Bi-local search based variable neighborhood search for job-shop scheduling problem with transport constraints

Moussa Abderrahim, Abdelghani Bekrar, Damien Trentesaux, Nassima Aissani, Karim Bouamrane

Optimization Letters (2022) 16:255–280
https://doi.org/10.1007/s11590-020-01674-0

ORIGINAL PAPER

Bi-local search based variable neighborhood search for job-shop scheduling problem with transport constraints

Moussa Abderrahim[1] · Abdelghani Bekrar[2] · Damien Trentesaux[2] · Nassima Aissani[3] · Karim Bouamrane[1]

Received: 12 April 2020 / Accepted: 16 November 2020 / Published online: 26 November 2020

Abstract
In job-shop manufacturing systems, an efficient production schedule acts to reduce unnecessary costs and better manage resources. For the same purposes, modern manufacturing cells, in compliance with industry 4.0 concepts, use material handling systems in order to allow more control on the transport tasks. In this paper, a job-shop scheduling problem in vehicle based manufacturing facility that is mainly related to job assignment to resources is addressed. The considered job-shop production cell has two types of resources: processing resources that accomplish fabrication tasks for specific products, and transporting resources that assure parts' transport to the processing area. A Variable Neighborhood Search algorithm is used to schedule product manufacturing and handling tasks in the aim to minimize the maximum completion time of a job set and an improved lower bound with new calculation method is presented. Experimental tests are conducted to evaluate the efficiency of the proposed approach.

Keywords Job-shop scheduling · Transport constraints · Variable neighborhood search

1 Introduction

Modern manufacturing facilities that comply with Industry 4.0 use flexible resources to ensure more control on their production lines. This allows production workshops to respond quickly and with a minimum investment to an unexpected growing of activities or compensating resources failures. In this field, flexibilization of the transport system

The first author of this work has been funded by the Algerian Ministry of Higher Education and Scientific Research through the Exceptional National Program scholarship under n.°:97/PNE/enseignant/France/2018–2019. The work described in this paper was conducted within the framework of the joint laboratory "SurferLab" founded by Bombardier, Prosyst and the Université Polytechnique Hauts-de-France. This Joint Laboratory is supported by the CNRS, the European Union (ERDF) and the Hauts-de-France region.

Extended author information available on the last page of the article

inside the manufacturing cell is a key element to design a more adaptive production schedule.

Technologies enable plants owners to easily reconfigure their process and set new objectives through flexible Material Handling System (MHS). Automated Guided Vehicles (AGV) are commonly chosen by manufacturers to implement truly flexible MHS [1]. They are used for transport and storage functions and can be managed to deal with other manufacturing task schedules to meet desired production objectives initially outlined. In compliance with this goal, an effective task scheduler needs to reorder the realization of a set of operations while considering allocation constraints to the required resources (transport resources, manufacturing resources, …) in the aim to optimize objectives values.

The organization of the transport system has an impact on the performance of the manufacturing task schedules [2], which has fostered us to consider it in our study. More precisely, in this paper, we consider the scheduling problem in job shop systems with transport vehicles. Like standard scheduling problem, it consists of assigning set of tasks (i.e. production and transport tasks) to a set of resources (i.e. processing machines, transport vehicles), while minimizing the maximum completion time of a production order (i.e. makespan), and taking into account the related constraints (i.e. production and transport constraints).

To solve this problem, we use the Variable neighborhood Search (VNS) metaheuristic to rearrange different task scheduling (i.e. transport and processing tasks). This metaheuristic incorporates asynchronous local search routines that operate together to find the better resources task allocation while keeping the makespan minimized.

Our paper is organized as follows: in the next section, we start by addressing a state of the art of the VNS based Job-shop studies and we position our contribution in this context. Then we present the proposed approach, describe the developed model and list related algorithms. Section 4 is dedicated to detail experiments and Sect. 5 reports numerical results which are later discussed in Sect. 6. Finally, Section 7 draws a conclusion from the obtained results and highlights our perspectives for future research.

2 Literature and related works

The Job-shop Scheduling Problem (JSP) is an optimization problem in which resources are allocated to perform a predetermined collection of tasks. A great deal of effort was invested for developing methods in this field. Several papers have been published to enumerate those studies in a comparative way in order to highlight their pros and cons for future works [3]. Approximate approaches family regroups the wide range of methods and contributions due to the continuous development of computer technology and intelligent algorithms [4,5]. It encompasses a large number of subfamilies from which Metaheuristics presents the most used approaches [6]. VNS belongs to this family and despite that, still remains insufficiently explored in the context of JSP [7].

VNS was firstly introduced by N. Mladenovic and P. Hansen in 1997, their motivation comes from the fact that the majority of metaheuristics have developed complex solutions to avoid being trapped by the first local optimum which make the effec-

tiveness check process of the solutions provided by those approaches very difficult [8]. The solution they proposed was very simple: jump to a different neighborhood from the current one, use a local search routine to find out a better solution within this new neighborhood and repeat this process until reaching the stopping condition (fixed number of neighborhoods, maximum running time, maximum loops within a local search, …).

To the best of our knowledge, the first implementation of VNS for solving JSP was introduced in 2006 by Mehmet Sevkli and M. Emin Aydin in [9]. In their study, they found that the main bottleneck of VNS was in the pairing strategy between the shake and local search functions so they propose a novel implementation of VNS in which they substitute the shake routine by a combination of insert and exchange heuristics and the local search process by a sequential application of the same heuristics within a loop. Afterward in 2009, Roshanaei et al. propose a new VNS implementation, to minimize makespan on job shop scheduling with set-up times, based on different local search technique. They used a systematic switch between three insertion based neighborhood search structures to overpass the notorious myopic behavior of the traditional VNS local search [10]. Karimi et al. in 2012 also focused on enhancing the local search routine in VNS for the flexible job shop scheduling problem, they incorporated in [11] a knowledge base module to guide the VNS local search process by extracting solutions and feed them back to the algorithm. More recently, authors in [12] treated the machine assignment to operations problem in a flexible JSP through a hybrid approach that combines Genetic Algorithm (GA) for global search process and Variable Neighborhood Descendant (VND) for local exploration. This technique allows at once the enhancement of the local search ability through a systematic change of neighborhood structures within the local search process, and encompasses both intensification and diversification. Reference [13] used also VND to enhance the local search ability; and the Differential based Harmony Search algorithm (DHS) for global enhancement, to accelerate the convergence speed, while maintaining the diversity of the explored population. They proved, through an extended series of tests, that their JSP optimization approach outperforms other proposed models in the same field.

