An Adapted Variable Neighborhood Search based algorithm for the cyclic multi-hoist design and scheduling problem

Emna Laajili, Sid Ahmed Lamrous, Marie-Ange Manier, Jean-Marc Nicod

An Adapted Variable Neighborhood Search based algorithm for the cyclic multi-hoist design and scheduling problem

Emna Laajili* Sid Lamrous† Marie-Ange Manier‡
Jean-Marc Nicod§
FEMTO-ST institute, Université de Bourgogne Franche-Comté
CNRS / UFC / ENSMM, Besançon, France

Computers & Industrial Engineering Journal (2021) https://doi.org/10.1016/j.cie.2021.107225

Abstract

This paper studies both the design and the cyclic scheduling of multi-hoist treatment surface facilities. Former research have always assumed that the design of the production line is already available with a fixed material handling fleet size. However, in this study, the number of hoists is considered as a decision variable of the problem. The latter is then bi-objective and called the Cyclic Hoist Design and Scheduling Problem (CHDSP). The optimization objective is to determine an optimal cyclic schedule for each possible number of hoists that minimizes the cycle time and thus maximizes the line throughput rate. The achieved results will contribute to building a decision support system that will enable to choose the required number of transportation resources for the production line regarding both aims of productivity maximization and cost savings. An original encoding approach is proposed that both assigns a number of hoists to the line and generates their move sequences, which are evaluated thanks to a mixed integer linear programming model. A rich and well-structured solving algorithm based on variable neighborhood search is developed and adapted to solve efficiently the whole problem. It is also improved by a backtrack procedure that further enhances the findings. Computational experiments are conducted on benchmark problems and demonstrate the high performance and effectiveness of the proposed algorithm that was, in most cases, even able to reach the optimal solutions.

Key words: Cyclic hoist scheduling problem; design dimension; multi-hoist material handling; time window constraints; Variable neighborhood search; backtrack; decision support system

*emna.laajili@utbm.fr
†sid.lamrous@utbm.fr
‡marie-ange.manier@utbm.fr
§Jean-Marc.Nicod@ens2m.fr

1 Introduction

Scheduling plays a key role in production processes as it aims to maximize operation efficiency. It has a fundamental impact on enhancing productivity by lowering production time and costs. The scheduling problem becomes more critical when production processes are automated. It is the case in most contemporary manufacturing systems where transporting parts, throughout the workstations, is performed by robots. This kind of problem is especially encountered in electroplating facilities, where computer-controlled robots (hoists) are used for material handling. One famous among these processes is the production of printed circuit boards (PCBs).

Electroplating is the process of coating an item (electrode) with a thin layer of metal to give it a desired surface property like electric conductivity, resistance to rust, protection against wear and corrosion or even aesthetic qualities. In electroplating facilities, parts must soak in several tanks containing chemical solutions. Each one is needed for a specific step of part processing such as cleaning, acid pickling, plating, rinsing, etc. Tanks are commonly arranged in a row and ordered following the process sequence (Figure 1). Parts, usually mounted on carriers to be processed in batches, are moved from one tank to another one thanks to automated handling hoists. Depending on production requirements, the electroplating line uses one or more identical hoists that circulate on a single track. The production line begins with an input station, the first tank of the line where carriers are prepared and waiting to be loaded. The output station is then the last tank

of the line where processed carriers are unloaded. Sometimes, the loading and unloading occur in the same tank: it is the case of associated input-output stations.

Electroplating systems are often mass production systems where a great and an unlimited number of parts are to be treated, following the same processing sequence. The processing here begins by loading a carrier from tank 1, soaking it in a sequence of tanks and finally unloading it at output station (tank N+1). As all parts go through the same processing sequence, the production becomes uniform and a fixed processing sequence is repeated. This means that the hoists repeat the same sequence of moves. The repetition of the same sequence is the production cycle and the fixed move sequence of the hoists is called the cyclic schedule. The problem of searching the cyclic schedule that minimizes the cycle period is known as the Cyclic Hoist Scheduling Problem (CHSP) [44, 24]. CHSP is one variant among others of the general Hoist Scheduling Problem (HSP) [41]. Most of previous approaches, whatever the HSP variant considered was, have only focused on scheduling hoists moves while supposing that the design of the line is already furnished and the number of hoists is a fixed constant. However, in our study, we deal with both design and scheduling problems together. We suppose that the number of transportation resources is unknown and we try to find the best couples that gather each possible number of hoists with its optimal cycle time. Hence, we tackle a dual problem that we call, the Cyclic Hoist Design and Scheduling Problem (CHDSP). Our approach is at the same time intriguing and challenging. It will provide a decision support system to enable decision-makers to choose a suitable couple of parameters to their production line while regarding investment costs as well as productivity objectives. In addition, as we add a new variable to the problem (the number of transportation resources) its complexity does increase. Thus, adaptive and efficient solving methods will be needed that may be computationally demanding compared to other approaches. This paper, thereby, outlines a new approach to deal with the CHDSP problem. Our objective is to determine an optimal cyclic schedule for each possible number of hoists that minimizes the cycle time and thus maximizes the line throughput rate. The proposed encoding method is original as it both assigns a number of hoists to the line and generates their move sequences. A mixed integer linear programming model [36] is used to evaluate these move sequences to deduce hoist move schedules and calculate associated cycle times. The whole problem is solved due to an adapted and rich Variable Neighbourhood Search (VNS) based approach. The developed solving algorithm is called "AVNS". It is also further improved thanks to a backtrack procedure that enhances its efficiency. We test its performance on benchmark data and compare it with the optimal results of the literature as well as our previous findings [19].

The paper is organized as follows. The next section gives an overview of related work. The third section states the problem and the constraints in scope. In the fourth section, we present a novel encoding idea. Resolution algorithm and fitness evaluation model are described in Section 5. In Section 6, computational experiments and benchmarks are provided and results are analyzed and discussed. A conclusion to our work is drawn in the final section.

2 Related work

2.1 The Hoist Scheduling Problem

The Hoist Scheduling Problem (HSP) has always been an important issue for research since 1976, when Phillips and Unger [41] were the first to propose a Mixed Integer Programming Model for CHSP to maximize the production throughput. They used it to solve an industrial 13-tanks numerical example. In 1988, Shapiro and Nuttle [44] described a linear programming model and associated algorithm to find an optimal cycle for a single hoist system. They used it to solve the same numerical example proposed by [41]. Thenceforth, HSP continued to receive much attention from scientists who proposed various models and methods to tackle one of its variants while considering specific objectives and constraints. An interesting study, worth to cite here, is the classification of HSPs made by Manier and Bloch [34] who identified four main classes (variants) of the problem: Cyclic (CHSP), Predictive (PHSP, [7, 17, 46]), Dynamic (DHSP, [15, 21, 48]) and Reactive (RHSP, [45, 49]). The authors also provided a typology of the different classes and proposed a generic notation for HSP. As in this study we are interested in a cyclic HSP, we will focus more on researches belonging to this class.

2.2 The CHSP in single hoist lines

The CHSP, was the most studied variant in literature with different physical system parameters and production specifications, and then solved with var-

published in Computers & Industrial Engineering Journal (2021)
https://doi.org/10.1016/j.cie.2021.107225

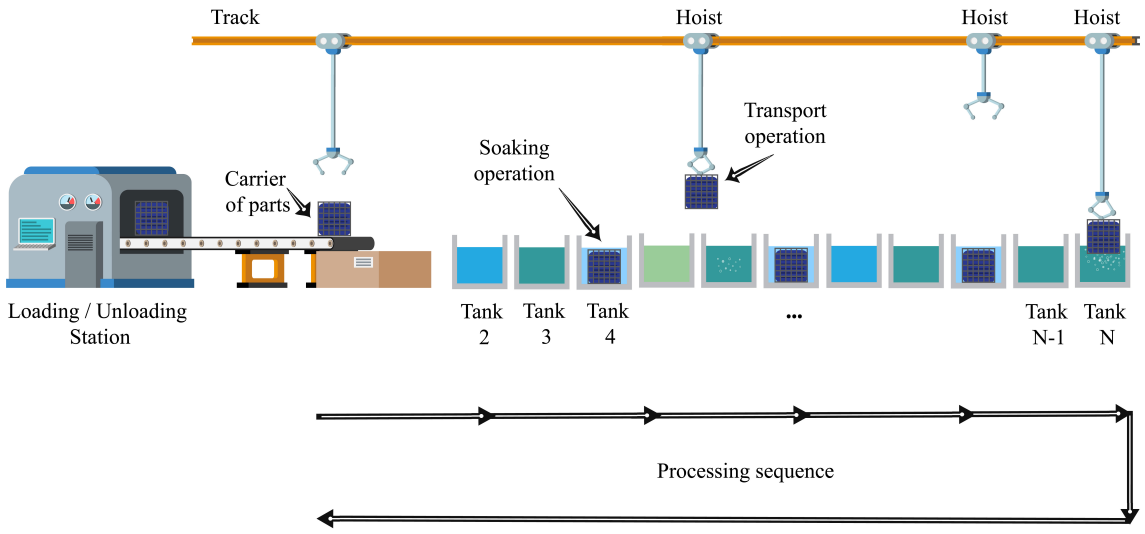

Figure 1: An electroplating line with associated loading-unloading station.

ious methods and approaches. It was broadly investigated in the single hoist case like in the work of [41] or in [44] which considered lines with duplicated tanks (*i.e.*, multi-capacity tanks). Then, many models were proposed. Baptiste et al. [4, 3] used a constraint logic programming model that was simply implemented and able to provide the optimal solution as in [44]. Lei [22] suggested a binary search procedure to find the optimal integer starting times for the operations in a cyclic transportation schedule. Armstrong et al. [2] used a branch and bound solution procedure based on the "Minimal Time Span", a lower bound of the cycle time, to solve an N-stage HSP with time-window constraints. Rather than mathematical programming based approaches, Lim [29] proposed a genetic algorithm based method to solve the HSP.

Unlike previous studies did, NG and Leung [40] assumed that the inter-tank move times should not be constant, *i.e.*, they supposed that pauses are allowed for hoists while moving parts. Other extensions of electroplating systems were also examined. Liu et al. [32] considered production lines with multi-function tanks and multi-tank stages and developed a mixed integer linear programming model to solve the problem. A multi-function tank is a processing tank that might be visited by a carrier more than once and a multi-tank stage is a same stage with many identical tanks. This kind of stages is used when a single tank stage performs a soaking operation with a long processing time and creates a bottleneck stage for the work-flow. Zhou and Li [53] have also treated the single cyclic HSP with bottleneck stages and proposed a multi-tank sequencing procedure to solve it. Che and Chu [9] investigated the single CHSP with these two extensions, as well. They called such systems "large real-life electroplating lines". To solve the problem, they studied analytic properties of this kind of systems and suggested an efficient branch and bound algorithm based on this analysis to avoid dominated or infeasible solutions. Otherwise, Che et al. [12] studied an electroplating facility with multi-type parts and fixed processing times. They used a dynamic branch and bound procedure and tested it on randomly generated test instances. Recently, Feng et al. [16] have tackled the cyclic jobshop hoist scheduling problem where multi-type parts are considered together with multi-capacitated and multi-function tanks. The authors have developed a mixed integer linear programming model to solve the problem.

2.3 The CHSP in multiple hoist lines

The CHSP was also widely studied in the multiple hoist case. A first work to quote here is that of Lei and Wang [25], in 1991. The authors suggested a heuristic algorithm that finds schedules for systems with two hoists using a partitioning approach. This approach was used to partition the system into two sets of contiguous work stations where each hoist is assigned to a set. Then, the "minimum common cycle" algorithm was proposed to search the optimal cycle between the common cycles that are derived from all generated partitions. They solved the industrial example of [41] for two hoists. They

pointed out the advantage of their approach that the hoist assignment was simplified by the partitioning approach (no risk of collision as non-overlapped zones is assumed) but they also affirmed that with the partitioning approach, not all possible solutions are examined. Lei et al. [23] aimed to find a cyclic schedule for a CHSP while minimizing the number of hoists required for the production line. They proposed a heuristic algorithm based also on a partitioning approach that partitions the system into a number of non-overlapping stages, so that the single track constraint is satisfied. The same problem was tackled by Armstrong et al. [1] where they used a greedy local optimization algorithm that maximizes the size of the divided stage zones. Hanen and Munier [18] studied the same problem while assuming that transport moves are arbitrarily assigned to the hoists. They proposed a mixed integer programming model and a branch and bound procedure to solve the problem. Otherwise, Manier et al. [37] dealt with the multi-hoist case in lines that may have duplicated tanks and multi-function tank. As they considered multi-directional sequences of treatments, the partitioning of the line into non-overlapped zones was no longer possible. Thus, they proposed a portioning approach that assigns the transfer operations to the hoists instead of dividing the line into non-overlapped zones that would be assigned to each hoist. To treat the collision free constraint, they imposed a security zone for each hoist that must lie in his circulation zone. The assignment problem was solved due to a heuristic rule whereas for the scheduling problem, the authors presented a model based on constraint logic programming. Later, a different approach to deal with the collision free constraints rather than partitioning approaches has been suggested by Che and Chu [8]. They formulated these constraints as disjunctive inequalities and studied two additional properties that check collision. To solve the problem, they presented a branch and bound algorithm where the two mentioned properties were relaxed. Leung et al. [27] and Che et al. [11] developed a mixed integer programming formulation to model the problem. Che et al. [11] have submitted an improved mixed integer programming approach where they assumed that the loaded moves of the hoists can start and end within different cycles. As for Leung and Levner [26], they studied the multi-hoist CHSP with fixed processing times and proposed an algorithm that finds the minimum number of hoists for all possible cycle times and then determines the minimal time cyclic schedule for the hoists. Zhou and Liu [55] studied the CHSP with overlapping hoist zones in a bi-hoist electroplating line. They proposed a heuristic algorithm to generate transfer operation sequences and assign the sequences to the hoists, and a linear programming model with the collision-free constraints to compute the optimal schedule for each hoist assignment. Che and Chu [10] have investigated the multi-hoist scheduling problem but with constant processing times rather than bounded ones and developed a mathematical model and a polynomial algorithm to solve it. They claimed that the proposed algorithm can serve as a heuristic in solving the same problem but with dependant time-windows. Zhou and Li [54] studied the multi-hoist CHSP with time windows. They suggested a method that divides the system into multi non-overlapping zones and assigns a hoist to each zone, like in [25] and developed a mixed integer linear programming model to treat the scheduling dimension. They also extended their model to solve the cases with bottlenecks that need multi-tank stages. Chtourou et al. [13] proposed a collision test procedure to check collisions, in a bi-hoist electroplating system. Other researches [28, 38] have studied the k-degree CHSP: it is the case where k identical parts join and leave the production line in only one cycle. Both studies developed a MILP model to solve the multi-hoist problem.

