mathematical programming solution of a hoist scheduling program," *IIETrans*, vol. 8, no. 2, p. 219–225, 1976.
[2] J. Feng, C. Chu, and A. Che, "Cyclic jobshop hoist scheduling with multi-capacity reentrant tanks and time-window constraints," *Computers and Industrial Engineering*, vol. 120, pp. 382 – 391, 2018.
[3] J. Leung and G. Zhang, "Optimal cyclic scheduling for printed circuit board production lines with multiple hoists and general processing sequence," *IEEE Trans. Robot. Autom.*, vol. 19, pp. 480–484, Jun 2003.
[4] Che and Chu, "Single-track multi-hoist scheduling problem: A collision-free resolution based on a branch-and-bound approach," *International Journal of Production Research*, vol. 42, 2004.
[5] E. Laajili, S. Lamrous, M.-A. Manier, and J.-M. Nicod, "Collision-free based model for the cyclic multi-hoist scheduling problem," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 873–878, IEEE, 2019.
[6] L.Lei and T.Wang, "The minimum common cycle algorithm for cycle scheduling of two material handling hoists with time window constraints," *Manage. Sci.*, vol. 12, no. 37, p. 1629–1639, 1991.
[7] M. Manier and S.Lamrous, "An evolutionary approach for the design and scheduling of electroplating facilities," *J Math Model Algor*, vol. 7, p. 197–215, 2008.
[8] K. Parsopoulos and M. Vrahatis, "Upso: A unified particle swarm optimization scheme," in *International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2004)*, p. 868–873.
[9] A. M. S.Chowdhury, W. Tong and J.Zhang, "A mixed-discrete particle swarm optimization algorithm with explicit diversity-preservation," *Structural and Multidisciplinary Optimization*, p. 367–388, 2013.
[10] S. S. A.Pal and K.Deep, "Use of particle swarm optimization algorithm for solving integer and mixed integer optimization problems," *International Journal of Computing Science and Communication Technologies*, vol. 4, no. 1, p. 663–667, 2011.
[11] L. Xiaoping and Y. Zhang, "Adaptive hybrid algorithms for the sequence-dependent setup time permutation flow shop scheduling problem," *IEEE Transaction on Automation Science and Engineering*, vol. 9, no. 3, 2012.
[12] J. Mercieca and S. Fabri, "Particle swarm optimization for nonlinear model predictive control," in *5th Int. Conference on Advanced Engineering Computing and Applications in Sciences*, 2011.
[13] K. Parsopoulos and M. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications*. 2010.
[14] L. Nicolosi, A. Brusaferri, and A. Ballarino, "A novel toolbox for advanced particle swarm optimization based industrial applications," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pp. 1–8, IEEE, 2014.