Adding the transport constraints to the classical JSP makes the resulting problem a combination of two NP-hard sub-problems [14], therefore, few papers that combine both sub-problems can be found in the literature compared to those that deal with each problem separately. Knust and Hurink studied JSP with transportation times and a single robot in [15] and [16]. They considered the transportation resource as an additional special machine that has a sequence-dependent setup times representing robot empty moves to carry job from different machines. They used a disjunctive graph to model the final problem. In addition to the work of [17], a benchmark for JSP scheduling problem with a single transporting robot was proposed [18]. Bilge and Ulusoy in [19] proposed a benchmark with two robots, four different layouts with an additional loading/unloading station and ten job sets examples. They studied the interaction between machine scheduling and MHS that is not allowed to return to the loading/unloading station after each transportation job. Both benchmarks consider conflict-free unidirectional manufacturing layouts with predetermined shortest paths routing problem and are widely used in the literature. Authors in [20] proposed a mathematical model to

schedule one vehicle based manufacturing facility with makespan minimization objective. They took into account additional parameters like the number of allowed jobs in the system and input/output buffer capacities. Their model's behavior was validated using a modified version of Bilge and Ulusoy benchmark. A. Ham proposed in [21] two constraint-programming (CP) approaches to modelise simultaneous scheduling of machine and transfer-robots in JSP. He provided tests on [19] and a large-scale JSP benchmark instances and proved that the proposed exact model converges to optimal values in record time (less than one second in most of the cases). References [22] and [23] treated a JSP with blocking constraints in a multiple AGV based MHS. Both contributions considered a job shop cell with an additional loading/unloading station, and provided a local search based approach to optimize the final schedule solution. Other papers in the literature used local search based metaheuristics for makespan optimization: researchers in [24] proposed an iterated local search, simulated annealing and an hybridization of both to deal with AGV and machine scheduling in JSP. They used [19] benchmark to provide both enhanced makespan results and new findings on minimizing the exit time of the last job from the system. In [25], a GA with tabu search procedure is implemented with an extended series of tests on both [15] and [19] benchmarks and L. Deroussi in [26] highlighted the non significant difference between a stochastic and a deterministic local search when combined with particle swarm optimization (PSO). Later on, authors in [27] introduced a local neighborhood search algorithm (LNSA) to minimize the makespan in JSP with different possible locations for processing machines, and researchers in [28] considered also controllable machine locations along with variable transport times in JSP and introduced four local search based metaheuristics to deal with facility energy cost and the job tardiness penalty optimization problem. Both last studies provided tests on small and large scale instances to validate the efficiency of proposed models.

Our contribution in this paper consists of adapting VNS for the first time to the JSP with transport constraints by proposing a novel implementation way using asynchronous local search routines. A new computing method is also proposed to improve the lower bounds calculation with an extended series of tests to demonstrate the efficiency and the value that would complement the existing literature on the topic.

3 The proposed model and solving approach

In this section, the treated problem is detailed with the associated notations, its mathematical programming model is formulated and the representation and solving approach are addressed.

3.1 Problem description

The studied scheduling problem is a JSP that takes in consideration transport duration between machines. The considered production process can be represented as follows : a set of independent jobs $j \in J, |J| = n$, each consists of an ordered list of tasks $i \in O_j, |O_j| = s_j$ that have to be processed separately; every one on a specific

uni-task machine $m \in M$. The task ij (i.e. the task i of job j) has to be performed on its associated machine m without preemption during a defined duration t_{ijm}. The notation $o(i, j, m)$ expresses that the task ij is performed on the machine m (for example $o(0, 1, 3)$ states that the task '01' is performed on machine '3'). We assume that the number of tasks s_j can differ from a job to another and that the number of machines is limited.

During its manufacturing process, a job j has to move from a machine to another to perform its next task in O_j. This is assured by a single uni-charge vehicle k from the available transport fleet set A. k can start transporting a combination $o(i, j, m)$ only when its previous combination $o(i-1, j, m')$ is achieved. Thus, we consider m' as the call node of a task $o(i, j, m)$.

The transport fleet A has a limited number of uni-charge vehicles which are typically AGV or forklifts in a MHS. The transportation process starts from the loading/unloading station R at time $t = 0$; the time in which all jobs and vehicles are considered available in that station R.

The moving process managed by a vehicle k to allow a job j to perform its task i with $o(i, j, m)$ is called : the transportation job $p(k, i, j)$. It is composed of two sequential transportation tasks (see Fig. 1): $p(k, i, j, 0)$ where the vehicle k is moving empty from its current position m'' to the call node m' with $o(i-1, j, m')$, and $p(k, i, j, 1)$ in which the vehicle k moves the job j to its next processing machine m. This means that there are precedence constraints between $p(k, i, j, 0)$ and $p(k, i, j, 1)$, $o(i-1, j, m')$ and $p(k, i, j, 1)$, and between $p(k, i, j, 1)$ and $o(i, j, m)$ (i.e. periods of time Δt, $\Delta t'$ and $\Delta t''$ in Fig. 1 that separate between tasks durations should be ≥ 0). Also, we suppose that machines have unlimited waiting lines for products and sufficient attached area to allow vehicle waiting for next transport call.

3.2 Mathematical programming formulation

The mathematical programming formulation of the treated problem is a time interval based representation. The proposed mixed integer linear program (MILP) is composed of three parts' constraints : JSP, transport and relationship between both. However, we assume that the behavior is ideal for the best execution characteristics (i.e. no machine or vehicle breakdowns, no vehicle conflicts).

In the following, our MILP's sets, parameters, and decision variables are first presented, then constraints of each part are detailed separately :

Sets

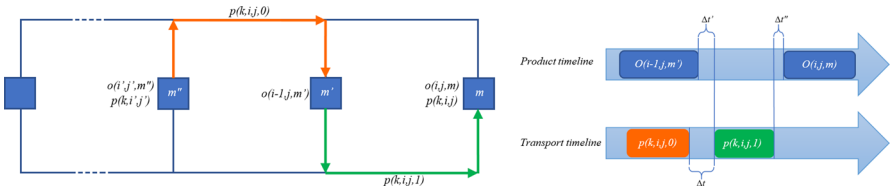

Fig. 1 Processing and transport jobs timelines

J set of jobs.
O_j set of tasks that belong to the job j.
M set of machines.
A set of transport vehicles.
P set of transport jobs (or $p(k, i, j)$ in the last paragraph).

Parameters

R loading/unloading station.
e_{ijm} a binary parameter that equals 1 if the task ij can be performed on machine m, 0 otherwise.
t_{ijm} processing time of the task ij on the machine m.
$tt_{mm'}$ transporting time from the node m to the node m' : a node is either a machine $\in M$ or the loading/unloading station R.
N a big number.

Decision variables

a_{ijm} an integer variable that represents the starting time of the task ij on the machine m.
c_{pk} an integer variable that represents the starting time of the transportation job p using the vehicle k.
d_{pk} an integer variable that represents the ending time of the transportation job p using the vehicle k.
$g_{i'j'm}^{ij}$ a binary variable that equals 1 if the task ij precedes the task $i'j'$ on the machine m, 0 otherwise.
h_{ij}^{pk} a binary variable that equals 1 if the transportation job p of the vehicle k is carrying the task ij, 0 otherwise.
C_{max} an integer variable that represents time required to achieve a list of jobs.

3.2.1 JSP constraints

The first part of this mathematical programming model describes the JSP formulation in a similar way to the formulation proposed in [29]. However, as the considered JSP in this paper is not flexible, processing tasks allocation to machines is avoided (i.e. in this paper, each task is processed on one and only one machine).