2.4 The CHSP with other optimization objectives

The cyclic hoist scheduling problem was most often treated with the objective to find the optimal schedule with the minimal cycle time which will maximize the throughput rate of the system. Few works have considered other optimization objectives rather than scheduling one. With environmental objective, Xu and Huang [47] associated the CHSP and waste minimization and proposed an environment friendly electroplating process. As for Liu et al. [30], they developed a triple objective model that simultaneously optimizes the productivity, the energy saving and minimizes freshwater. The design dimension, however, was rarely tackled associated to the scheduling one. It was sometimes linked to the production line arrangement where the aim was to optimize the spatial allocation of the processing resources, like in Zhao et al. [52]. Qu et al. [42], yet, studied the simultaneous design and operation problem of CHSP but in 2-Dimension case. They assumed that the production line is not necessarily a 1-D line structure but a compact 2-D production line. Otherwise, the simultaneous design and operation hoist problem tackled in our study, called

published in Computers & Industrial Engineering Journal (2021)
https://doi.org/10.1016/j.cie.2021.107225

CHDSP, does not concern the allocation of the processing resources (the soaking tanks) but the quantification of the fleet size of the material handling resources. In other studies dealing with the multi-hoist case, the size of the hoist fleet was fixed beforehand to perform the assignment of transfer operations and then their scheduling. Differently, in our study, the hoist fleet size is assumed unknown and then considered as a decision variable of the problem. Of course, that would bring a harsh challenge in modeling and computation, but it will offer a worthy decision support system allowing to consider at the same time both objectives of productivity maximization and energy and cost savings.

3 Problem description and notation

As was pointed out above, we investigate the CHDSP problem, where we aim to find the optimal cyclic schedule that minimizes the cycle time for each possible number of hoists. The cyclic schedule is a repetitive sequence of moves performed by the hoists. The hoist move durations are not negligible and they are as important as processing times. So, they cannot be ignored. As a result, to ensure productivity, the hoists (also called robots) are seen as the critical resources of the line. These robots are programmed to perform a sequence of loaded and unloaded moves (Figure 2). The hoist performs a loaded move when it transports a carrier between two consecutive tanks (i and $i+1$) as follows: it raises the carrier from the first tank (i), pauses over the first tank (i), if necessary, to allow the carrier drip-off, transports the carrier to the next tank ($i+1$) and finally lowers it into this tank ($i+1$). After each loaded move, the hoist performs an unloaded move where it travels empty to another tank (j) of the line to execute the next scheduled loaded move (transport operation). Therefore, the repetitive sequence of hoist moves is composed of alternate loaded and unloaded moves.

We consider a production line with N tanks (Figure 1). We refer by $E(i,j)$ to every hoist's empty move from tank i to tank j, and by $L(i,i+1)$ to every hoist's loaded move between two consecutive tanks i and $i+1$, where $i,j = 1,\ldots,N$. Times required for loaded and unloaded moves are given constants. Let $d_{i,j}$ be the empty move time and r_i be the loaded move time; $r_i = d_{i,i+1} + c$, where $d_{i,i+1}$ is the empty move time between any two consecutive tanks; c is a constant that gathers the time needed to raise a carrier and to let it drip-off above

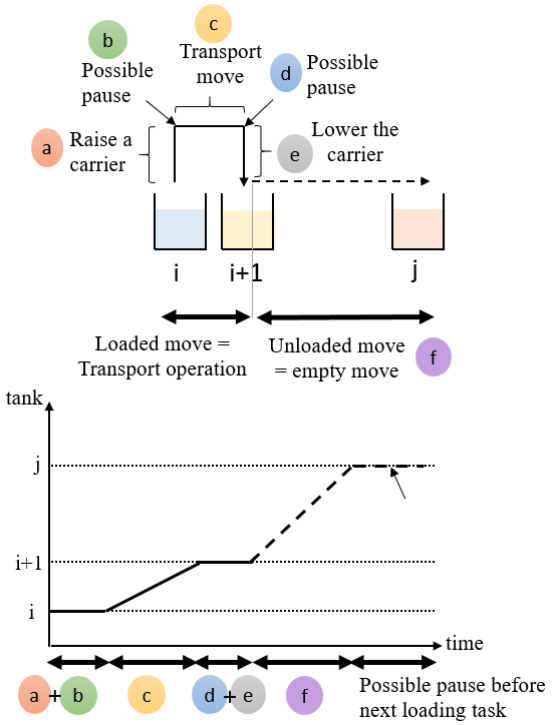

Figure 2: Decomposition of hoist moves.

tank i, plus the time needed to stabilize and lower the carrier into tank $i+1$. The line has associated input-output stations, where loading and unloading carriers belong to tank 1. Tank $N+1$ may be used to refer to the unloading station, which is here equivalent to tank 1 (associated stations). The soaking tanks are then tank 2, tank 3, ..., tank N. Let (nop) be the number of operations of the processing sequence. Loading and unloading carriers are the first and the last operations of this sequence, respectively. Thus, $nop = N+1$. We denote the cycle period by T and the number of hoists of the line by H; $h = 1,\ldots,H$ is the hoist number. We suppose that hoist 1 always begins the processing sequence, *i.e.*, it is the responsible for the first transport operation $L(1,2)$ and then, the empty move $E(2,j)$. In case of multiple hoists, each one has to perform z_h moves and will have a repetitive sequence of moves that we denote by S_h.

The problem we investigate has numerous constraints. First, each carrier has to follow the same processing sequence. Second, tanks are disjunctive resources so that every tank can process only one carrier at a time. This also involves no preemption constraint because a processing step can never be interrupted. Besides, it implies no re-circulation

constraint (*i.e.*, no re-entrance), as each carrier visits every tank just once a time (the tanks are said mono-function ones). Third, a carrier, while soaking, must respect time bounds, *i.e.*, there is a minimal and a maximal processing time (m_i and M_i, respectively) to respect in each tank. If the soaking time lies out of this interval, the parts are less or more coated than the standard requirements and then, they are considered as defective. These results in a no-wait constraint where parts must leave tanks as soon as they finish processing, without any delay. Moreover, each hoist can only transport one carrier at a time and must have enough time to perform empty moves between ordered tanks. As well, we assume that there is no buffer between two soaking operations and that there is no pause during transport operations (we mean hoist pauses are not allowed while moving a carrier, otherwise, the drip-off pause is a constant defined by the process requirements). Besides, the multi-hoist case necessitates to study the collision free constraints because, collision may obviously occur between hoists which share the same moving track. Nevertheless, in the literature of shop scheduling problems with transportation resources (Flexible Manufacturing Systems with AGVs, Robotic Cells with robots and Surface Treatment Facilities with hoists), we can see that the authors often focus mainly on assignment and sequencing problems of machines and transport resources without considering risks of collision ([6], [14], [20], [31], [36], [43], [50], [51]). In our studied problem, these constraints are relaxed in the mathematical model used for evaluation.

As regards the suitable notation for our problem, with respect to the general notation given in [34], we can write it as follows:

$$CHSP|H, N-1, 1//ass|/N+1|(T_{min}, H_{min})$$

It is the Cyclic Hoist Scheduling Problem, with H hoists and N tanks in a single line production system, where the loading and unloading tanks are associated; *nop* operations are performed by each carrier. It is a bi-objective problem aiming at minimizing both the cycle time T and the number of the hoists H.

4 Original encoding/decoding approach

The problem considered does not allow neither wait nor preemption related to processing operations. Hence, hoists' transport operations and processing operations will have correlated beginning and end times. That is, the beginning of a transport operation exactly corresponds to the end of a soaking operation and the end of a transport operation corresponds to the beginning of a soaking one, as can be seen in the GANTT representation of the schedule provided in Figure 4. As a result, the scheduling of processing operations becomes equivalent to the scheduling of hoist moves. On the other hand, the repetitive sequence of hoist moves is composed of alternate loaded and unloaded moves. Thus, scheduling hoist moves is possible based on either loaded or unloaded moves, from which the final cyclic schedule can easily be deduced.

4.1 Empty-move based encoding/decoding

We call solution any possible combination of tank numbers and we denote it by L. It is a list of integers i, $i = 1, \dots N$, where i refers to "tank i" or also "soaking operation i" (except for the unloading operation which takes place in tank 1). Every couple of two successive numbers refers to an empty move occurring between these two tanks. The decoding of the solution is also cyclic where the last couple of tanks is composed by the last element of the list L and the first one of it. That is, if we consider a solution $\{ijk\}$, we can deduce three empty moves associated to this solution: the first is $E(i, j)$ that occurs from tank i to tank j, the second $E(j, k)$ is from tank j to tank k and the last $E(k, i)$ is from tank k to tank i. A solution has a size that ranges between 2 and N. Accordingly, it may not contain all the indices of tanks. The absent numbers will form fictive empty moves as follows: if l is a tank number that is absent in the solution $\{ijk\}$, then we add $E(l, l)$ as fictive empty move and the solution will have four empty moves in all. Once the decoding of overall empty moves from the solution is done, a number H of cyclic move sequences S_h can be deduced. As mentioned before, the move sequence is an alternation of loaded and unloaded moves and the loaded ones occur between two successive tanks i and $i+1$. Therefore, every empty move occurs between two transport operations, *i.e.,* an empty move $E(i, j)$ is preceded by the transport operation $L(i-1, i)$ and followed by $L(j, j+1)$. If it is a fictive empty move such as $E(l, l)$, it is preceded by the transport operation $L(l-1, l)$ and followed by $L(l, l+1)$. In this last case, and after performing the transport operation $L(l-1, l)$, the hoist waits for the carrier above tank l during the processing time before performing next transport operation $L(l, l+1)$. Thus,

published in Computers & Industrial Engineering Journal (2021)
https://doi.org/10.1016/j.cie.2021.107225

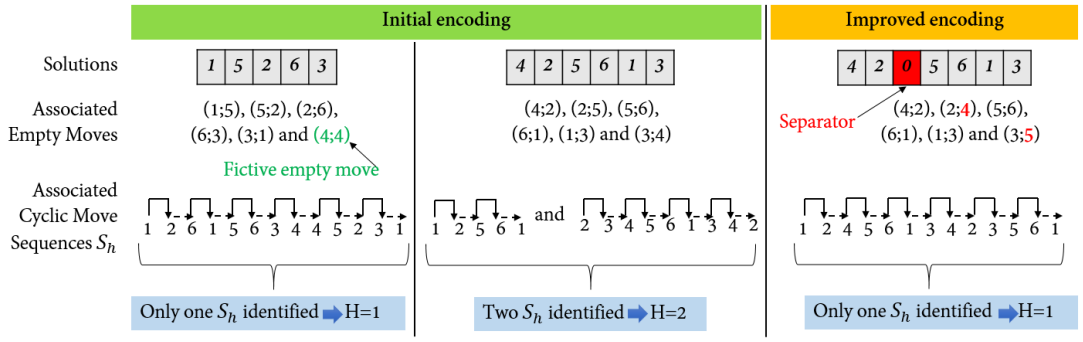

Figure 3: Decoding procedure of the empty moves and identification of move sequences and hoist number.

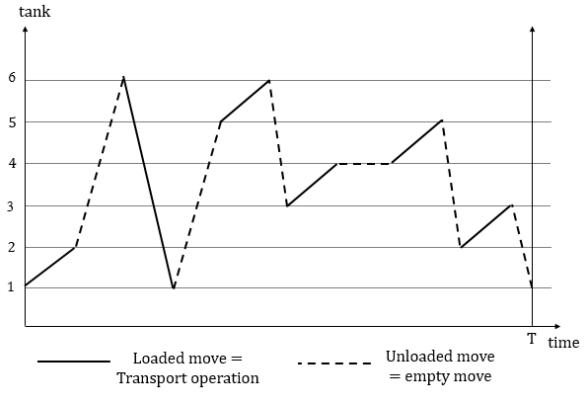

Figure 4: The hoist schedule corresponding to the encoded solution $\{1, 5, 2, 6, 3\}$.

the move is called fictive because it has a duration even though there is no real spatial move of the hoist. Consequently, with this empty-move based encoding, we are able to identify from every possible solution L one or more cyclic move sequences S_h, whose number H corresponds to the number of hoists associated to the solution. The complete decoding process will be detailed in section 5.2, in the Algorithm 1. Figure 3 (on first and middle examples) outlines this ability with two examples of solutions in a six-tank problem ($N = 6$). The first solution is $\{1, 5, 2, 6, 3\}$ where 4 is an absent tank number. We identify the couples $E(1, 5)$, $E(5, 2)$, $E(2, 6)$, $E(6, 3)$, $E(3, 1)$ as empty moves and $E(4, 4)$ as a fictive one. We then can construct the move sequences related to this solution while alternating transport and empty moves. The transport operations, as they occur between successive tanks do not change within the same case problem. Here, with six-tank problem, we will always have the loaded moves $L(1, 2)$, $L(2, 3)$, $L(3, 4)$, $L(4, 5)$, $L(5, 6)$ and $L(6, 1)$. To build the move sequences, and without loss of generality, we always begin a cycle by the loaded move $L(1, 2)$. By applying this method on the first solution, we identify only one cyclic move sequence (as depicted in Figure 3) performed by hoist 1 ($H = 1$, $S_h = S_1$). The schedule of this move sequence is given in Figure 4.

With the same procedure, however, from the second solution $\{4, 2, 5, 6, 1, 3\}$ of Figure 3, we deduce two cyclic move sequences respectively executed by hoists 1 and 2 ($H = 2$, $S_h = \{S_1, S_2\}$). Indeed, as already said, we always start by building the move sequence of hoist 1 ($H = 1$) with the loaded move $L(1, 2)$. After alternation of loaded and unloaded moves, this first cyclic sequence is obtained $L(1, 2) - E(2, 5) - L(5, 6) - E(6, 1)$. As this sequence does not contain all the loaded and unloaded moves, a second sequence can be built. The latter will begin with one of the loaded moves that has not been used yet and that has the lowest tank number for departure. In our case, the remaining loaded moves after building the first move sequence are $L(2, 3), L(3, 4), L(4, 5)$ and $L(6, 1)$. Then, we begin the second sequence with the loaded move $L(2, 3)$ and after the alternation procedure, we get a cyclic move sequence: $L(2, 3) - E(3, 4) - L(4, 5) - E(5, 6) - L(6, 1) - E(1, 3) - L(3, 4) - E(4, 2)$. As the second move sequence obtained includes all the remaining loaded and unloaded moves, the decoding procedure is closed.