Task time interval constraints Processing tasks on machines are presented by the starting time and the task duration parameter.

$$a_{ijm} + t_{ijm} \leq e_{ijm} \times N; \forall j \in J,\ \forall i \in Oj,\ \forall m \in M \tag{1}$$

*Precedence constraints*This constraint ensures the sequence inside a job. Each starting time of a processing task should consider ending time of its previous task.

$$\sum_{m \in M} a_{ijm} \geq \sum_{m \in M} (a_{i-1jm} + t_{i-1jm}); \forall j \in J,\ \forall i \in O_j,\ i > 0 \tag{2}$$

*Disjunctive constraints*A processing task is performed by only one machine, for two different tasks that are affected to the same machine one should precede the other (i.e. the second should start after the first ends), and a task can not precede its self.

$$\sum_{m \in M} e_{ijm} = 1;\ \forall j \in J, \forall i \in O_j \tag{3}$$

$$g^{ij}_{i'j'm} + g^{i'j'}_{ijm} \leq 1;\ \forall j, j' \in J, j \neq j', \forall i \in O_j, \forall i' \in O_{j'}, \forall m \in M \tag{4}$$

$$e_{ijm} \times e_{i'j'm} = g^{ij}_{i'j'm} + g^{i'j'}_{ijm}; \forall j, j' \in J, j \neq j', \forall i \in O_j, \forall i' \in O_{j'}, \forall m \in M \tag{5}$$

$$a_{i'j'm} \geq a_{ijm} + t_{ijm} - (1 - g^{ij}_{i'j'm}) \times N; \forall j, j' \in J, j \neq j', \forall i \in O_j, \forall i' \in O_{j'}, \forall m \in M \tag{6}$$

$$g^{ij}_{ijm} = 0;\ \forall i \in O_j, \forall j \in J, \forall m \in M \tag{7}$$

Makespan calculationC_{max} (or makespan) refers to the time at which the last task of the last job ends. Note that $s_j - 1$ corresponds to the index of the last task of the job j (by taking in consideration that tasks' indexation starts from 0 for all jobs).

$$C_{max} \times e_{s_j - 1jm} \geq\ a_{s_j - 1jm} + t_{s_j - 1jm}; \forall j \in J, s_j = |O_j|, \forall m \in M \tag{8}$$

3.2.2 Transport constraints

The second part of our MILP describes the transport schedule formulation.

*Transport job time interval constraints*For all vehicles, the first transport job must start at time $t = 0$ and a transport job ending time must be greater than its starting time. For general case, it ends after performing a move to the call node and transporting the job to its next station. In case when the processing job is transported for the first time (i.e. it is the first task of the job j), the call node is the station R. Finally, the moving to the call node may be omitted if both the transport vehicle k and the job j operate for the first time (as both will be located at the station R).

$$c_{0k} = 0;\ \forall k \in A \tag{9}$$

$$d_{pk} \geq c_{pk}; \forall k \in A, \forall p \in P \tag{10}$$

$$d_{pk} \geq c_{pk} + tt_{m'm} + tt_{mm''} - (2 - h^{pk}_{ij} \times e_{i-1jm} \times e_{ijm''} - h^{p-1k}_{i'j'} \times e_{i'j'm'}); \forall p \in P,\ p > 0, \forall k \in A, \forall m, m', m'' \in M, \forall j, j' \in J, \forall i \in O_j, i > 0, \forall i' \in O_{j'} \tag{11}$$

$$d_{pk} \geq c_{pk} + tt_{m'R} + tt_{Rm} - (2 - h^{pk}_{0j} \times e_{0jm} - h^{p-1k}_{i'j'} \times e_{i'j'm'}); \forall j, j' \in J, \forall m, m' \in M, \forall i' \in O_{j'}, \forall p \in P,\ p > 0, \forall k \in A \tag{12}$$

$$d_{0k} \geq tt_{Rm} - (1 - \times h^{0k}_{0j} \times e_{0jm}) \times N; \forall j \in J, \forall m \in M,\ \forall k \in A \tag{13}$$

*Transport disjunctive constraints*These constraints ensure that a product job task must be transported by only one vehicle, a transport job concerns at most one processing task, and a vehicle k can perform only one transport job at once.

$$\sum_{p\in P,k\in A} h_{ij}^{pk} = 1;\ \forall j \in J, \forall i \in O_j \tag{14}$$

$$\sum_{j\in J,i\in O_j} h_{ij}^{pk} \leq 1; \forall p \in P, k \in A \tag{15}$$

$$c_{pk} \geq d_{p-1k}; \forall p \in P,\ p > 0,\ \forall k \in A \tag{16}$$

*Transport jobs ordering constraints*These constraints ensure that a vehicle can't perform any move if it is not affected to a transport task, and place the unassigned transport jobs (i.e. those having $\sum_{j\in J,i\in O_j} h_{ij}^{pk} = 0$) at the end of the transport assignment list. This allows to constraints (11) and (12) to successfully retrieve previous position of a vehicle k for a transport job p by simply querying the previous transport job $p-1$ and also prevent vehicles from performing additional moves to ameliorate their position for next transport calls. Note that these constraints are logical and are automatically transformed into equivalent linear formulations when they are extracted by the IBM ILOG CPLEX Solver [30].

$$\sum_{j\in J,i\in O_j} h_{ij}^{pk} \leq 0 \implies d_{pk} = c_{pk}; \forall p \in P, k \in A \tag{17}$$

$$\sum_{j\in J,i\in O_j} h_{ij}^{pk} \leq 0 \implies \sum_{p'\in P, j'\in J, i'\in O_{j'}} h_{i'j'}^{p'k} \leq 0; \forall p \in P,\ p' > p, k \in A \tag{18}$$

3.2.3 JSP and transport relationship constraints

These constraints ensure the precedence between both processing task and its transport job: a task ij starts after its transportation finishes, and a vehicle k can carry a task ij only after its previous task $i-1j$ ends.

$$a_{ijm} \geq d_{pk} + (h_{ij}^{pk} \times e_{ijm} - 1) \times N; \forall j \in J, \forall i \in O_j, \forall m \in M, \forall p \in P, \forall k \in A \tag{19}$$

$$d_{pk} \geq a_{i-1jm} + t_{i-1jm} + tt_{mm'} + (h_{ij}^{pk} \times e_{i-1jm} \times e_{ijm'} - 1) \times N;$$
$$\forall j \in J, \forall i \in O_j, i > 0, \forall m, m' \in M, \forall p \in P, \forall k \in A \tag{20}$$

The objective is to generate an optimal scheduling for both machine and transport jobs that minimizes the maximum completion time (makespan) of a given set of jobs. This implies finding simultaneously the optimal allocation time for both $o(i, j, m)$ and $p(k, i, j)$ combinations (or Machine-vehicle Schedule).

$$Minimize\ C_{max}$$

As we employ a metaheuristic to reach this goal, the basic and key feature of these methods is that they work on coding space [31]. Hence, using an encoding technique that handles our problem constraints and avoids infeasible solutions is the key element for an efficient approach. In the next section, we describe the technique used to represent the Machine-vehicle Schedule in VNS.