It is to note that in literature, almost all previous works used the hoist's loaded moves to encode the move sequences [41, 44, 22, 27, 13, 16]. However, in [35] and [36], Manier et al. were the first to propose a different approach based on the hoists' empty moves. The latter is original as it enables to represent one or more cyclic move sequences S_h at a time, whose number corresponds to the number of hoists. The approach is then able to generate solutions

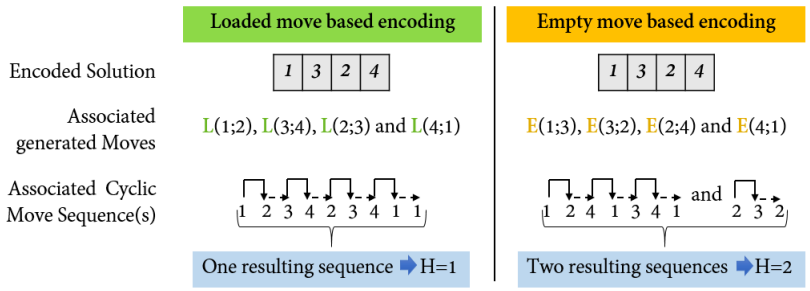

Figure 5: Different outcomes between the loaded move and the empty move based encodings.

with variable number of hoists (H), and to deal with the design dimension of the CHDSP. Figure 5 shows the difference on the outcome between the loaded move based encoding and the empty move based one. The encoded solution used to generate the move sequences is $\{1, 3, 2, 4\}$. The principle of loaded move generation is to consider one by one the numbers figuring in the solution as the origin tanks of the loaded moves of the sequence. Hence, the scheduling of the loaded moves following the encoded solution is $L(1, 2), L(3, 4), L(2, 3)$ and finally, $L(4, 1)$ which results in only one move sequence corresponding to a single hoist. The principle of empty move generation [19], as deployed in details above, provides the empty moves $E(1, 3), E(3, 2), E(2, 4)$ and $E(4, 1)$ which results in two move sequences corresponding to two hoists. Therefore, this example outlines the ability of the empty move based encoding to provide more than one move sequence, then to deal with more than single hoist lines, compared to the loaded move based one and generate solutions with different couples (H, S_h).

4.2 Improvement Approach

The encoding approach, described above, has revealed a limitation as it cannot generate all possible solutions of the search space. This was proven based on analysis carried out on some best solutions of benchmarks of the literature [19]. Hence, to maintain the same decoding properties and at the same time to enable the empty-move encoding approach to generate the solutions that were not reachable before, we suggest introducing separators to the encoding approach. The addition of separators to the adopted encoding enables to increase the number of the represented solutions. Indeed, more couples (H, S_h) will be generated, which allows to diversify the search space, or even complete it. In any solution, a separator is encoded as the number zero located in random positions. Accordingly, every solution will contain the tank numbers $i \in \{1, \ldots, N\}$ and some zeros, each referring to a separator (see Figure 3, right case). The total size of the new code is then the sum of the integer numbers i and the number of separators. Nonetheless, the principle of empty moves' generation is slightly modified when applying separators. The separators are inserted between two consecutive tank numbers in the solution. We denote these insertion positions by inter-tank positions. Thus, each separator divides the solution into two parts. A separator indicates the end of the first part and the beginning of the second one. Then the decoding procedure indubitably remains the same for each part. As the decoding of one solution has to be cyclic, the decoding of each part is cyclic too. Then, we can get a different list of empty moves that results in a different couple (H, S_h). Figure 3, at its right case, depicts this advantage on the same second solution $\{4, 2, 5, 6, 1, 3\}$ where a separator is inserted on the second inter-tank position between tanks 2 and 5 (*i.e.* it is equivalent to obtain two gathered cyclic solutions:$\{4, 2\}$ and $\{5, 6, 1, 3\}$). The separator modifies two of the empty moves of the initial solution example (middle case) generated without separator: the empty move $(2, 5)$ is replaced by $(2, 4)$ and the empty move $(3, 4)$ is replaced by $(3, 5)$. This modification of empty moves influences the resulting couples (H, S_h). In the above example, only one cyclic sequence move (instead of two) is identified, that corresponds to a single hoist ($H = 1, S_h = S_1$). Next, we will refer by empty-move based encoding approach to the whole improved encoding approach with separators described above. We then have noticed that the application of separators on some given inter-tank positions should be avoided as it does not generate new move sequences. The situations to be evaded are that two separators occupy:

(a) the same inter-tank position, because both are considered as only one separator;

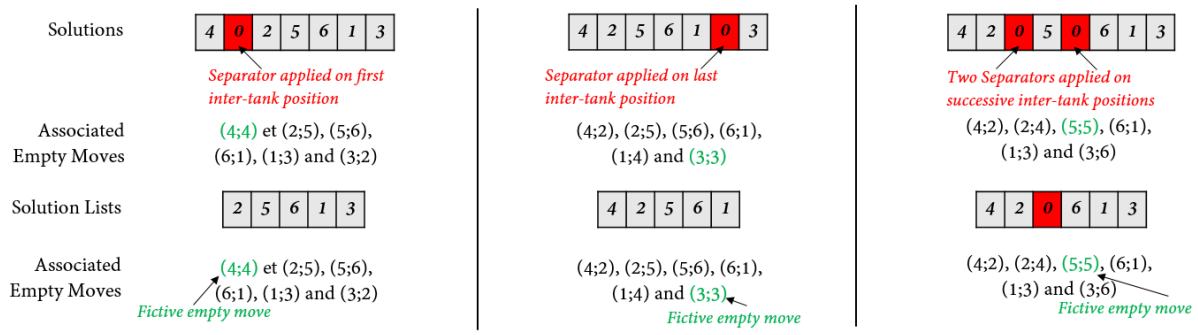

Figure 6: Inter-tank positions to be avoided when applying separators

(b) the first or the last inter-tank positions;

(c) two successive inter-tank positions.

Actually, the application of a separator on one of the situations (b) or (c) leads to obtain a segment of solution of only one tank number $\{i\}$. This single-tank-segment represents a single empty move $E(i, j)$, which, according to the cyclic decoding method, begins and ends at the same tank number. The resulting empty move is then similar to a fictive empty move provided by an absent tank number. Thus, in these both situations, we can meet the same resulting empty moves as from a solution in which number i is absent. Figure 6 escorts the last idea with the solution example $\{4, 2, 5, 6, 1, 3\}$, considered in a six-tank problem ($N = 6$). On the left side, we apply a separator on the first inter-tank position of the solution. The associated empty moves are the same as those identified from the solution $\{2, 5, 6, 1, 3\}$ where $(4, 4)$ is a fictive empty move. On the middle, we show the case where a separator is applied on the last inter-tank position. The associated empty moves are the same as those deduced from the solution $\{4, 2, 5, 6, 1\}$ where $(3, 3)$ must be added as a fictive empty move. On the right side, we have applied two separators on two successive inter-tank positions: the second and the third ones. The associated empty moves are the same as those identified from the solution $\{4, 2, 0, 6, 1, 3\}$ with only one applied separator on the second inter-tank position and where $(5, 5)$ is a fictive empty move. Therefore, we prohibit the application of separators on these positions to avoid repetition of a number of generated move sequences. Possible inter-tank position is then defined as the inter-tank position that, after applying a separator, can generate different list of empty moves and then totally different move sequences, that cannot be reachable without the application of this separator.

5 Adapted Variable Neighborhood Search Algorithm

The CHDSP problem that we tackle has two dimensions: the design dimension where we search the minimal number of hoists, and the scheduling dimension where we aim to find the optimal cyclic schedules of hoist moves and deduce the minimal cycle period T for a given number of hoists; the second dimension was largely studied in the literature and is known as the Cyclic Hoist Scheduling Problem, CHSP. It has been proven to be NP-hard, even for the single hoist case [24]. Accordingly, the multi-hoist scheduling problem with a single track is further complicated and makes it hugely harder to find the optimal solution. It is the case for our problem with a further difficulty as the number of hoists is a variable of our problem. In fact, the search space is already huge whereas few feasible solutions generally exist considering the numerous constraints to respect. Then, to be able to achieve the best solutions, an extra challenge is to increase the efficiency of the search procedure, in order to reach a maximum of feasible solutions and to avoid local minima. This problem extension is further challenging and due to its complexity, efficient solving algorithms should have a real potential for application. Metaheuristic algorithms are then the first to be nominated regarding their high performance to solve sub-optimally NP-hard problems, among which, we have chosen the Variable Neighborhood Search (VNS) to solve the handled CHDSP problem. VNS algorithm was introduced by Mladenovic and Hansen [39] as a metaheuristics that proceeds to a systematic change of the neighborhood within a local search algorithm. This makes it a simple and effective metaheuristics for combinatorial optimization problems. According to them, VNS, unlike

most of other local search methods, does not follow a trajectory, but explores increasingly distant neighborhoods of the current incumbent solution, and jumps from there to a new one if and only if an improvement was made. VNS, therefore, was chosen regarding its structural characteristics of exploring the search space that crucially influence the performance of heuristic or metaheuristic algorithms. Due to the size and other characteristics of the CHDSP search space, the VNS metaheuristics, itself was not applied as its standard version described in [39] but modified and adapted to potentially deal with the specificity of the handled problem. We call it the Adapted Variable Neighborhood Search (AVNS). It is then applied to solve the handled bi-objective optimization problem CHDSP. As for each number of hoists H, there is associated cyclic move sequences S_h (deduced from a solution L) evaluated to have a cycle time T, we denote a solution of the CHDSP, as the triplet (H, S_h, T) or (H, L, T) or simply the couple (H, T).

The general solving algorithm of AVNS is shown in Figure 7. Other procedures are needed implicitly either to decode solutions to identify associated move sequences and hoist number, or to evaluate move sequences to deduce the corresponding cycle time. They are called respectively, "decoding and identification procedure" and "evaluation procedure". Table 1 provides the notations that will be used throughout the next sections, where useful details will be deployed to describe all the consistent steps and procedures of AVNS.

Table 1: Notation

N	number of processing tanks.
i	tank index, $i = 1, \ldots, N$.
$ITER$	maximum number of iterations of the general solving algorithm.
$iter$	iteration index, $iter = 1, \ldots, ITER$.
L_1	initial solution.
$(S_h)_1$	identified move sequences from L_1.
H_1	identified number of hoists from L_1.
T_1	cycle time corresponding to L_1.
L_{inc}	incumbent solution of the general solving algorithm. It is always the origin of neighboring solutions.
$L_{best}(H)$	global best solution for the number of hoists H.
$T_{best}(H)$	global best cycle time for the number of hoists H.
S	neighborhood size. It is the number of neighboring solutions generated at each iteration.
s	neighborhood size index, $s = 1, \ldots, S$.
N_{type}	preselected neighborhood structures used to generate neighbor solutions, "type" is the title given to the transformation or the move method adapted to create the neighbor solution (details are next afforded).
$N_{type}(L_{inc})$	the set of neighbor solutions obtained with the neighborhood structure of title "type" from incumbent solution L_{inc}.
L_{neigh}	neighbor solution.
$(S_h)_{neigh}$	identified move sequences from L_{neigh}.
H_{neigh}	identified number of hoists from L_{neigh}.
T_{neigh}	cycle time corresponding to L_{neigh}.
$(L_{neigh})_{best}$	best neighboring solution.
$(H_{neigh})_{best}$	number of hoists of the best neighbor solution.
$(T_{neigh})_{best}$	cycle time of the best neighbor solution.

5.1 Initialization procedure

The variables $iter$, $L_{best}(H)$, $T_{best}(H)$ are first initialized. As we solve the problem for different number of hoists H, and beforehand, we do not know how many the search will reach for the number of hoists, we initialize $L_{best}(H)$ and $T_{best}(H)$ for $H = i = 1, \ldots, N$. Precisely, $L_{best}(H)$ is a matrix of a size N and $T_{best}(H)$ is a vector of size N. $L_{best}(i)$ are initialized as empty vectors and all $T_{best}(i)$ begin with the value 2000 as an upper bound of the objective T. This upper bound was chosen randomly, but a little bigger than the optima of the cycle time T of the solutions of one hoist ($H = 1$), found in the literature, for all tested instances.

We start at $iter = 1$. An initial solution L_1 is randomly generated, as follows: First, we generate randomly a number N_1 between 2 and N (the number of processing tanks). Then, we iteratively build a solution in N_1 steps. At each step, we complete the partial solution by adding a randomly selected tank

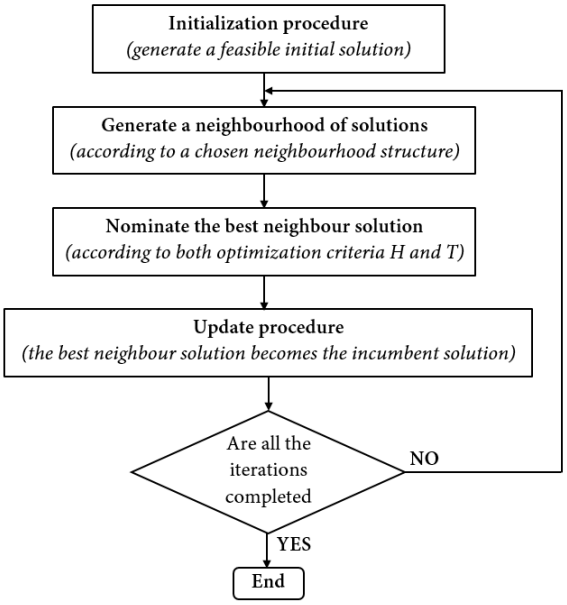

Figure 7: The general solving approach

number, such as any number can appear only once in the solution. After that, we integrate a random number of zeros (separators) on randomly chosen inter-tank positions, according to the rules defined in Section 4.2, and illustrated in Figure 6. To obtain this initial solution L_1, we generate as many solutions as necessary until we get a feasible solution ($T_1 \neq 0$), whose number of hoists is 1 or 2 hoists ($H_1 = 1$ or $H_1 = 2$). This kind of solution is judged as a good initial solution to begin the search with as it is forced to be feasible and with the smallest number of hoists.

L_1 is then decoded using the decoding and identification procedure (Algorithm 1), to deduce its related move sequences $(S_h)_1$ and hoist number H_1. The identified move sequences $(S_h)_1$ are evaluated using the evaluation procedure, to compute the cycle time T_1. We get here the initial resulting solution (H_1, T_1). Then, the initial solution becomes both the incumbent solution L_{inc} and the global best solution L_{best} of hoist number H_1. The cycle time T_1 is assigned to the global best cycle time T_{best} of H_1 and we iterate $ITER$ times the procedures 5.5, 5.6, 5.7 and 5.8.

5.2 Decoding and identification procedure

This procedure enables us to deal with the design dimension of the CHDSP. It is needed each time a new solution is generated. With respect to the description given in Section 4, it identifies the associated empty moves and deduces the associated cyclic move sequences S_h and hoist number H for each generated solution. See Algorithm 1 for more details.

5.3 Evaluation procedure

The evaluation procedure uses the mixed integer linear programming model that was proposed by Manier et al. [36]. It was designed and applied to deal with the same problem as that undertaken in this study. This model enables us to deal with the scheduling dimension of the CHDSP. For each decoded solution, it evaluates the deduced cyclic move sequences S_h to test if they respect all the considered constraints (stated in paragraph 3), and to compute the minimal cycle time T, while synchronizing the H move sequences. If the move sequences respect all constraints, the corresponding solution is said feasible and the model provides a minimized non-zero value for the cycle time T, otherwise, the solution is said non-feasible and the resulting cycle time is set to null.

After both procedures 5.2 and 5.3, we get for each generated solution the triplet (H, S_h, T).

5.4 Preselected neighborhood structures

The implemented AVNS uses several preselected neighborhood structures to generate rich neighborhoods. A neighborhood structure is a method that slightly perturbs a solution to give a new different one. It acts on one or some attributes of the solution to change their values or their places. The main idea is to improve, if possible, the incumbent solution by performing series of transformations or moves. The new solution is the neighbor solution. The transformations are then specified by the neighborhood structure. They predefine a type of moves with which neighbor solutions are generated. Neighborhood structure that limit changes to k components of the solution is often called k-optimal (k-opt). In this study, we have adopted six neighborhood structures N_{type} of different types, to be involved in the AVNS. The six types are insertion, deletion, replacement, swap, shift and inversion and are respectively denoted by $N_{insertion}, N_{deletion}, N_{replacement}, N_{swap}, N_{shift}$ and $N_{inversion}$. We next give further details on the adapted neighborhood structures.

Algorithm 1: Decoding and identification

```
Input: L: a solution including separators
Output: S, H: set of move sequences and the number of hoists
begin
    S ← ∅
    h ← 1                                   // 1st move sequence
    k ← 2          // index of the move sequence under construction
    S_h(1) ← 1          // 1st sequence performs the 1st loaded
    S_h(2) ← 2      // 1st move (from tank 1 to tank 2)
    while not all empty moves are identified do
        // define S_h(k+1) s.t. (S_h(k), S_h(k+1)) is the next identified empty move:
        if S_h(k) does not exist in the solution L then
            S_h(k+1) ← S_h(k)   // fictive empty move
        else
            if S_h(k) is followed by "0" in L (separator) then
                if first separator in L then
                    S_h(k+1) ← L(1)      // close the 1st cycle
                else
                    Ns ← separator number of S_h(k) in L
                    Ps ← position of separator Ns − 1 in L
                    S_h(k+1) ← L(Ps+1)   // close the current cycle
            else
                if S_h(k) is the last tank of L then
                    Ps ← position of the last separator
                    S_h(k+1) ← L(Ps+1)   // close the last cycle
                else
                    Ps ← the position of S_h(k) in L
                    S_h(k+1) ← L(Ps+1)
        if S_h(k+1) = S_h(1)   // The move sequence is a cycle then
            S ← S ∪ S_h
            if not all empty moves are identified then
                h ← h+1                 // next hoist h
                List ← tanks that have not been affected yet as a destination of one empty move
                S_h(1) ← min(List)      // 1st tank of the new sequence of hoist h
                S_h(2) ← S_h(1)+1   // destination of the 1st loaded move insured by hoist h
                k ← 2
        else
            // define S_h(k+2) s.t. (S_h(k+1), S_h(k+2)) is a loaded move:
            if S_h(k+1) = N   // number of processing tanks then
                S_h(k+2) ← 1   // loaded move (N, 1)
            else
                S_h(k+2) ← S_h(k+1)+1      // each loaded move goes to the next tank
            k ← k+2   // 2: empty move plus loaded move
    H ← h
    return S, H
```

5.4.1 $N_{insertion}$

The insertion neighborhood structure can only be applied if the size of the solution is different from N. In this case, we identify the tank numbers that are absent in the incumbent solution. Then, we randomly choose some of them that we insert on random positions of the incumbent solution. This neighborhood structure is $k - opt$ where k is equal to the number of the tank numbers that are inserted. Every added tank number causes to remove one empty move from the identified empty moves of the incumbent solution and replace it by two new empty moves in the list of identified empty moves of the neighbor solution.

5.4.2 $N_{deletion}$

The deletion neighborhood structure can be applied whatever the size of the incumbent solution. We first choose a random position on this solution while avoiding separator positions. We avoid the case that the tank number occupying this position, after deletion, will lead to a unique tank number in between two separators. We then delete the tank number that occupies this position. We always delete only one random tank number from the incumbent solution. This neighborhood structure is then $1 - opt$. Every removed tank number causes to remove two empty moves from the identified empty moves of the incumbent solution and replace them by one new empty move in the list of identified empty moves of the neighbor solution.

5.4.3 $N_{replacement}$

Like the insertion structure, the replacement neighborhood structure cannot be applied if the incumbent solution has a size of N tanks. We first identify the tank numbers that are absent in the incumbent solution, and randomly choose one of them. We then replace it with one tank number of the incumbent solution, randomly selected. We never make a replacement with a separator. This neighborhood structure does not alter the size of the incumbent solution. The replacement neighborhood structure is $1 - opt$. Every changed tank number causes to modify two empty moves from the identified empty moves of the incumbent solution. The neighbor solution will then have two new identified empty moves.

5.4.4 N_{swap}

The swap neighborhood structure, yet, does not have any constraint on the solution size. We begin with choosing randomly two different positions t_1 and t_2 on the incumbent solution. Moreover, we never allow picking at the same time two separator positions as this case does not produce any difference between the incumbent and the neighbor solution. Afterward, we interchange the values at these two positions t_1 and t_2. This neighborhood structure does not alter the size of the incumbent solution, as well. Unlike the previous structures, the swap neighborhood is $2-opt$, because there is perturbation of two components of the incumbent solution. The swapped values, cause to modify four empty moves from the identified empty moves of the incumbent solution. The neighbor solution, thereby, will have four new identified empty moves.

5.4.5 N_{shift}

The shift neighborhood structure starts with choosing randomly two positions t_1 and t_2 on the incumbent solution. Hereafter, we circularly shift the part of the incumbent solution delimited by the two positions t_1 and t_2 (with $t_1 < t_2$). Clearly, this neighborhood structure does not modify the size of the incumbent solution. The shift neighborhood is $2-opt$. That is, the tank number on the position t_2 is removed from its position and is inserted on the position t_1. The tank numbers from the position t_1 to the position $t_2 - 1$ are just shifted each by one position to the right and this move does not cause to modify the empty moves identified from this part of the solution. As a result, the shift neighborhood is similar to a deletion-insertion neighborhood that performs a deletion of the tank number of position t_2 before its insertion on the position t_1. The removal of tank number from position t_2 causes to replace two empty moves of the incumbent solution by a new one. Its insertion on t_1 causes to substitute one empty move of the incumbent solution by two new empty moves. The neighbor solution, hence, will have three new identified empty moves.

5.4.6 $N_{inversion}$

The inversion neighborhood structure, is like the shift structure, can be applied whatever the size of the solution but has an interesting effect when the size of the tank part is important. It first chooses two random positions t_1 and t_2 on the incumbent solution. Then, it inverses the part in between so that the last tank number on position t_2 becomes the first one of this part and occupies the position t_1, the tank number before last of position $t_2 - 1$ becomes the second one, and so on until the first tank number of position t_1 takes the last position t_2. The size of the incumbent solution is never modified. This neighborhood structure is $k-opt$, as it perturbs k components of a random size on the incumbent solution, where k is equal to the part being inverted. In cases when the decoding of the solutions results in couples or arcs on which a different orientation does not affect the cost of the move (the move cost or distance on the arcs does not change if the move orientation changes), this neighborhood structure is only $2-opt$. In fact, the solution part from t_1 to t_2 is inverted such that it causes to modify only the orientation of the empty moves that are identified beginning with tank number of position t_1 until tank number of position t_2. The couples of tank numbers that make up these empty moves remain the same. Thus, only the tank numbers on positions t_1 and t_2 cause to completely modify two empty moves, as they are connected each with a new tank number.

To ease the comprehension of these neighborhood structures and their predefined moves, we illustrate them using the following example of incumbent solution L_{inc}: $\{0, 7, 2, 5, 13, 8, 4, 9, 3, 10, 6, 1, 0\}$, in a 13-sized-problem-case (see Figure 8). The size of the incumbent solution is 13 with 11 tank numbers and 2 separators, placed on the extremities of the solution. The two absent tank numbers are 11 and 12. The figure depicts each already described neighborhood structure N_{type} where $type \in \{insertion, deletion, replacement, swap, shift, inversion\}$. We show, for every neighborhood structure, how moves are performed on the incumbent solution L_{inc} to obtain the neighbor solutions L_{neigh}. For the insertion neighborhood structure $N_{insertion}$, the tank number 12 is randomly chosen from the absent tank numbers of L_{inc} and is randomly inserted on the inter-tank position which lies between the tank numbers 8 and 4, as shown on the neighbor solution L_{neigh}. For the deletion neighborhood structure $N_{deletion}$, the random position which lies between the tank numbers 8 and 9 is randomly chosen and the tank number 4 that occupies this position, is deleted as shown by the corresponding L_{neigh}. For the replacement neighborhood structure $N_{replacement}$, 12 is randomly chosen from the absent tank numbers of L_{inc} and randomly takes the place of the tank number 9 on the neighbor solution L_{neigh}, which lies between tank numbers 4 and 3. For the swap neighborhood structure N_{swap}, the two positions of tank numbers

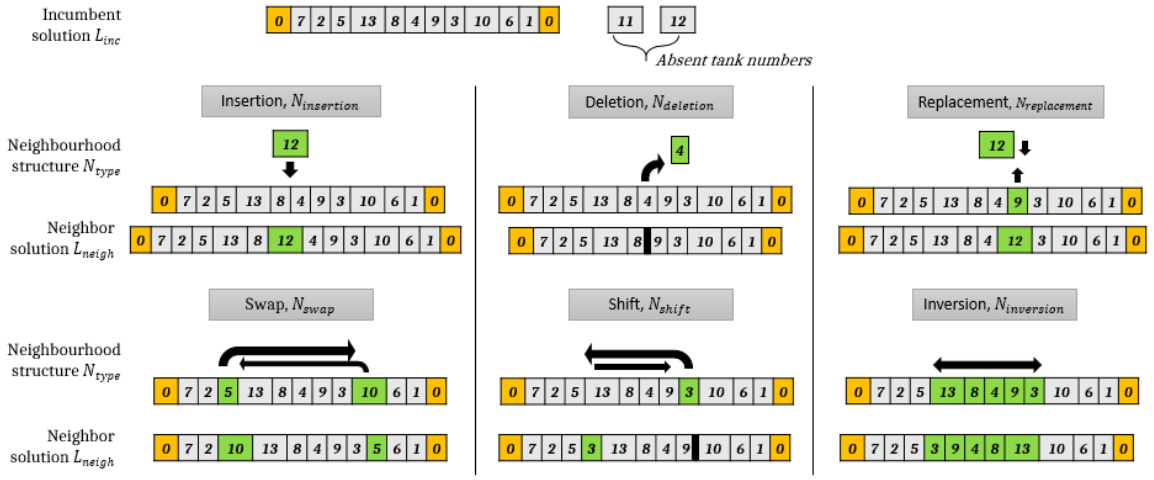

Figure 8: Illustration of the preselected neighborhood structures and their predefined moves

10 and 5 are randomly chosen and then, the tank numbers 5 and 10 are interchanged between these two positions. For the shift neighborhood structure N_{shift}, the two positions of tank numbers 13 and 3 are randomly chosen. The tank number 3 is inserted on the position of the tank number 13, and the tank numbers between both chosen positions (beginning from 13, 8, 4 to 9) are shifted to the right each by one position. For the inversion neighborhood structure $N_{inversion}$, the same two random positions of tank numbers 13 and 3 are randomly chosen and the part of tank numbers in between is inverted as explained in Section 5.4.6.

5.5 Choice of a neighborhood structure

At each iteration *iter* of the general solving algorithm AVNS, only one neighborhood structure is selected from the six predefined structures N_{type}. It is then applied for the generation of the S-sized-set $N_{type}(L_{inc})$ of neighbor solutions L_{neigh}. To accomplish this selection, we use a procedure which performs a random choice of one neighborhood structure depending on the size of the tank part of the incumbent solution L_{inc}. In fact, the tank part is the size of the solution without counting the number of its separators. We denote it by TP. Here, we debrief this procedure:

- If $2 \leq TP \leq 4$, this procedure chooses always only the insertion structure $N_{insertion}$ to be applied.
- If this size is equal to its maximum ($TP = N$), we can only apply one of the following four structures: deletion, swap, shift or inversion. Thus, the choice is performed randomly as follows: each structure has the probability of $1/4$ to be chosen. A number *prob* in the interval $[1, 4]$ is randomly generated and depending on its value, one neighborhood structure among these four stated is selected. The number *prob* is generated by a random function that returns a random scalar integer between 1 and the value of its attribute.
- Otherwise, if $5 \leq TP \leq N-1$, we have no constraint on choice, so that, we choose randomly one of the six predefined structures. Likewise, each structure here has the probability of $1/6$ and the number *prob* is randomly generated in the interval $[1, 6]$. Depending on its value, one of the six neighborhood structures is chosen.

5.6 Neighborhood generation procedure

At the iteration *iter*, after specifying the neighborhood structure to be applied, the Neighborhood generation procedure generates S different solutions L_{neigh} from the incumbent solution L_{inc}, with respect to the N_{type}-move-mechanism. Afterwards, every generated neighbor solution L_{neigh} goes through the decoding and identification procedure (5.2) to deduce its related hoist number H_{neigh} and move sequences $(S_h)_{neigh}$. Then, the identified move sequences $(S_h)_{neigh}$ are evaluated with respect to the MILP model (5.3) to verify their feasibility and to compute their cycle

time T_{neigh}. Eventually, we obtain S neighbor solutions L_{neigh} with their variable triplet-results $(H_{neigh}, (S_h)_{neigh}, T_{neigh})$, among which there may be non-feasible solutions.

5.7 Election procedure

This procedure gets the resulting variables H_{neigh}, T_{neigh} related to the generated neighbor solutions L_{neigh} and compares them to elect the best neighbor solution $(L_{neigh})_{best}$ of the iteration *iter*. At the beginning, we exclude all the non-feasible neighbor solutions and we keep only the feasible ones ($T_{neigh} \neq 0$). Next, we nominate the best neighbor solution based on the two objective variables H and T. As the tackled problem is bi-objective, we prioritize one objective over the other one during the election operation. Hence, two actions are possible here: either determine the neighbor solutions with the minimal cycle time $(T_{neigh})_{best}$ (we can find more than one neighbor solution with the same cycle time) and if there are many, we nominate the one with the minimal number of hoists $(H_{neigh})_{best}$, or identify the neighbor solutions with the minimal number of hoists $(H_{neigh})_{best}$ and if there are several, we nominate the one of them with the minimal cycle time $(T_{neigh})_{best}$. It was obvious, and even shown with some first simulations, that the first choice is not the perfect one. When we prioritize the minimization of the cycle time, we will often get the neighbor solutions with the minimal cycle times whose number of hoists H are obviously not among the minimal ones *i.e.* the more the number of hoists of a production line increases, the more the cycle time value decreases. When this election option gets repeated over the iterations, we get further from the neighborhoods of solutions of minimal number of hoists. Accordingly, we have assumed to apply the second election option, which privileges first the minimization of the number of hoists H, to determine the best neighbor solution during the election procedure.

In other respects, in very rare cases, there may not have any feasible solution within a generated neighborhood.Thus, in this case, we randomly elect one of the neighbor solutions that have the minimal number of hoists $(H_{neigh})_{best}$ to be the best neighbor solution $(L_{neigh})_{best}$, even though $(T_{neigh})_{best} = 0$. This case has a positive impact on the diversification of the search space of solutions of minimal number of hoists that are the more difficult to achieve.