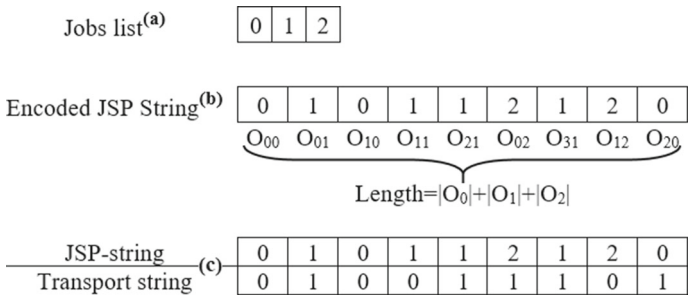

Fig. 2 The used schedule representation in VNS

3.3 Schedule representation

To consider the transportation stage along with an existing classical JSP, a two-string based representation has been applied where processing tasks allocation to machines is scheduled in the first string (JSP-string) and transport vehicles are affected to those tasks in the second (transport string).

In the first part, the system receives an entry list of n integers representing the requested jobs, thus, the list '012' refers to three jobs of type '0', '1' and '2' respectively (see Fig. 2a). From this entry, a JSP-string is generated by repeating each job from the list according to the size of its tasks set (i.e. if the job '1' has four tasks, the character '1' will be repeated four times in the JSP-string). Hence, each element from the new string takes the name from its parent job and is interpreted according to its appearance order in that string (i.e. the first '1' in JSP-string corresponds to the first task of the job '1', the second '1' is referring to the second task of the job '1' and vice versa) (see the example in Fig. 2b where O_{ij} is the task i of the job j). This is called : operation (or task) based representation, as detailed in [31]. Using this representation, infeasible solutions for our schedule have been avoided and thus, an additional cleaning step in our metaheuristic has been omitted [32].

The second part is the representation needed to append the vehicle allocation to the JSP schedule tasks. The simplest way has been utilized by reproducing the same JSP-string from previous step, and substituting tasks numbers by vehicle ids to generate the transport-string. Thus, the combination of the two strings will represent both assignment ordering of job operations to machines and assignment of transport vehicles to those operations (i.e. the job task in the position q of the JSP-string will be transported by the vehicle id in the same position q in the transport string) (see Fig. 2c). Transport string elements are integers between 0 (for the first vehicle) and $length(A) - 1$. In line with our JSP-string, no infeasible transport-string could be generated using the suggested representation.

Still for our example, using the notation of the Sect. 3.1, the transportation plan of Fig. 2c is described as follows: p(0,0,0), p(1,0,1), p(0,1,0), p(0,1,1), p(1,2,1), p(1,0,2), p(1,3,1), p(0,1,2), p(1,2,0).

3.4 The proposed approach

VNS is based on neighborhood change, and acts both vertically to look for the local optimum through the local search and horizontally to escape the current valley for a new local optima [8].

Using the schedule representation presented in previous section, strings representing a particular JSP will have the same letter enumeration (i.e. only the order of letters can change from a JSP string to another). For example, all possible strings for the JSP of Fig. 2b will have three '0', four '1' and two '2', while, strings of the transport schedule can have different letter enumeration and thus a larger number of possible solutions. Hence, we choose to consider horizontal exploration only for the transport schedule and incorporate local searches for both transport and JSP neighborhoods in the same parallel vertical exploration.

3.4.1 Comparison with previous parallel VNS approaches

To gain in performances in terms of computational time and solution quality efficiency, the parallel implementation of VNS metaheuristic has been advantaged [33]. According to Crainic et al. 2004 in [34], a parallel metaheuristic implementation can be realized through three main manners :

- *Low-level parallelism* in which parts of the code are dispatched between a set of process, according to the master-slave computing model, to minimize the execution time [35,36].
- *Domain decomposition* where the solution space is divided into multiple areas that are later on dispatched between a set of parallel processors.
- *Multiple search* characterized by exploring the solution space using multiple concurrent processes that may inter-communicate depending on the chosen strategy.

Based on this classification, Table 1 has been constructed to list featured characteristics of the main parallel VNS implementations, and point out the differences with our proposed implementation.

García-López et al. 2002 in [37] proposed the Synchronous Parallel VNS (SPVNS) to parallelize VNS local search routines, the most consuming time part. SPVNS is based on domain decomposition, independent parallel local search processes with common initial solution and shaking step. The Replicated Parallel VNS (RPVNS) differs from SPVNS in the fact that all searching processes explore simultaneously the same solution space with different initial solutions and shaking phases, while the Replicated Shaking VNS (RSVNS) allows concurrent processors to cooperate, by exchanging best found solutions, to quickly enhance results quality. All RSVNS, Cooperative Neighborhood VNS (CVNS) by Crainic et al. 2004 in [34], Unidirectional-ring and Mesh topology proposed by Sevkli et al. 2007 in [33] have the same characteristics but differ in the application of the cooperative mode as described in [38]. The proposed asynchronous VNS is a cooperative parallel VNS implementation, but it is quite similar to the SPVNS. In fact, the domain decomposition feature allows each of the two local search routines to isolate its own exploring area, while the cooperation mechanism initiates several searches that asynchronously exchange information

Table 1 Comparison of the proposed VNS approach with the main parallel VNS variants in literature

Ref	VNS Variant	Domain decomposition	Multiple search			Concurrent mode
			Parallel Shake	Parallel LS	Same initial solution	
[37]	SPVNS	Yes	No	Yes	Yes	Independent
	RPVNS	No	Yes	Yes	No	Independent
	RSVNS	No	Yes	Yes	No	Cooperative
[34]	CNVNS	No	Yes	Yes	No	Cooperative
[33]	Unidirectional-ring topology	No	Yes	Yes	No	Cooperative
	Mesh topology	No	Yes	Yes	No	Cooperative
	Our approach	Yes	No	Yes	Yes	Cooperative