5.8 Update procedure

Given the adopted encoding method, any perturbation of a solution may change the value of the two objectives (H, T) and leads to a totally different solution. Thus, throughout the optimization procedure, each possible number of hoists will have an improvement scheme, *i.e.,* we will improve the cycle time T for every possible hoist number. Each time a new solution (H, T) is accepted, we will compare its cycle time T to the best cycle time that has been registered for the hoist number H. That is, comparison must always occur between cycle times of the same number of hoists.

Here, two tests are applied on $(L_{neigh})_{best}$: if it is already feasible and $(T_{neigh})_{best}$ is lower than the global best cycle time T_{best} ever found for hoist number $H = (H_{neigh})_{best}$, then $T_{best}(H)$ gets the value of the new best neighbor cycle time $(T_{neigh})_{best}$ and likewise, $L_{best}(H)$ gets the best neighbor solution elected $(L_{neigh})_{best}$. After that, the incumbent solution L_{inc} gets the best neighboring solution $(L_{neigh})_{best}$ and the algorithm loops with the procedure 5.5 until it reaches the last iteration $ITER$.

The last step means that the algorithm always moves to the new best neighbor solution $(L_{neigh})_{best}$ even if there is no further improvement of the cycle time, because the new best neighbor solutions will have different and random number of hoists. In fact, we test if there is an improvement of the best cycle time T_{best} registered for $H = (H_{neigh})_{best}$. If an improvement exists, this means that a new improved value of T, $(T_{neigh})_{best}$, is found for T_{best} of $H = (H_{neigh})_{best}$. Then, we assign $(T_{neigh})_{best}$ to $T_{best}((H_{neigh})_{best})$ and the corresponding new improved solution $(L_{neigh})_{best}$ to the last best solution $L_{best}((H_{neigh})_{best})$ registered for $H = (H_{neigh})_{best}$.

As the incumbent solution may have a number of hoists H_{inc} different from $(H_{neigh})_{best}$, then the cycle time of the best neighbor solution $(T_{neigh})_{best}$ may also belong to a different number of hoists than H_{inc}. Thus, the comparison between the cycle time of incumbent solution T_{inc} and $(T_{neigh})_{best}$ is not logic here. That is why we do not define or use neither H_{inc} nor T_{inc} and on each iteration, we move to the new best neighbor solution without need of improvement of T_{inc}. This also means that we change on each iteration the incumbent solution L_{inc} by the best generated neighbor solution $(L_{neigh})_{best}$. However, in general cases, when any meta-heuristic based on local search is applied, there are two possible actions that can be performed at each iteration: either accept a move to the new elected best neigh-

bor solution only if an improvement of the incumbent solution exists, or accept a move even if there is no improvement. The first case is known as a "descent method" while the second one is a "descent-ascent method". Moreover, if at each iteration of the algorithm, there is only one generated neighbor solution (unique size neighborhood, $S = 1$), we will always judge the first explored solution in the neighborhood, which is known as "the first-improvement" method. However, when all neighbor solutions are generated among which we select the best one regarding the objectives to optimize, we are then applying the "best-improvement" method. Yet, when the problem is NP-hard, its complexity growths exponentially with the increase of its size. It becomes then harder and time costly to generate all possible solutions in each considered neighborhood. In these cases, it remains possible and more logical to generate a given number of neighbor solutions at each iteration of the algorithm, from which we select the best one. The neighborhood will then have a defined size. Consequently, according to the latter definitions of possible search strategies, we deduce that our applied **AVNS is a descent-ascent, best improvement search method**. In addition, a stopping criterion in meta-heuristic algorithms can be either a maximum number of iterations, a maximum number of iterations after last improvement, or a maximum allowed computing time. In our case, we have chosen to iterate the main steps for a given maximum number of iterations $ITER$.

6 AVNS algorithm with backtrack

In order to further improve the outcome of the AVNS algorithm, we have also tested it with an integrated backtrack procedure. The latter has only interfered into the procedure 5.1 for initialization and mainly the update procedure 5.8 of the AVNS. All the other procedures have not been altered. The principle of this backtrack procedure is to return back on some already visited solutions that we judge promising for one or more reasons. The new algorithm is called the AVNSBT. Figure 9 depicts the slight change in the structure of the AVNS with the introduced backtrack steps with a zoom on the update procedure. To implement this procedure, we need to define the two following new variables:

$Threshold$ **:** a number of iterations after which the backtrack is performed ($Threshold$ should be a divisor of $ITER$).

$L_{threshold}(H_{threshold})$ **:** a threshold solution belonging to the hoist number $H_{threshold}$ to which the search procedure returns back every $Threshold$ iterations.

The threshold solution $L_{threshold}$ is the best solution that has ever been registered for a given number of hoists, $H_{threshold}$. When we consider more than one hoist number for the backtrack procedure, the threshold solution is then the best solution that has ever been registered for the hoist number that has the oldest improvement. The backtrack procedure occurs as follows (see Figure 9). First, we initialize the $Threshold$ variable in the initialization step. Then, throughout the iterations, although always assign the best neighboring solution $(L_{neigh})_{best}$ to the incumbent solution L_{inc} (procedure 5.8 of AVNS), a test is performed beforehand: If the search has gone through $Threshold$ iterations ($ITER$ is divisible by $Threshold$), then the incumbent solution L_{inc} must return back to the threshold solution $L_{hreshold}$, otherwise, the incumbent solution gets the best neighboring solution $(L_{neigh})_{best}$.

7 Computational Experiments

Our general solving algorithm, AVNS, was implemented in Matlab (R2016a) and runned on an Intel(R) Core(TM) i5-6500 CPU with 3.20 GHz. The MILP was solved thanks to the "Hybrid Toolbox" [5] which is a MATLAB/Simulink toolbox used for modeling, simulating, and verifying hybrid dynamical systems.

7.1 Benchmark problems

The AVNS was tested on three benchmark problems taken from the literature. The first one, well-known, is the benchmark of Phillips and Unger [41] that we denote by "P&U". This benchmark was mainly used to adjust the different parameters of the algorithm This first benchmark is a 13-sized-problem-case where tank 1 is an associated loading-unloading station. The related data are presented in the appendix (Table A1 gives the empty move times $d_{i,j}$ defined in seconds and shows per tank number, the lower and upper bounds m_i and M_i of each processing task and transport travel time r_i). The second and the third benchmarks were proposed by Manier [33] and labelled "ligne1" and "ligne2". The "ligne1" example is also a 13-sized-problem-case with tank 1 as associated loading-unloading station. This problem example includes a circulation constraint that was explained by Shapiro [44]:

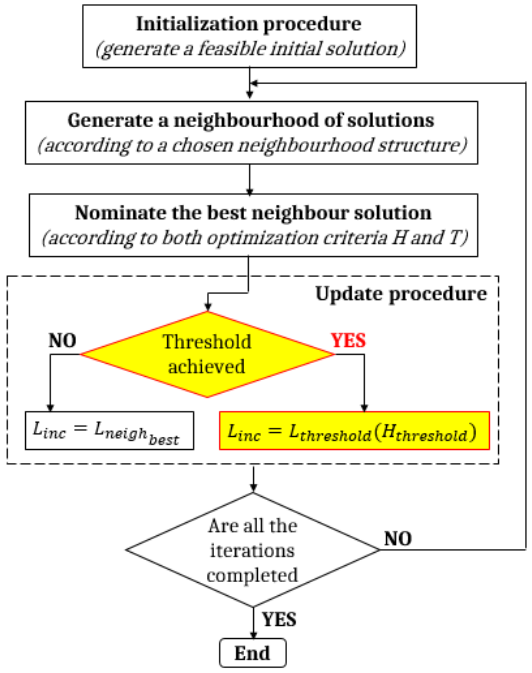

Figure 9: The AVNS algorithm with backtrack

The same carrier unloaded in a cycle, is loaded again to get in the line at the next cycle. As the MILP formulation used here to evaluate the solutions does not take into account this constraint, we were obliged to slightly modify the data, provided also in Manier and Lamrous [36]: rather than m_i(tank 1, loading operation)$= 180s$ and m_i(tank 1, unloading operation)$= 180s$, we put m_i(tank 1, loading operation)$= 180 + 180 = 360s$ as a sum of the two minimal times and m_i (tank 1, unloading operation)$= 0s$ to oblige considering that the two minimal times correspond to the same carrier. The "ligne2" example is a 15-sized-problem-case with dissociated loading-unloading stations (loading station is tank 1 and the unloading one is tank 15). Here, every unloaded carrier must be returned by a hoist to the loading station. The data related to benchmarks "ligne1" and "ligne2" are provided in the appendix with Tables A2 and A3. These two benchmarks were tested to validate the parameter set already chosen at the first step and to prove the efficiency of the proposed algorithm AVNS on solving different benchmarks.

Table 2: Benchmarked results for "P&U" example

number of hoists (H)	optimal of the literature T^*_{best}	"M&L" results T_{best}	T_{mean}
1	521	665	673.8
2	251	332	346
3	—	196	205.8
4	—	210	226.4
5	—	151	159.4
6	—	—	—
7	—	151	151

7.2 First results and parameter adjustment

The results depicted in this section belong to tests that have been carried on "P&U" benchmark. We have compared the results of our AVNS to the best results obtained on this benchmark. In the following tables, T^*_{best} represents the optimal cycle time of the literature, without considering the collision constraints.

In the literature, only best cycle time on that benchmark exits for one or two hoists ($H = 1$ or $H = 2$), denoted resp. $T^*_{best}(1)$ and $T^*_{best}(2)$. $T^*_{best}(1)$ has been proven optimal by Shapiro and Nuttle [44] ($T^*_{best}(1) = 521\,s$). The second one has been provided by Lei and Wang [25] ($T^*_{best}(2) = 251\,s$) by using a heuristic approach. Their resolution is collision free by solving the single hoist problem respectively on two parts of contiguous stations. The result is the minimum common cycle (MCC) of two hoists. In the best of our knowledge, it does not exist any optimal resolution for the CHSP for $H > 2$. On the other hand, Manier and Lamrous have proposed a fundamental basis of comparison for our work as we tackled is the same problem with the same three criteria [36]. Recall that the objective of this study (see Section 4) is to improve the CHSP problem using the empty-move based encoding initially proposed in [35] and [36]. Their results are denoted by "M&L" in the following. Table 2 presents the last results from the literature. For "M&L" results, we exhibit the best cycle time T_{best} with their solving method and the mean best cycle time T_{mean} for $H = 1, ..., 7$.

First, main parameters of the method have to be adjusted one after the other. First the two choices explained in Section 5 are determined by calculation. The first belongs to the election procedure, the best neighbor solution based on the two variables H and T is nominated and applied to the second election option. The latter privileges the minimization

Table 3: Influence of the first and second election options on results for "P&U" benchmark

H	T-option		H-option	
	T_{best}	T_{mean}	T_{best}	T_{mean}
1	1202	1361.6	733	775.6
2	434	498.2	324	359.1
3	214	223.7	204	209.9
4	151	158.7	151	165
5	151	151	151	151
6	151	151	151	151.8
7	151	151	201	290.8

Table 4: Influence of the incumbent solution change options on results for "P&U" benchmark

H	T-improvement-option		T-non-improvement-option	
	T_{best}	T_{mean}	T_{best}	T_{mean}
1	776	1014	733	775.6
2	338	453.4	324	359.1
3	210	236.4	204	209.9
4	163	171.5	151	165
5	151	156.9	151	151
6	151	153.6	151	151.8
7	151	899.6	201	290.8

of the number of hoists H rather than the cycle time T, to find the best neighbor solution. Therefore, Table 3 gives the results of the AVNS with the two election procedure options, minimizing T and H, respectively denoted by $T-option$ and H-option. Both best and mean found cycle times (resp. T_{best} and T_{mean}) over five runs are also presented using a maximum number of iterations $ITER = 10\,000$ and a neighborhood size $S = 20$. One can remark that the second election option H-option outperforms the first one, mainly on smallest number of hoists, not only on best cycle time T_{best} but also, on the mean best cycle time T_{mean}. The H-option is less performing with $H = 7$ and on T_{mean} with $H = 4, 6$ and 7. This result is explained by the fact that the research is oriented towards minimizing the number of hoists, and can be improved with other parameter values, mainly when the number of iterations increases.

The second choice, motivated by numerical results, is to change the current solution L_{inc} by $(L_{neigh})_{best}$ every iteration of the algorithm, even if T_{inc} is not improved (see Section 5.8). Table 4 shows the results of AVNS in two cases: the first option (T-improvement-option) by replacing L_{inc} with $(L_{neigh})_{best}$ if $T_{best}(h_{neigh})_{best}$ is improved and by $(L_{neigh})_{best}$ in all cases with the second option (T-no-improvement-option). The first option supports the intensification of the search (in-depth research) whereas the second reinforces the diversification of the search (multiple hollow research). Likewise, the best found cycle time T_{best} and the mean best cycle time T_{mean} over 5 tests with the same AVNS set of parameters are shown. When the T-non-improvement-option is applied in the AVNS, the results are much better on T_{best} and T_{mean}. When H is increasing (e.g., $H = 7$) T_{best} is a little bigger than when using T-improvement-option (as we apply the H-option here). Nonetheless, with a larger number of iterations ($ITER > 10\,000$), the algorithm has reached optimal values even for the highest number of hoists. So more than intensification, a high level of diversification is mandatory to access to the best solutions. Morover, the multiple-hollow research seems to be the suitable search strategy for our problem. In the following the two best elected options (*i.e.*, H-option and T-non-improvement-option) are kept in the AVNS.