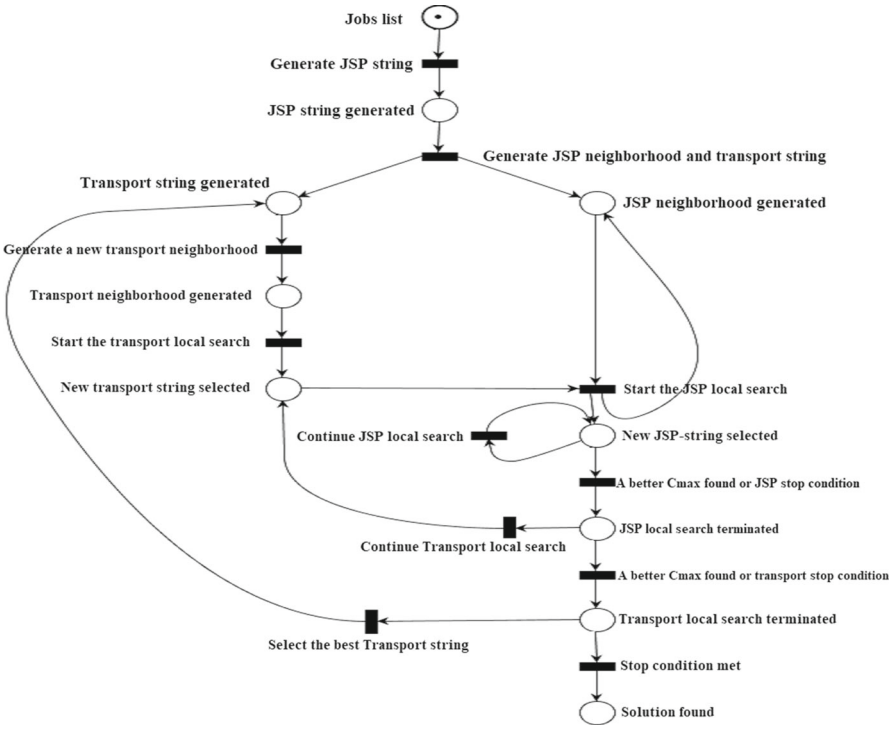

Fig. 3 The proposed VNS model behavior

about the best solution and thus accelerate the exploration process. The details of this implementation are described in the next section.

3.4.2 The description of the proposed approach

Our model's behavior is described via Petri net graph in Fig. 3. As VNS results depend very little on the chosen rule to generate initial solution [39], we use for our metaheuristic input a randomly generated strings for both JSP and transport schedules (it has been highlighted in the previous section that the used schedule representation generates always feasible solutions). A random JSP string is generated from the jobs list in compliance with the representation described in the previous section. It is used as a root for the JSP schedule neighborhood and to generate randomly a same length first transport schedule string as described in the second part of Sect. 3.3. This last is used to generate the first transport neighborhood. A local search is then initiated to explore the solution set for a better makespan.

The novelty of our approach within VNS is the use of two asynchronous local search stages; one for each scheduling sub problems: a transport schedule local search is started within the current transport neighborhood, then, a second local search is used on JSP neighborhood to find the JSP-string that gives the best C_{max} with the current transport string; while keeping the JSP local search process available for next

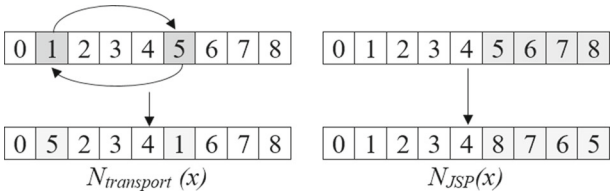

Fig. 4 Both transport and JSP neighborhood structures

transport strings. We repeat the same with all elements from the current transport neighborhood until the local search stop condition is met or a better JSP-transport combination is found. Thereafter, the whole process is repeated with a new transport neighborhood until reaching the stop condition. Note that since the initial solution is chosen at random, we are using the first improvement technique for both local searches as recommended in [40]. The details of our VNS main functions are described in the next section.

3.5 The proposed VNS implementation

Let a feasible solution T composed of two strings T_{JSP} and $T_{transport}$ for each sub problem. T is used as an initial solution for the VNS iteration process. It represents a randomly generated solution if it is the first run, or the best-found solution from previous steps.

When it is not the first run, the shake function, described in Algorithm 1, uses $T_{transport}$ as an input to generate a new string $T'_{transport}$ serving as root (or seed) for the next local search phase. Later on, two asynchronous routines *JSPLocalSearch* (Algorithm 2) and *TransportLocalSearch* (Algorithm 3) are initialized to perform the local search process on both JSP and transport neighborhoods respectively.

Algorithm 1: The shake function

```
1 function shake(x)
    Input  : A transport schedule string
    Output : A new random transport schedule string
2 choose random different indexes c1, c2 ≤ length(x) where c1 < c2
3 for i : c1 ... c2 do
4 |   x[i] ← random AGV id
5 end
6 return x
```

A transport neighborhood $N_{transport}$ is defined by all random exchanges between two random elements from the input transport string, while a JSP neighborhood N_{JSP} by all random block reverses of length l as described in Fig. 4 with $2 \leq l \leq length(T_{JSP})$.

The *TransportLocalSearch* function starts exploring the transport neighborhood $N_{transport}(T'_{transport})$ (or $N_{transport}(T_{transport})$ if it is the first run) where $T'_{transport}$

Algorithm 2: The local search function for JSP schedule

```
1  function JSPLocalSearch (x, C_max)
   Input  : A transport schedule x and the best found C_max value
   Output : The best JSP schedule y found for the input transport schedule x
2  noImprovementCount ← 0
3  y ← randomJSPString()
4  repeat
5      if cost(x, y) < C_max then
6          return y
7      else
8          noImprovementCount++
9          y ← select the next string from the current JSP neighborhood
10     end
11 until noImprovementCount > maxNoLocalImprovements;
12 return y
```

Algorithm 3: The local search function for transport schedule

```
1  function TransportLocalSearch (x, C_max)
   Input  : Current transport neighborhood root element x and the best C_max
   Output : The best found combination x, y and its C_max value
2  noImprovementCount ← 0
3  repeat
4      y ← JSPLocalSearch(x, C_max)
5      if cost(x, y) < C_max then
6          return x, y, cost(x, y)
7      else
8          noImprovementCount++
9          x ← select the next string from the current transport neighborhood
10     end
11 until noImprovementCount > maxNoLocalImprovements;
12 return x, y, C_max
```

is the solution generated in the shake phase. At each iteration, the *JSPLocalSearch* function is called to select the JSP-string within $N_{JSP}(T_{JSP})$ neighborhood that gives a better C_{max} value for the current transport string (C_{max} calculation is performed through the function $cost(T_{transport}, T_{JSP})$ function).

Both implemented local search routines perform asynchronously and are first-improvement-based in which the local search process is interrupted once the first progress of the last known best solution is detected. In VNS, this choice depends on the mechanism used to generate the initial solution as recommended by Hansen et al in [41].

If no better solution is found within *maxNoLocalImprovements* loops, the local search process is stopped and the best combination is saved in the upper level. Afterwards, the whole process restarts over the shake function until reaching the stop condition. VNS stops generating new transport neighborhoods if no better solution is found within *maxNoGlobalImprovements* loops parameter.

4 Experimentation

To validate our approach, series of tests have been led on widely used benchmark instances in JSP with transport constraints domain. Also, a sequential implementation of our model has been tested to highlight the potential benefit of the proposed model.

4.1 Instance description

We use the benchmark of Bilge and Ulusoy 1995 presented in [19] which is a common reference for job-shop scheduling with transport constraints. It considers collision free unidirectional routes manufacturing facility with 4 different topology layouts, 2 identical transport robots, 4 processing machines and one loading/unloading station with the following assumptions:

- Loading and unloading times are included in the travel duration,
- Travel durations are constant either traveling empty or loaded,
- Robots are unicharge vehicles,
- The considered makespan is the completion time of the last job on the machine (not the exit time of the last job in the L/U station).