Let's now show the influence of the number of iterations on the results. To do that all other parameters remain constant ($S = 20$) and $ITER$ takes the values 100, 1000, 10 000, 100 000 and 1 000 000. Table 5 summarizes these findings with T_{best} and T_{mean} over five tests, except for $ITER = 1\,000\,000$ with only two tests because of too large computation times. The last line of that table provides the mean CPU time (in seconds) over the test repetitions for each class of $ITER$. Recall that each $T_{best}(H)$ begins with the value 2000 as an upper bound. These results show that the bigger the number of iterations, the better the cycle time for each the number of hoists H, not only the best one over the different tests T_{best} but also, the mean best T_{mean}. Except for $ITER = 1\,000\,000$, the T_{best} is not better than T_{best} of $ITER = 100\,000$. That can be explained by the lack of test repetitions for $ITER = 1\,000\,000$. The small improvement of T_{mean} for $ITER = 1\,000\,000$ compared to T_{mean} for $ITER = 100\,000$ can be a result of the growth of the number of iterations but also the lack of the test repetitions. Figure 10 plots these results with T_{best} and T_{mean} for the different numbers of hoists over the five proposed values of $ITER$ and reveals that $ITER = 100\,000$ either outperforms the other tested values or is as performing as $ITER = 1\,000\,000$. As for the CPU time, it also growths with the number of iterations $ITER$. As a result, as the improvement is not significant comparing the results for $ITER = 100\,000$ and 1 000 000. The second one needs more computational time (that becomes unacceptable). So, we decide to choose the 100 000 as the best $ITER$

Table 5: Influence of the number of iterations $ITER$ on results for "P&U" benchmark

H \ $ITER$	100		1000		10000		100000		1000000	
	T_{best}	T_{mean}	T_{best}	T_{mean}	T_{best}	T_{mean}	T_{best}	T_{mean}	T_{best}	T_{mean}
1	1147	1191	918	981.8	733	775.6	563	715.6	697	715
2	479	534.2	358	409.6	323.5	359.1	253	282.2	264	266.8
3	223	305.9	215	249.5	204	209.9	183	197.5	192	194.5
4	178	236.9	164	191	151	165	151	157.9	151	151
5	160	917.3	152.5	174.8	151	151	151	151	151	151
6	170	1634	162	255	151	151.8	151	151	151	151
7	2000	2000	215	1643	201	290.8	151	151	151	151
CPU time (s)	—	9.336	—	93.176	—	1038.55	—	10237.5	—	123560

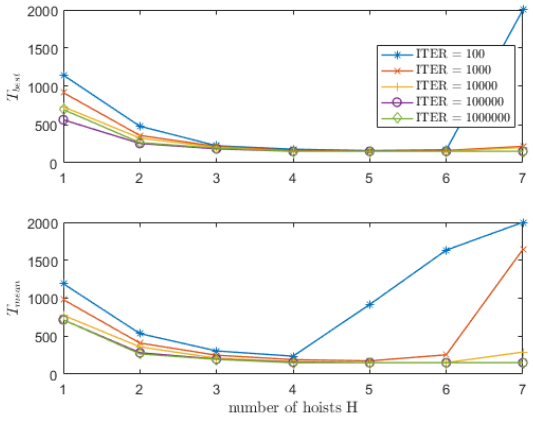

Figure 10: T_{best} and T_{mean} under the five tested values of $ITER$ on "P&U" benchmark

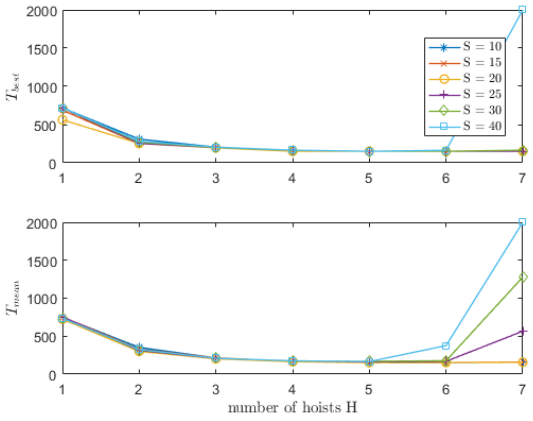

Figure 11: T_{best} and T_{mean} under the six tested values of S on "P&U" benchmark

value. Its mean CPU time is equivalent to 2.8 h per test. Until now, this is an acceptable computational time regarding the great size of the research space due to the problem complexity. Moreover, the CHDSP we tackle in this paper is considered in an offline context. It means that the problem is solved once when a layout or reconfiguration of an electroplating facility is required. So, the CPU time is not a critical factor for us.

The neighborhood size S is another determinant parameter to adjust for the smooth running of AVNS. Tests were conducted successively with values 10, 15, 20, 25, 30, 40 for S without changing the other parameters ($ITER$ = 100 000, with the two already elected options). Table 6 provides the obtained results (T_{best} and T_{mean}) over 10 runs for every test class. The last line gives the mean CPU time (in seconds). The two values of 2 000 mean that there were no explored solutions corresponding to neighborhood size $S = 40$ and number of hoists $H = 7$. One can remarks that the best results are obtained with a size $S = 20$ for T_{best} and T_{mean}. Figure 11 plots the same results and confirms that $S = 20$ assures the best cycle times. Indeed, the larger is the size S, the higher is the CPU time, because more neighbor solutions are generated at each iteration. The cycle times do not get better from $S = 20$ because of the randomness of the search. Hence, $S = 20$ is considered the appropriate value when considering both the quality of solutions and the CPU times.

Thanks to the previous adjustment phase, $S = 20$ and $ITER$ = 100 000 is the best choices for H-option and T-non-improvement-option as parameters for AVNS. Moreover, the algorithm has been improved to lead even better results. Indeed, the research of new solutions in the iterative process of the algorithm gave a significant number of solutions that had been explored yet because of the usage of a circular shift neighborhood structure (N_{shift}). As a

Table 6: Influence of the neighborhood size S on the results for "P&U" benchmark

H \ S	10		15		20		25		30		40	
	T_{best}	T_{mean}	T_{best}	T_{mean}	T_{best}	T_{mean}	T_{best}	T_{mean}	T_{best}	T_{mean}	T_{best}	T_{mean}
1	718	750	689	754.8	563	723.2	718	755.1	718	736.5	718	736.3
2	310	352.2	253	330.3	253	299.7	254	312.2	270	320.7	282	324.6
3	205	213.4	207.5	214	196	202.4	200	208.6	200	210.7	205	209.4
4	162.5	171.7	151	170.3	151	163.9	165	175.6	164	173.5	165	175.2
5	151	152.1	151	152.9	151	151.6	151	161	151	168.6	151	165.6
6	151	151	151	153.3	151	153.2	151	168.7	151	176.7	164	375.7
7	151	157.3	151	159.5	151	156	151	565.5	165.33	1278	2000	2000
CPU time (s)	—	4289.02	—	6197.56	—	8497.57	—	9691.96	—	11791.78	—	15461.14

Table 7: Results of the AVNS on "P&U" benchmark after deletion of the shift neighborhood structure

H	With N_{shift}		without N_{shift}	
	T_{best}	T_{mean}	T_{best}	T_{mean}
1	563	723.2	521	701.5
2	253	299.7	251	257.3
3	196	202.4	185.3	191.6
4	151	163.9	151	151.7
5	151	151.6	151	151
6	151	153.2	151	151
7	151	156	151	151

result, the circular decoding step generated already explored move sequences. Table 7 clearly exhibits better results when using this new version of AVNS over ten test repetitions.

7.3 Comparison with benchmarked results

Here, we should compare the obtained results to those of past researches to evaluate the performance of our proposed AVNS. We begin with "P&U" benchmark results. Table 8 gathers the best AVNS results on this benchmark together with optimal results of the literature T^*_{best} and "M&L" results. It also provides the gaps between the best cycle times T_{best} and the mean best cycle times T_{mean} of both "M&L" and AVNS, denoted by gap_{best} and gap_{mean}, such as:

$$gap_{best}(H) = 100\left(\frac{T_{best\_"M\&L"}(H) - T_{best\_AVNS}(H)}{T_{best\_"M\&L"}(H)}\right) \quad (1)$$

$$gap_{mean}(H) = 100\left(\frac{T_{mean\_"M\&L"}(H) - T_{mean\_AVNS}(H)}{T_{mean\_"M\&L"}(H)}\right) \quad (2)$$

Interestingly, these results indicate that AVNS was able to find the optimal cycle times given in literature $T^*_{best}(1) = 521\,s$ and $T^*_{best}(2) = 251\,s$ for one and two hoists. Moreover, it outperforms the "M&L" algorithm on the best cycle time T_{best} with a positive mean gap of 13.26% and on the mean best cycle time T_{mean} with a positive mean gap of 11.06%. Important improvement has only concerned the results of the small number of hoists ($1 \leq H \leq 4$), because for $5 \leq H \leq 7$, "M&L" has already found the best results. Indeed, for the smaller number of hoists, the hoists still are the critical resources of the line as the throughput of the line depends on the efficiency of the hoists. But, for bigger numbers of hoists, the tanks become the critical resources of the line instead of the hoists, and because they influence the cycle time with their restrictive soaking-time bounds, the critical tank of the line will be the one that has the biggest minimal soaking time. In the case of "P&U"'instance, the bottleneck resource is tank number 2 (see appendix in Table A1). It has the biggest minimal soaking time $m_i = 150$. The cycle time then stagnates at $T = 151$ even if H increases. "M&L" found the best cycle time $T_{best} = 151$ for number of hoists $H \geq 5$, whereas, AVNS provides the same best cycle time for number of hoists $H \geq 4$. AVNS was only less performing on the mean best cycle time $T_{mean}(1)$ of hoist number $H = 1$. This can be explained by the fact that the solutions with one hoist ($H = 1$) are not abundant in a huge research space and the separator based decoding has increased the size of this space. Hence, AVNS should be further improved to overcome this shortage so that to be able to reach more solutions for one hoist. However, AVNS was clearly able to improve the results compared to those of "M&L" and to reach the optimal solutions. This positive outcome proves the advantage of the separator based encoding on exploring the huge search space and upholds the positive effect of the encoding improvement approach proposed in this study. It enabled to discover more zones of the search that were not achievable with the empty move based encoding without separa-

Table 8: Results with both variants of AVNS for the all the tested benchmarks

Benchmark		OPT	"M&L"		AVNS				AVNSBT			
	(H)	T^*_{best}	T_{best}	T_{mean}	T_{best}	T_{mean}	gap_{best}	gap_{mean}	T_{best}	T_{mean}	gap_{best}	gap_{mean}
"P&U"	1	521	665	673.8	**521**	706.6	21.65	-4.86	**521**	**656.2**	0	7.13
	2	251	332	346	**251**	**255.9**	24.39	26.04	**251**	**252.2**	0	1.44
	3	—	196	205.8	**185.3**	**191.1**	5.45	7.14	**182**	**189.5**	1.78	0.83
	4	—	210	226.4	**151**	**152.1**	28.09	32.81	**151**	**151.7**	0	0.26
	5	—	151	159.4	151	**151**	0	5.26	151	**151**	0	0
	6	—	—	—	**151**	**151**	—	—	**151**	**151**	0	0
	7	—	151	151	151	151	0	0	151	151	0	0
mean							13.26	11.06			0.25	1.38
"ligne1"	1	425	452	677	**428**	**498.2**	5.31	26.41	**425**	**480.4**	0.7	3.57
	2	—	361	402	361	**361**	0	10.19	361	**361**	0	0
	3	—	361	361	361	361	0	0	361	361	0	0
	4	—	361	361	361	361	0	0	361	361	0	0
	5	—	361	361	361	361	0	0	361	361	0	0
	6	—	361	361	361	361	0	0	361	361	0	0
	7	—	361	361	361	361	0	0	361	361	0	0
mean							0.75	5.22			0.1	0.51
"ligne2"	1	712	858	1294	**712**	**772.4**	17.01	40.31	**712**	**731.1**	0	5.34
	2	—	661	662	661	**661**	0	0.15	661	**661**	0	0
	3	—	661	661	661	661	0	0	661	661	0	0
	4	—	661	661	661	661	0	0	661	661	0	0
	5	—	661	661	661	661	0	0	661	661	0	0
	6	—	661	661	661	661	0	0	661	661	0	0
	7	—	678	678	**661**	**661**	2.50	2.50	**661**	**661**	0	0
mean							2.79	6.13			0	0.76

tors. That also proves the high performance of the AVNS algorithm to reach the optimal solutions and to provide better results for the different numbers of hoists. For this benchmark, we provide in Figure 12 a schedule scheme of an optimal solution for two hoists.

The AVNS algorithm was also tested on two other benchmarks "ligne1" and "ligne2" to validate the parameter set already chosen with "P&U" benchmark. In the literature, for these two benchmarks, we only find the best cycle time T_{best} for the cyclic hoist scheduling problem (CHSP) with only one hoist ($H = 1$). We denote it by $T^*_{best}(1)$. We also compare our results to "M&L" results for these two benchmarks. Hence, we present in Table 8 the last mentioned results for benchmarks "ligne1" and "ligne2" together with results got from the AVNS. We still exhibit results in best cycle time T_{best} and mean best cycle time T_{mean}, for the number of hoists H where $1 \leq H \leq 7$. We also provide the two gaps gap_{best} and gap_{mean} computed by Equations eq1 and (2).

On whole, Table 8 presents better results got from AVNS. For benchmark "ligne1", AVNS improves the results obtained by "M&L" approach on the best cycle time T_{best} by a positive mean gap of 0.75% and on the mean best cycle time T_{mean} by a positive mean gap of 5.22%. Yet, we observe improvement only for one and two hoists, as for number of hoists higher than 2, "M&L" has already detected the best results. In fact, when the number of hoists exceeds one, the best cycle time T_{best} stagnates. From $H = 2$ hoists, the critical resource for benchmark "ligne1" is tank 1 with $m_1 = 360$ (see appendix in Table A2). And then, for $H \geq 2$, the best cycle time is $T_{best} = 361$. For one hoist, AVNS was able to reach a solution with a best cycle time $T_{best} = 428\,s$, so close to the optimal $T^*_{best}(1) = 425\,s$ (gap 0.7 %). For this benchmark, we provide in Figure 13 a schedule scheme of an optimal solution for two hoists.

As for benchmark "ligne2", AVNS outperforms the "M&L" algorithm on the best cycle time T_{best} with a positive mean gap of 2.79% and on the mean best cycle time T_{mean} with a positive mean gap of 6.13%. Improvement can be observed for 1, 2 and 7 hoists. For $3 \leq H \leq 6$, "M&L" has already found the best results. Indeed, T_{best} stagnates from number of hoists $H = 2$, as tank 14 becomes the critical resource of "ligne2", with $m_14 = 660$ (see see appendix in Table A3) and the best cycle time for $H \geq 2$ is $T_{best} = 661$. AVNS improves the results for one hoist on the best cycle time T_{best} with a positive gap equal to 17.01% and on the mean best cy-

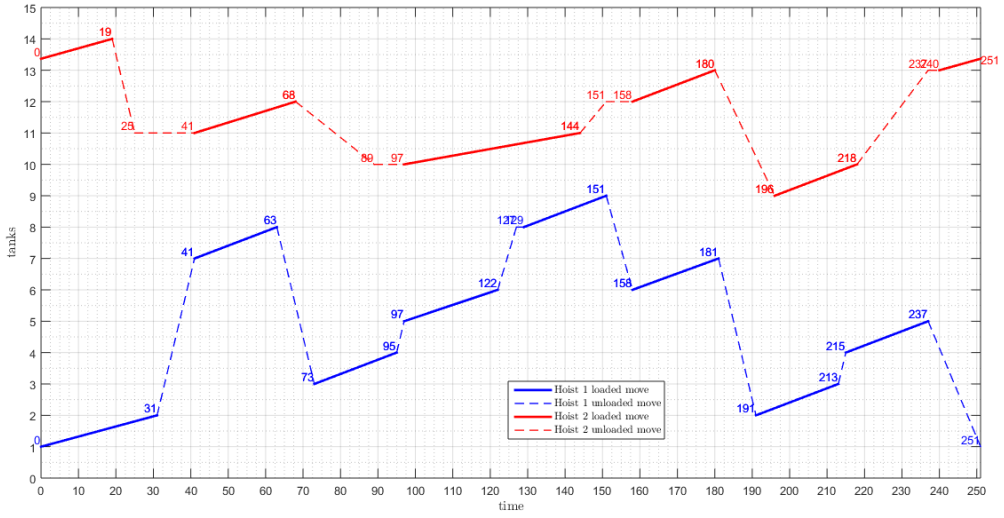

Figure 12: An optimal cyclic 2-hoist schedule of a solution of cycle time T=251s from "P&U" benchmark

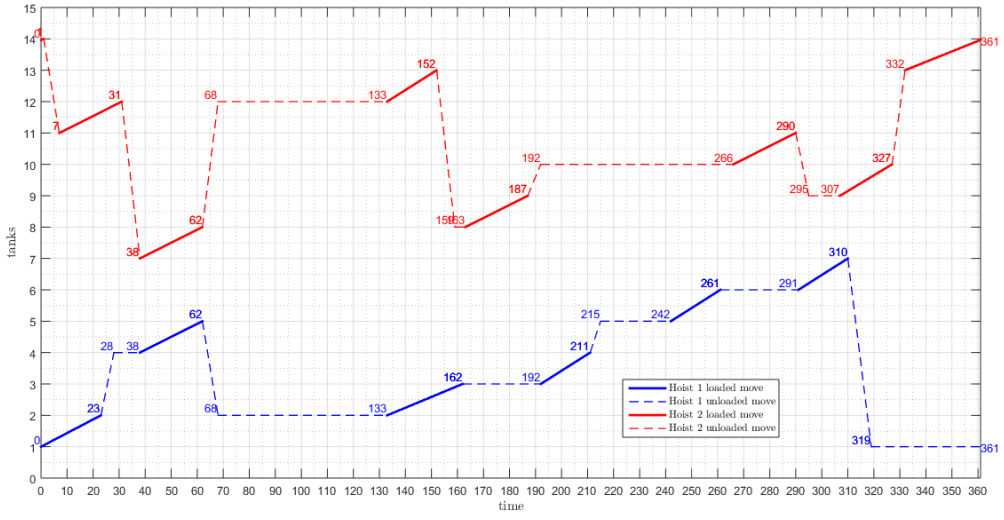

Figure 13: An optimal cyclic 2-hoist schedule of a solution of cycle time T=361s from "ligne1" benchmark

cle time T_{mean} with a positive gap equal to 40.31%. Indeed, it was able to detect the solution of the optimal cycle time of the literature $T^*_{best}(1) = 712\,s$.