According to [19], the major parameter that influencing the interaction of the vehicle and machine scheduling is the relative magnitude of the processing time ($\bar{p}_i$) with respect to the travel times ($\bar{t_{ij}}$), therefore, Bilge and Ulusoy 1995 design problem instances at two different levels of the $\bar{t_{ij}}/\bar{p}_i$ ratio :

1. First problem group in which $\bar{t_{ij}}/\bar{p}_i > 0.25$ representing instances using a data set the real with distance/processing times defined in the appendix of [19],
2. Second problem group in which $\bar{t_{ij}}/\bar{p}_i < 0.25$ representing instances using the same data set but with halved travel times and doubled or tripled processing times for instances according to 0 or 1 digit at last instance name respectively.

Instances nomination code uses EX (abbreviation of EXample) followed by job set and layout digits. An additional 0 or 1 digit for the second problem group implies that travel times are halved and processing times are doubled or tripled respectively. For example: EX610 represents instance using the job set 6 with layout 1. The additional 0 digit at the end implies that travel times are halved, and processing times are doubled compared to those used in EX61.

4.2 VNS tuning

To tune parameters of the proposed VNS, experiments are conducted on both problem groups described in previous section. Tuned instances are selected randomly from the 82 available problems, to investigate three VNS parameters:

1. Maximum authorized neighborhood change without improvement.
2. Maximum authorized transport local-search loops without improvements.
3. Maximum authorized JSP local-search loops without improvements.

We found that increasing the maximum authorized loops without improvement for the JSP local-search routine has not a great impact on the quality of VNS results. Contrarily, we obtained better results by increasing transport local search loops. This may be due to the large number of possible transport strings comparing to JSP strings. The value for both parameters has been set to 50 and 15 respectively according to the conducted experiments. Finally, our VNS has been configured to stop the searching process if no solution improvement is detected within the last 30 neighborhood changes.

4.3 Computation of a new lower bound

To reduce the calculation time for our mathematical programming model we suggest a new lower bound calculation method. In fact, the problem with machine-vehicle scheduling resides in selecting vehicles that minimize the idle times between two successive operations (i.e. operations of the same job and/or on the same machines). Therefore, we have reduced to zero all waiting times for the transport vehicle and it has been assumed that there is always an available vehicle when a job need to be transported to its next machine. As a result, the lower bound will be equal to the best expected value for C_{max} after relaxing all transport constraints. The formulation of the new lower bound calculation can be presented by reproducing the JSP constraints from (1) to (8) from Sect. 3.2.1 extended with the following two equations :

$$a_{ijm'} \geq a_{i-1jm} + t_{i-1jm} + tt_{mm'} + ((e_{i-1jm} \times e_{ijm'}) - 1) \times N; \quad \forall j \in J, \forall i \in O_j, \forall m, m' \in M \tag{21}$$

$$a_{0jm} \geq tt_{Rm} + (e_{0jm} - 1) \times N; \forall j \in J, \ \forall m \in M \tag{22}$$

The constraint (21) states that processing a task ij can starts only after the time needed to finish its previous task $i-1j$ plus the time distance between both $i-1j$ and ij processing machines while constraint(22) represents the special case with the first task of the job (i.e. when $i = 0$).

5 Numerical results

Experiments are performed using a Personal Computer with an Intel i5 6300u CPU and 8Gb of RAM running on Windows10 64-bit. The mathematical programming model has been executed using the IBM CPLEX Solver v12.6.3.0 to calculate the optimal values for the studied instances. The VNS implementation is written in Microsoft Visual C# language while makespan calculation uses the Google constraint solver Ortools SAT library (v7.5.7466). Obtained results and comparisons with previous works are presented in Table 2 and Table 3 for both $\bar{t_{ij}}/\bar{p_i}$ ratio problems respectively.

The enhanced lower bound values are described in the column "New LB" and are compared with "LBu" column presenting the referential lower bound values proposed by Ulusoy et al. in [42] containing some calculation corrections provided by Zheng et al. in [43].

Table 2 Comparison of the obtained results with the literature for C_{max} minimization (instances with $\bar{t_{ij}}/\bar{p_i} > 0.25$)

Instance	LBu	New LB	Bilge et al. 1995	BKS	Asynchronous LS VNS			Sequential LS VNS			cplex
					BFS	Time (Sec)	Gap (%)	BFS	Time (Sec)	Gap (%)	
EX11	72	76	96	96	96	2,45	0	98	37,15	2,08	96
EX21	86	86	105	100	100	9,18	0	104	62,14	4	–
EX31	81	**88**	105	99	102	16,36	3,03	107	11,33	8,08	–
EX41	76	**78**	118	112	116	42,51	3,57	118	29,26	5,36	–
EX51	60	**65**	89	87	87	0,33	0	92	152,94	5,75	–
EX61	96	**108**	120	118	120	4,54	1,69	122	189,04	3,39	–
EX71	76	**77**	119	111	121	30,15	9,01	124	23,65	11,71	–
EX81	146	**161**	161	161	161	0,33	0	161	30,96	0	161
EX91	93	**105**	120	116	117	46,87	0,86	121	42,65	4,31	–
EX101	124	**133**	153	146	152	26,6	4,11	154	42,33	5,48	–
EX12	68	**76**	82	82	82	19,72	0	82	6,18	0	82
EX22	76	76	80	76	76	0,7	0	79	12,68	3,95	76
EX32	75	**80**	88	85	85	3,25	0	85	17,79	0	85
EX42	64	**70**	93	87	92	40,72	5,75	95	10,93	9,2	–
EX52	59	**64**	69	69	69	2,88	0	72	25,16	4,35	69
EX62	86	**98**	100	98	98	0	0	98	8,02	0	98
EX72	74	74	90	79	84	25,91	6,33	91	13,81	15,19	–
EX82	140	**151**	151	151	151	0	0	151	0,38	0	–
EX92	91	**98**	104	102	102	0,2	0	102	36,68	0	102
EX102	114	**128**	139	135	139	57,07	2,96	139	74,86	2,96	–
EX13	66	**74**	84	84	84	1,11	0	86	9,99	2,38	84
EX23	82	82	86	86	86	0,5	0	86	6,27	0	–
EX33	77	**82**	86	86	86	1,48	0	89	34,71	3,49	86

Table 2 continued

Instance	LBu	New LB	Bilge et al. 1995	BKS	Asynchronous LS VNS			Sequential LS VNS			cplex
					BFS	Time (Sec)	Gap (%)	BFS	Time (Sec)	Gap (%)	
EX43	66	**71**	95	89	92	30	3,37	100	44,45	12,36	–
EX53	57	**63**	76	74	74	0,28	0	79	19,86	6,76	74
EX63	88	**100**	104	103	104	0,72	0,97	104	18,84	0,97	–
EX73	76	76	91	83	87	21,61	4,82	91	19,69	9,64	-
EX83	142	**153**	153	153	153	0	0	153	6,29	0	–
EX93	93	**100**	110	105	105	14,19	0	105	19,14	0	–
EX103	116	**133**	143	137	143	15,23	4,38	146	14,38	6,57	–
EX14	68	**76**	108	103	103	4,71	0	108	0,43	4,85	103
EX24	84	84	116	108	113	11,5	4,63	115	38,68	6,48	–
EX34	84	**87**	116	111	113	16,73	1,8	119	51,84	7,21	–
EX44	76	**81**	126	121	131	10,83	8,26	134	15,67	10,74	–
EX54	56	**62**	99	96	97	15,24	1,04	100	76,03	4,17	–
EX64	90	**103**	120	120	129	9,9	7,5	123	13,09	2,5	–
EX74	76	**78**	136	126	134	16,99	6,35	137	199,69	8,73	–
EX84	148	**163**	163	163	163	0,96	0	169	190,42	3,68	–
EX94	91	**102**	125	120	123	59,61	2,5	127	9,63	5,83	–
EX104	120	**136**	171	157	169	2,09	7,64	177	390,1	12,74	–
ANOVA *p*-value:					**66,10%**			**38,06%**			

Table 3 Comparison of the obtained results with the literature for C_{max} minimization (instances with $\bar{t_{ij}}/\bar{p_i} < 0.25$)

Instance	LBu	New LB	Bilge et al. 1995	BKS	Asynchronous LS VNS			Sequential LS VNS			cplex
					BFS	Time (Sec)	Gap (%)	BFS	Time (Sec)	Gap (%)	
EX110	126	126	126	126	126	0	0	126	0,31	0	126
EX210	148	148	148	148	148	0	0	148	0	0	148
EX310	138	**148**	150	150	150	0	0	150	7,46	0	150
EX410	112	**116**	121	119	119	0,12	0	119	5,77	0	–
EX510	102	102	102	102	102	0	0	102	0,17	0	102
EX610	163	**186**	186	186	186	0	0	186	0,16	0	186
EX710	137	137	137	137	137	0	0	137	10,31	0	137
EX810	271	**292**	292	292	292	0	0	292	0	0	292
EX910	150	**174**	176	176	179	0,57	1,7	179	0	1,7	176
EX1010	218	**238**	238	238	238	0	0	238	0	0	238
EX120	123	123	123	123	123	0	0	123	0,16	0	123
EX220	143	143	143	143	143	0	0	143	0	0	143
EX320	135	**145**	148	145	145	0	0	145	7,3	0	145
EX420	111	**114**	116	114	114	0,24	0	114	19,67	0	114
EX520	99	**100**	100	100	100	0	0	100	0	0	100
EX620	160	**181**	183	181	181	0	0	181	0	0	181
EX720	136	136	136	136	136	0	0	136	0	0	136
EX820	268	**287**	287	287	287	0	0	287	0	0	287
EX920	150	**173**	174	173	173	0,41	0	173	1,86	0	173
EX1020	213	**236**	236	236	236	0	0	236	0	0	236

Table 3 continued

Instance	LBu	New LB	Bilge et al. 1995	BKS	Asynchronous LS VNS			Sequential LS VNS			cplex
					BFS	Time (Sec)	Gap (%)	BFS	Time (Sec)	Gap (%)	
EX130	122	122	122	122	122	0	0	122	0	0	122
EX230	146	146	146	146	146	0	0	146	0	0	146
EX330	136	**146**	149	146	146	0	0	146	0	0	146
EX430	110	**113**	116	114	114	18,63	0	114	10,95	0	–
EX530	98	**99**	99	99	99	0	0	99	0,34	0	99
EX630	161	**182**	184	182	182	0	0	182	0	0	182
EX730	137	137	137	137	137	0,41	0	137	8,85	0	137
EX830	269	**288**	288	288	288	0	0	288	0	0	288
EX930	151	**174**	176	174	174	0,49	0	174	0,76	0	174
EX1030	214	**237**	237	237	237	0	0	237	1,2	0	237
EX140	124	124	124	124	124	0	0	124	0,33	0	124
EX241	217	217	217	217	217	0	0	217	0	0	217
EX340	138	**148**	151	151	151	0	0	151	5,77	0	151
EX341	203	**218**	222	221	221	1,82	0	221	0,68	0	221
EX441	166	**169**	179	172	172	25,36	0	172	73,92	0	172
EX541	148	148	154	148	148	0	0	148	0	0	148
EX640	161	**184**	185	184	184	0	0	184	0	0	184
EX740	137	137	138	137	137	0	0	138	23,54	0,73	137
EX741	203	203	203	203	203	1,92	0	203	9,31	0	203
EX840	272	**293**	293	293	293	0	0	293	0	0	293
EX940	149	**173**	177	175	175	0,91	0	181	0	3,43	175
EX1040	216	**237**	240	240	240	0	0	240	0,43	0	–
ANOVA *p*-value:					**99,52%**			**98,41%**			

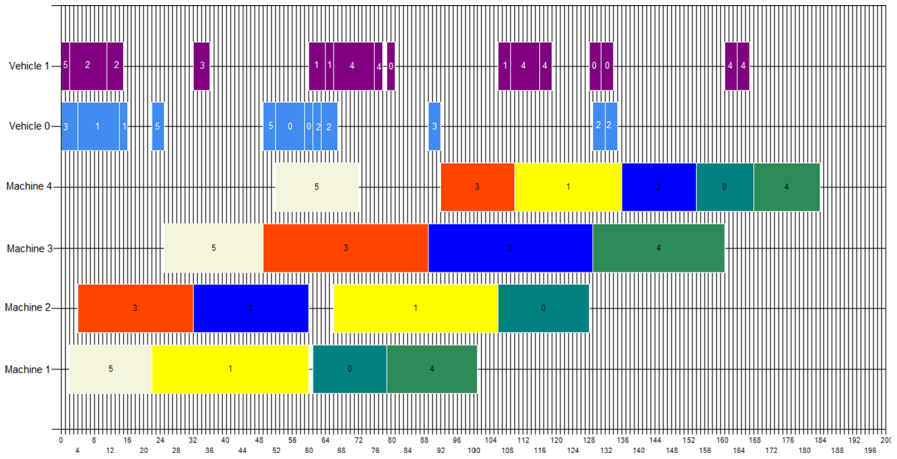

Fig. 5 Gantt diagram of the EX640 instance with makespan=184

Values obtained by our mathematical programming model using the IBM CPLEX Solver are presented in the column "cplex". Note that cplex results marked with '-' mean that their executions have been interrupted once they exceeded 1 hour.

The column "BKS" (for Best Known Solutions) references the best literature results obtained, on the same benchmark instances, by Zheng et al. 2014 in [43] for Table 2 and by Lacomme et al. 2013 in [44] for Table 3.

To allow the comparison of our asynchronous local search VNS and its sequential alternative, two results' sets are provided :

1. Asynchronous LS VNS : in which we provide our proposed model results,
2. Sequential LS VNS : in which a modified version of the Transport local search function is used by omitting the line 4 of the Algorithm 3 and inheriting the JSP schedule value for y as a function input. Furthermore, both Transport and JSP local search routines are called sequentially from upper level (The call order is not important).