Accordingly, we can first confirm the positive effect of the original encoding approach that was proposed in this study. Indeed, this empty move based encoding improved with separators has enabled to reach more areas in the search space that were not detectable with the encoding of solutions without separators. The new encoding has added more varied solutions to the search space, that is, it has made them visible. Therefore, it has enhanced the results by making possible to reach better solutions, even optimal ones. At the same time, it has made harder the search procedure because the number of solutions has been multiplied. Thus, that required an efficient algorithm to perform the search and avoid falling into local minima. From that point, we can second confirm the choice of the parameter set already fixed with "P&U" benchmark, as AVNS with these adjusted parameters was able to bring improvement on the results for the three tested benchmarks. As a consequence, the obtained results show the efficiency of AVNS in providing better results and even the optimal ones for different number of hoists. That indicates that AVNS was robustly constructed, either for the choice of parameter values and options or for its ordered steps and procedures.

7.4 Better version algorithm AVNSBT

Hereafter, we will show the further performance of the AVNS algorithm with the integrated backtrack procedure. For the benchmark "P&U", the AVNSBT provided the best results with a threshold of 100 iterations ($Threshold = 100$) where the backtrack has considered the best registered solutions of 1 and 2 hoists ($L_{threshold}(1)$ and $L_{threshold}(2)$ as $H_{threshold} \in \{1, 2\}$). For benchmarks "ligne1" and "ligne2", the best tested threshold is also 100 iterations and the backtrack has only considered the best registered solutions for 1 hoist ($L_{threshold}(1)$ as $H_{threshold} = 1$). Then, we gather in Table 8, on right side, the obtained results for the three tested benchmarks. We also provide the gap between the best cycle times T_{best} of both AVNS and AVNSBT and the gap between the mean best cycle times T_{mean}, denoted also by gap_{best} and gap_{mean}.

It is clear from the results that AVNSBT algorithm somewhat outperforms the AVNS algorithm. This first confirms the earnest effect of the integrated backtrack procedure to improve the outcome of the AVNS. Amelioration has mainly concerned the results of one hoist. Indeed, the gap between the mean best cycle times of AVNS and AVNSTB, gap_{mean}, for $H = 1$ is of 7.13% for the benchmark "P&U", 3.57% for benchmark "ligne1" and 5.34% for benchmark "ligne2". Moreover, the improvement has mostly affected the results of "P&U" example, with a positive gap_{mean} for 1 to 4 hoists. As for the best cycle times, AVNSBT has only make improvement on T_{best} for hoist number $H = 3$ for "P&U" example and for hoist number $H = 1$ for "ligne1", where AVNSBT reaches the optimal solution. Otherwise, most AVNSBT upgrade has affected the mean cycle times T_{mean} which means that the backtrack procedure has enhanced the robustness of the AVNS algorithm. It enabled to reach more best solutions than AVNS. However, it cannot further boost the best cycle times whose optimal values were already reached by AVNS. Therefore, AVNSBT has proved its effectiveness to reach better solutions. It interestingly remedies the AVNS shortage to reach better solutions for one hoist. As a consequence, AVNS, with the new separator based encoding and the well set parameters, was able to cope with the browsing challenge of the huge search space in order to reach so promising solutions. Together with its backtrack procedure, it was able to optimize the search to further enhance the findings, mainly the algorithm robustness.

7.5 Further experiments

In the previous two paragraphs, we have shown the advantage of separator based encoding over the encoding without separators, while comparing both variants of AVNS with "M&L" approach. We have also shown the advantage of the backtrack procedure, added in our AVNS. As we previously explained, the difference between our AVNS and the "M&L" approach relies on both the encoding procedure and the solving method. To confirm the interest of our AVNS, we have modified the Genetic Algorithm (GA) proposed in "M&L", in which we have implemented the same empty-move based encoding with separators [19]. We call it SGA. Hence, we compare the performance of the last version of AVNS algorithm (AVNSBT) and the SGA approach.

Moreover, in order to better highlight the interest of our solving method, we have extended our tests to five additional benchmarks. Apart from "P&U", "ligne1" and "ligne2", we have considered the benchmarks used in Leung et al. [27]. These are "Mini-Phil", and 4 instances taken from Shapiro et al. [44] and modified by Leung et al. [27], as they

do not consider the duplicated tanks in the original instances. We denote them "BO1-v2", "BO2-v2", "Copper-v2" and "Zinc-v2". "Mini-Phil" is the same problem of "P&U" but truncated to the first eight tanks.

In Table 9 and Table 10, we show the results obtained with algorithms SGA and AVNSBT, respectively for the benchmarks "P&U", "ligne1" and "ligne2", and for the five new considered instances "Mini-Phil", "BO1-v2", "BO2-v2", "Copper-v2" and "Zinc-v2". We also provide gap_{best} which is the gap between the best cycle times T_{best} of both SGA and AVNSBT, and gap_{mean} which is the gap between the mean best cycle times T_{mean} of SGA and AVNSBT. Note that in Table 10, we give the optima over one hoist found in [27], and the $T_{best}(H)$ for the SGA algorithm begin with the value 4000 as an upper bound.

Compared with the "M&L" results in Table 8, we can first observe in Table 9 that SGA finds the optimal cycle time with a single hoist for the benchmarks "P&U" and "ligne2", which were not reachable by "M&L". Nevertheless, the mean cycle time T_{mean} remains slightly better with the initial Genetic Algorithm, at least for 1 and 2 hoists.

Evaluating the results of AVNSBT, we can observe that it was able to find the optimal solutions with one hoist ($H = 1$) for 6 out of the 8 tested benchmarks (for all of them except "Mini-Phil" and "BO1-v2"). For these two last instances, the gaps with the optimal cycle time are respectively 21% and 11.25%. For two hoists ($H = 2$), our algorithm AVNSBT finds the optimal solutions for all the benchmarks, except "BO1-v2" for which T_{best}=275.65s (3.67% from T^*_{best}). For $H \geq 3$ and for the 5 last benchmarks, the values of T_{best} are the same for all the numbers of hoists: for instance "Mini-Phil", T_{best}=151s is the same value found on the original benchmark "P&U" for $H = 4$, which means that for "Mini-Phil" and with more than 2 hoists, the transportation resources are no longer the critical resources of the line, but the tanks; it is the same for the other benchmarks, as soon as the cycle time practically reaches the biggest minimal soaking time (240 s for "BO1 and 2", 1800 s for "Copper-v2" and 1680 s for "Zinc-v2"), as previously explained. Note that for instance "BO1-v2", we have not referenced optimal values for $H \geq 3$, but we observe that the value of T_{best}=247.3s stagnates. It probably also means that the tanks are the critical resources of the line. Hence, all these results show the performance of our proposed algorithm and outline its ability to solve problems of different sizes. As for the mean best cycle times T_{mean}, AVNSBT finds the same T_{best} for most of the benchmarks and hoists'number (including with $H = 1$ for "Copper-v2" and "Zinc-v2"), and near T_{best} for other ones, which shows again the robustness of our algorithm over the tested instances. Even if it does not find the optimal solutions for benchmarks "Mini-Phil" and "BO1-v2" for $H = 1$ or 2, it was able to keep its stability over the tests for all the number of hoists as the T_{mean} is either equal or near to the best values T_{best}. For the cases where it does not find the optimal solutions, it is almost the case for one hoist, and this can be explained by two facts: the first obvious one is because we employ an approximate solving method, and the second one is because the solutions for one hoist are so rare in a search space all the bigger as it gathers solutions for different numbers of hoists.

Comparing now the results of AVNSBT over those of SGA for the eight tested instances, as it can be seen in Tables 9 and 10, we obtain better results mainly on the lowest number of hoists. It is also obvious from the positive gaps that AVNSBT enhances the results over the eight benchmarks. Indeed, gap_{best} varies from 0 to 42.68% for one hoist ($H = 1$), with a mean improvement of 13.64%, whereas gap_{mean} varies from 1.83% to 53.44%, with a mean improvement of 29.3%. Improvements are more important for the mean cycle times T_{mean} and that is an element which shows the robustness of our algorithm AVNSBT. It is able to discover more better quality solutions than SGA. Improvements over the best cycle times T_{best} outline the ability of AVNSBT to reach more the best solutions over the search space and show that the search trajectory is well guided.

Overall, even if SGA was able to detect sometimes the best solution for a given number of hoists, it was not able to keep the same level of performance over the executions of the algorithm, the challenge on which AVNSBT has proven the aptitude. Indeed, AVNSBT has shown its ability to reach a bigger number of near-optimal solutions over the runs and over the tested benchmarks.

8 Conclusion

In this paper, we have studied the cyclic hoist scheduling problem within multi-hoist electroplating lines with both dimensions of design and scheduling. Almost all previous researches have assumed that the design of the line is already done and the number of hoists has been fixed. However, in our study, we consider the size of the mate-

Table 9: Comparison between AVNSBT and SGA for the first set of instances

Bench-mark	H	OPT T^*_{best}	SGA T_{best}	SGA T_{mean}	AVNSBT T_{best}	AVNSBT T_{mean}	AVNSBT gap_{best}	AVNSBT gap_{mean}
P&U	1	521	521	1024	521	**656.2**	0	35.91
	2	251	264	360	**251**	**252.2**	4.92	29.94
	3	—	192.33	231.5	**182**	**189.5**	5.37	18.14
	4	—	163	207	**151**	**151.7**	7.36	26.71
	5	—	151.33	154.14	**151**	**151**	0.21	2.03
	6	—	151	—	151	**151**	0	—
	7	—	151	151	151	151	0	0
mean							2.55	16.10
ligne1	1	425	492	1032	**425**	**480.4**	13.61	53.44
	2	—	361	404	361	**361**	0	10.64
	3	—	361	361	361	361	0	0
	4	—	361	361	361	361	0	0
	5	—	361	361	361	361	0	0
	6	—	361	361	361	361	0	0
	7	—	361	361	361	361	0	0
mean							1.94	9.15
ligne2	1	712	712	1332	712	**731.1**	0	45.11
	2	—	661	662	661	**661**	0	0.15
	3	—	661	661	661	661	0	0
	4	—	661	661	661	661	0	0
	5	—	661	661	661	661	0	0
	6	—	661	661	661	661	0	0
	7	—	661	661	661	661	0	0
mean							0	6.46

rial handling resources as a decision variable. The dual problem is called the Cyclic Hoist Design and Scheduling Problem (CHDSP). The optimization objective, hence, is to determine an optimal cyclic schedule for each possible number of hoists that minimizes the cycle time and thus maximizes the line throughput rate. We have proposed an original empty-move based encoding method with separators. It both assigns a number of hoists to the line and generates their move sequences. We have used the mixed integer linear programming model developed in [36] to evaluate the move sequences to check their feasibility and to compute the scheduled move times together with the cycle time. We have developed an AVNS, which is a rich and structured solving algorithm based on variable neighborhood search, and we have adapted it to solve efficiently the bi-objective problem. It affords decision-makers a decision support system to enable them to choose the number of material handling resources required for the production line, while regarding investment costs as well as productivity objectives. The proposed AVNS was then highly performing and provided valued results compared to previous ones, reaching the optimal solutions for most cases. Enhanced with a backtrack procedure, it allowed to reach even better solutions and further optimized the search, which confirms its robustness.

Many outlooks of this work can be suggested. We could further improve the AVNS performance so that it reaches more solutions for one hoist. Likewise, we could implement the search procedure in parallel environment so that the search in each possible number of hoists goes separately and simultaneously. That would have an advantage on the computational times. As for theoretic basis, we would complete our approach by studying the collision free constraints, as we deal with the multi-hoist case.

Acknowledgments

This work was supported in part by the EIPHI Graduate school (contract "ANR-17-EURE-0002").

References

[1] Armstrong, R., Gu, S., Lei, L., 1996. A greedy algorithm to determine the number of transporters in a cyclic electroplating process. IIE transactions 28, 347–355.

[2] Armstrong, R., Lei, L., Gu, S., 1994. A bounding scheme for deriving the minimal cycle time

Table 10: Comparison between AVNSBT and SGA for the second set of instances

Benchmark	H	OPT	SGA		AVNSBT			
		T^*_{best}	T_{best}	T_{mean}	T_{best}	T_{mean}	gap_{best}	gap_{mean}
Mini-Phil	1	281	340	346.36	340	340	0	1.83
	2	165.5	165.5	166.23	165.5	165.5	0	0.44
	3	—	151	153.18	151	151	0	1.42
	4	—	151	151	151	151	0	0
	5	—	151	151	151	151	0	0
	6	—	151	151	151	151	0	0
	7	—	151	151	151	151	0	0
BO1-v2	1	299.5	460,4	460,4	333.2	333.95	27.63	27.46
	2	265.9	275.65	285.29	275.65	275.65	0	3.38
	3	—	247.3	247.3	247.3	247.3	0	0
	4	—	247.3	247.3	247.3	247.3	0	0
	5	—	247.3	247.3	247.3	247.3	0	0
	6	—	247.3	247.3	247.3	247.3	0	0
	7	—	247.3	247.3	247.3	247.3	0	0
BO2-v2	1	279.3	327.6	337.6	279.3	283.27	14.74	16.09
	2	240.1	240.1	240.1	240.1	240.1	0	0
	3	—	240.1	240.1	240.1	240.1	0	0
	4	—	240.1	240.1	240.1	240.1	0	0
	5	—	240.1	240.1	240.1	240.1	0	0
	6	—	240.1	240.1	240.1	240.1	0	0
	7	—	240.1	240.1	240.1	240.1	0	0
Copper-v2	1	1847.2	2063.3	2096.66	1847.2	1847.2	10.47	11.90
	2	1800.1	1800.1	1800.1	1800.1	1800.1	0	0
	3	—	1800.1	1800.1	1800.1	1800.1	0	0
	4	—	1800.1	1800.1	1800.1	1800.1	0	0
	5	—	1800.1	1800.1	1800.1	1800.1	0	0
	6	—	1800.1	1800.1	1800.1	1800.1	0	0
	7	—	1800.1	1800.1	1800.1	1800.1	0	0
Zinc-v2	1	1743.4	3041.6	3041.6	1743.4	1743.4	42.68	42.68
	2	1680.1	1680.1	1738.49	1680.1	1680.1	0	3.36
	3	—	1680.1	1680.1	1680.1	1680.1	0	0
	4	—	1680.1	1680.1	1680.1	1680.1	0	0
	5	—	1680.1	1680.1	1680.1	1680.1	0	0
	6	—	1680.1	1680.1	1680.1	1680.1	0	0
	7	—	1680.1	1680.1	1680.1	1680.1	0	0

of a single-transporter n-stage process with time-window constraints. European Journal of Operational Research 78, 130–140.

[3] Baptiste, P., Legeard, B., Manier, M.A., Varnier, C., 1993. Optimization with constraint logic programming: the hoist scheduling problem solved with various solvers. Application of Artificial Intelligence in Engineering , 599–614.

[4] Baptiste, P., Legeard, B., Varnier, C., 1992. Hoist scheduling problem: an approach based on constraint logic programming, in: Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on, IEEE. pp. 1139–1144.

[5] Bemporad, A., 2004. Hybrid Toolbox - User's Guide. http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox.

[6] Bilge, Ü., Ulusoy, G., 1995. A time window approach to simultaneous scheduling of machines and material handling system in an fms. Operations Research 43, 1058–1070.

[7] Caux, C., Pierreval, H., 1997. Solving a hoist scheduling problem as a sequencing problem. IFAC Proceedings Volumes 30, 315–319.

[8] Che, A., Chu, C., 2004. Single-track multi-hoist scheduling problem: a collision-free resolution based on a branch-and-bound approach. International Journal of Production Research 42, 2435–2456.

[9] Che, A., Chu, C., 2007. Cyclic hoist scheduling in large real-life electroplating lines. OR Spectrum 29, 445–470.

[10] Che, A., Chu, C., 2008. Optimal scheduling of material handling devices in a pcb production line: problem formulation and a polynomial algorithm. Mathematical Problems in Engineering 2008.

[11] Che, A., Lei, W., Feng, J., Chu, C., 2013. An improved mixed integer programming approach for multi-hoist cyclic scheduling problem. IEEE Transactions on Automation Science and Engineering 11, 302–309.

[12] Che, A., Yan, P., Yang, N., Chu, C., 2010. Optimal cyclic scheduling of a hoist and multi-type parts with fixed processing times. International Journal of Production Research 48, 1225–1243.

[13] Chtourou, S., Manier, M.A., et al., 2013. A hybrid algorithm for the cyclic hoist scheduling problem with two transportation resources. Computers & Industrial Engineering 65, 426–437.

[14] Deroussi, L., Norre, S., 2010. Simultaneous scheduling of machines and vehicles for the flexible job shop problem, in: International conference on metaheuristics and nature inspired computing, Djerba Island Tunisia. pp. 1–2.

[15] El Amraoui, A., Elhafsi, M., 2016. An efficient new heuristic for the hoist scheduling problem. Computers & Operations Research 67, 184–192.

[16] Feng, J., Chu, C., Che, A., 2018. Cyclic jobshop hoist scheduling with multi-capacity reentrant tanks and time-window constraints. Computers & Industrial Engineering .

[17] Fleury, G., Goujon, J.Y., Gourgand, M., Lacomme, P., 1996. A hoist sheduling problem, containing a fixed number of carriers, solved with an opportunistic approach, in: CESA'96 IMACS Multiconference: computational engineering in systems applications, pp. 473–478.

[18] Hanen, C., Munier, A., 1994. Periodic scheduling of several hoists, in: Proceedings of the Fourth International Workshop on Project Management and Scheduling, pp. 12–15.

[19] Laajili, E., Lamrous, S., Manier, M.A., Nicod, J.M., 2019. Genetic algorithm based approach for the multi-hoist design and scheduling problem, in: The International Conference on Industrial Engineering and Systems Management, IEEE.

[20] Lacomme, P., Larabi, M., Tchernev, N., 2013. Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Economics 143, 24–34.

[21] Lamothe, J., Thierry, C., Delmas, J., 1996. A multihoist model for the real time hoist scheduling problem, in: CESA'96 IMACS Multiconference: computational engineering in systems applications, pp. 461–466.

[22] Lei, L., 1993. Determining the optimal starting times in a cyclic schedule with a given route. Computers & operations research 20, 807–816.

[23] Lei, L., Armstrong, R., Gu, S., 1993. Minimizing the fleet size with dependent time-window and single-track constraints. Operations Research Letters 14, 91–98.

[24] Lei, L., Wang, T.J., 1989. A proof: the cyclic hoist scheduling problem is np-complete. Graduate School of Management, Rutgers University, Working Paper , 89–0016.

[25] Lei, L., Wang, T.J., 1991. The minimum common-cycle algorithm for cyclic scheduling of two material handling hoists with time window constraints. Management Science 37, 1629–1639.

[26] Leung, J.M., Levner, E., 2006. An efficient algorithm for multi-hoist cyclic scheduling with fixed processing times. Operations Research Letters 34, 465–472.

[27] Leung, J.M., Zhang, G., Yang, X., Mak, R., Lam, K., 2004. Optimal cyclic multi-hoist scheduling: A mixed integer programming approach. Operations Research 52, 965–976.

[28] Li, X., Chan, F.T., Chung, S., 2015. Optimal multi-degree cyclic scheduling of multiple robots without overlapping in robotic flowshops with parallel machines. Journal of Manufacturing Systems 36, 62–75.

[29] Lim, J.M., 1997. A genetic algorithm for a single hoist scheduling in the printed-circuitboard electroplating line. Computers & industrial engineering 33, 789–792.

[30] Liu, C., Zhao, C., Xu, Q., 2012. Integration of electroplating process design and operation for simultaneous productivity maximization, energy saving, and freshwater minimization. Chemical engineering science 68, 202–214.

[31] Liu, J., Jiang, Y., 2005. An efficient optimal solution to the two-hoist no-wait cyclic scheduling problem. Operations Research 53, 313–327.

[32] Liu, J., Jiang, Y., Zhou, Z., 2002. Cyclic scheduling of a single hoist in extended electroplating lines: a comprehensive integer programming solution. Iie Transactions 34, 905–914.

[33] Manier, M.A., 1994. Contribution à l'ordonnancement cyclique du système de manutention d'une ligne de galvanoplastie. Ph.D. thesis. Besançon.

[34] Manier, M.A., Bloch, C., 2003. A classification for hoist scheduling problems. International Journal of Flexible Manufacturing Systems 15, 37–55.

[35] Manier, M.a., Lamrous, S., 2006. Design and scheduling of electroplating facilities, in: the International Conference on Service Systems and Service Management,, IEEE. pp. 1114–1119.

[36] Manier, M.A., Lamrous, S., 2008. An evolutionary approach for the design and scheduling of electroplating facilities. Journal of Mathematical Modelling and Algorithms 7, 197–215.

[37] Manier, M.a., Varnier, C., Baptiste, P., 2000. Constraint-based model for the cyclic multihoists scheduling problem. Production Planning & Control 11, 244–257.

[38] Mao, Y.n., Tang, Q.h., Li, Z.x., Zhang, L.p., 2018. Mixed-integer linear programming method for multi-degree and multi-hoist cyclic scheduling with time windows. Engineering Optimization 50, 1978–1995.

[39] Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Computers & operations research 24, 1097–1100.

[40] Ng, W., Leung, J., 1997. Determining the optimal move times for a given cyclic schedule of a material handling hoist. Computers & industrial engineering 32, 595–606.

[41] Phillips, L.W., Unger, P.S., 1976. Mathematical programming solution of a hoist scheduling program. AIIE transactions 8, 219–225.

[42] Qu, H., Wang, S., Xu, Q., 2017. Simultaneous 2d hoist scheduling and production line design for multi-recipe and multi-stage material handling processes. Chemical Engineering Science 167, 251–264.

[43] Riera, D., Yorke-Smith, N., 2002. An improved hybrid model for the generic hoist scheduling problem. Annals of Operations Research 115, 173–191.

[44] Shapiro, G.W., Nuttle, H.L., 1988. Hoist scheduling for a pcb electroplating facility. IIE transactions 20, 157–167.

[45] Thesen, A., Lei, L., 1990. An expert scheduling system for material handling hoists. Journal of Manufacturing Systems 9, 247–252.

[46] Varnier, C., Baptiste, P., 1995. A clp approach for finding a transition schedule between two cyclic mono-product productions in electroplating facilities, in: International Conference on Industrial Engineering and Production Management, pp. 372–381.

[47] Xu, Q., Huang, Y., 2004. Graph-assisted cyclic hoist scheduling for environmentally benign electroplating. Industrial & engineering chemistry research 43, 8307–8316.

[48] Yih, Y., 1994. An algorithm for hoist scheduling problems. The International Journal of Production Research 32, 501–516.

[49] Yih, Y., Chiu, C., 1993. Incremental learning for hoist scheduling problems in circuit board electroplating lines. ASME-PUBLICATIONS-PED 65, 119–119.

[50] Zhang, Q., Manier, H., Manier, M.A., 2012. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. Computers & Operations Research 39, 1713–1723.

[51] Zhang, Q., Manier, H., Manier, M.A., Bloch, C., 2014. Heuristics for predictive hoist scheduling problems. European Journal of Industrial Engineering 8, 695–715.

[52] Zhao, C., Fu, J., Xu, Q., 2013. Production-ratio oriented optimization for multi-recipe material handling via simultaneous hoist scheduling and production line arrangement. Computers & Chemical Engineering 50, 28–38.

[53] Zhou, Z., Li, L., 2003. Single hoist cyclic scheduling with multiple tanks: a material handling solution. Computers & Operations Research 30, 811–819.

[54] Zhou, Z., Li, L., 2009. A solution for cyclic scheduling of multi-hoists without overlapping. Annals of Operations Research 168, 5–21.

[55] Zhou, Z., Liu, J., 2008. A heuristic algorithm for the two-hoist cyclic scheduling problem with overlapping hoist coverage ranges. IIE Transactions 40, 782–794.

Appendices

See Table A1 for data belonging to "P&U" benchmark, Table A2 for data belonging to "ligne1" benchmark and Table A3 for data belonging to "ligne2" benchmark.

Table A1: "P&U" benchmark data (empty move times $d_{i,j}$ from tank i (line i) to tank j (column j), m_i, M_i and r_i)

$d_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1	0	11	14	16	14	19	22	24	26	29	6	8	10
2	11	0	2	5	2	8	10	13	15	17	10	3	1
3	14	2	0	2	0	5	8	10	13	15	12	6	3
4	16	5	2	0	2	3	5	8	10	13	15	8	6
5	14	2	0	2	0	5	8	10	13	15	12	6	3
6	19	8	5	3	5	0	3	5	7	10	18	11	9
7	22	10	8	5	8	3	0	2	5	7	20	14	11
8	24	13	10	8	10	5	2	0	2	5	23	16	14
9	26	15	13	10	13	7	5	2	0	2	25	19	16
10	29	17	15	13	15	10	7	5	2	0	27	21	19
11	6	10	12	15	12	18	20	23	25	27	0	7	9
12	8	3	6	8	6	11	14	16	19	21	7	0	2
13	10	1	3	6	3	9	11	14	16	19	9	2	0
m_i	120	150	90	120	90	30	60	60	45	130	120	90	30
M_i	∞	200	120	180	125	40	120	120	75	∞	∞	120	60
r_i	31	22	22	22	25	23	22	22	22	47	27	22	30

Table A2: "ligne1" benchmark data (empty move times $d_{i,j}$ from tank i (line i) to tank j (column j), m_i, M_i and r_i)

$d_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1	0	13	12	11	10	9	9	8	8	6	6	5	5
2	13	0	4	5	6	7	7	8	9	10	10	11	12
3	12	4	0	4	5	6	6	7	8	9	9	10	11
4	11	5	4	0	4	5	6	6	7	8	9	10	10
5	10	6	5	4	0	4	5	6	6	7	8	9	9
6	9	7	6	5	4	0	4	5	5	6	7	8	8
7	9	7	6	6	5	4	0	4	4	5	6	7	8
8	8	8	7	6	6	5	4	0	4	4	5	6	7
9	8	9	8	7	6	5	4	4	0	5	5	6	7
10	6	10	9	8	7	6	5	4	5	0	4	5	5
11	6	10	9	9	8	7	6	5	5	4	0	4	5
12	5	11	10	10	9	8	7	6	6	5	4	0	4
13	5	12	11	10	9	8	8	7	7	5	5	4	0
m_i	360	60	30	120	180	30	30	60	60	300	60	60	120
M_i	∞	120	90	240	240	90	90	120	120	420	120	120	180
r_i	23	29	19	24	19	19	24	24	20	24	24	19	30

Table A3: "ligne2" benchmark data (empty move times $d_{i,j}$ from tank i (line i) to tank j (column j), m_i, M_i and r_i)

$d_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	0	5	5	6	8	9	10	12	13	15	16	18	19	21	25
2	5	0	4	5	7	8	9	11	12	14	15	16	18	19	23
3	5	4	0	4	6	7	7	10	11	12	14	15	17	18	22
4	6	5	4	0	5	6	6	8	9	11	13	14	15	17	21
5	8	7	6	5	0	4	5	7	8	10	11	12	14	15	20
6	9	8	7	6	4	0	4	6	7	8	10	11	13	14	18
7	10	9	7	6	5	4	0	5	6	7	9	10	12	13	17
8	12	11	10	8	7	6	5	0	4	6	7	8	10	11	16
9	13	12	11	9	8	7	6	4	0	5	6	7	9	10	14
10	15	14	12	11	10	8	7	6	5	0	5	5	7	8	13
11	16	15	14	13	11	10	9	7	6	5	0	4	6	7	11
12	18	16	15	14	12	11	10	8	7	5	4	0	5	6	10
13	19	18	17	15	14	13	12	10	9	7	6	5	0	5	9
14	21	19	18	17	15	14	13	11	10	8	7	6	5	0	8
15	25	23	22	21	20	18	17	16	14	13	11	10	9	8	0
m_i	300	180	60	60	180	60	30	60	60	180	60	60	60	660	240
M_i	∞	300	120	120	240	120	120	120	120	300	180	120	150	720	∞
r_i	15	24	14	25	29	19	20	24	20	25	19	20	25	18	35