For each VNS implementation, "BFS" (for Best Found Solution), "Time" and "Gap" columns represent respectively the minimum attained makespan within five VNS runs, the consumed CPU time for the best found makespan, and the relative difference percentage between BFS and BKS using the formula described in [45]. Note that a time value=0 means that the BFS value has been obtained from the initial random solution. Benchmark original results are given as a reference in the column "Bilge et al. 1995".

An example of the Gantt diagram generated for the best-found solution for the instance EX640 is given by Fig. 5.

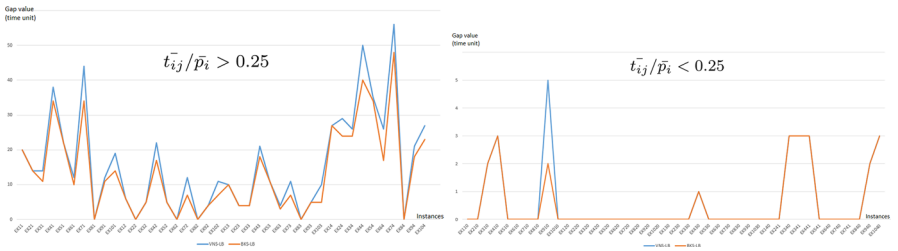

Fig. 6 Gap between VNS/BKS results and LB values for both [19] instances groups

6 Discussion

Our asynchronous approach offers good results compared to those provided by the previous experiments. In fact, our model behaves in different ways for both instances group. For instances with $t_{ij}/\bar{p}_i$ ratio < 0.25, our proposed VNS provides almost identical results as the BKS with global deviation equal to 0,04%. For instances with $t_{ij}/\bar{p}_i$ ratio > 0.25 the deviation average of our proposed VNS from BKS is 2,26%. This indicates that increasing the $t_{ij}/\bar{p}_i$ ratio deteriorate the quality of the obtained results which can be explained by a possible limited number of solutions (i.e. JSP-Transport string combinations) that provide the best C_{max}. Also, findings show clearly that the asynchronous LS technique has an indisputable advantage over the sequential LS implementation as the quality and the calculation time of the obtained results in both techniques are remarkably enhanced.

The analysis of variance (one-way ANOVA) has been conducted to evaluate the statistical difference between the obtained results, represented by BFS columns from one side, and the literature represented by BKS column from the other. It was applied on two data groups of both asynchronous and sequential local search results from Tables 2 and 3. As we can see in the last line of both tables, all the calculated p-values are greater than 5%; this means that the difference between BFS and BKS results is not significant. Also, in one-way ANOVA, the closer the p-value is to 100%, the more similar are the compared data groups. Thus, results obtained in the Table 3 are better than those in Table 2 and, in a same way, the asynchronous LS implementation performs better than the sequential one in both tables which reinforces its usefulness.

Another observation in regard to the whole set behavior. Fig. 6 provides graphs of gaps between the obtained C_{max} results in either the literature (in red) or our VNS approach (in blue) and their related new lower bounds in both $t_{ij}/\bar{p}_i > 0.25$ and < 0.25 instances groups respectively where a gap=0 means that the obtained C_{max} is equal to the lower bound of the instance. By analyzing both graphs, one can deduce that the quality of the obtained C_{max} for the studied instances depends on the characteristics of the target manufacturing layout. In fact, the calculated C_{max} is closer to its lower bound when it concerns an instance that belongs to both layouts 2 and 3 while instances related to layout 4 give the maximum gaps values. This can be related to the nature of the layout, as layouts 4 and 1 are more complex than layouts 2 and 3 [19]. In other hand, job set 8 provides always Gap = 0 in the eight cases whatever the layout, contrariwise, job set 4 results the greatest gaps per layout group for all cases. A comparison between

both job sets reveals that in the process routes of job set 8, all six jobs visit the same machines in the same sequence, while, job set 4 is the unique job set that contains jobs with returning to a previsited machines in their process routes [19].

7 Conclusion

The purpose of this paper is to demonstrate the effectiveness of the VNS in the Job shop scheduling problem with transport constraints. We have proposed a mathematical programming formulation for the scheduling problem, an improved lower bound with new calculation method and used a novel local search technique to enhance the quality of our VNS outcomes. The comparison of the obtained results with previous approaches indicates the effectiveness of the proposed VNS and the lower bound calculation method.

As a prospect, the proposed VNS can be extended to integrate energy-related aspects [46] and it can also incorporate other performance criteria in the schedule cost calculation routine. The obtained results can be improved through an in-depth study of the schedule representation, the neighborhood structures, or the implemented local search process [47]. Also, based on the obtained lower bound values and the optimized makespan results for the studied problems, a predictive approach can be proposed to highlight the implicit relation between the complexity degree of the input manufacturing plant description and scheduling results quality.

Acknowledgements Authors would like to thank the Directorate General for Scientific Research and Technological Development (DGRSDT), an institution of the Algerian Ministry of Higher Education and Scientific Research, for their support on this work.

References

1. Martinez-Barbera, H., Herrero-Perez, D.: Development of a flexible AGV for flexible manufacturing systems. Ind. Robot **37**, 459–468 (2010)
2. Pu, P., Hughes, J., Integrating AGV schedules in a scheduling system for a flexible manufacturing environment, In: Proceedings of the 1994 IEEE International Conference on Robotics and Automation, pp 3149–3154 (1994)
3. Zhang, J., Ding, G., Zou, Y., Qin, S., Fu, J.: Review of job shop scheduling research and its new perspectives under Industry 4.0. J. Intell. Manuf **30**, 1809–1830 (2019)
4. Wojakowski, P.: Research study of state-of-the-art algorithms for flexible job-shop scheduling problem. Czasopismo Techniczne. 1-M (5) 2013, 381–388 (2013). https://doi.org/10.4467/2353737XCT.14.048.1974

5. Jones, A., Rabelo, L.C., Sharawi, A.T.: Survey of Job Shop Scheduling Techniques. In: Webster, J.G. (ed.) Wiley Encyclopedia of Electrical and Electronics Engineering W3352. Wiley, USA, Hoboken, NJ (1999)
6. Sevaux, M., Sörensen, K., A genetic algorithm for robust schedules in a one-machine environment with ready times and due dates. 4OR. 2, 129–147 (2004). https://doi.org/10.1007/s10288-003-0028-0
7. Chaudhry, I.A., Khan, A.A.: A research survey: review of flexible job shop scheduling techniques. Int. Trans. Oper. Res. **23**, 551–591 (2016)
8. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**, 1097–1100 (1997)
9. Sevkli, M., Aydin, M.E.: A Variable Neighbourhood Search Algorithm for Job Shop Scheduling Problems. In: Gottlieb, J., Raidl, G.R. (eds.) Evolutionary Computation in Combinatorial Optimization, pp. 261–271. Springer, Berlin (2006)
10. Roshanaei, V., Naderi, B., Jolai, F., Khalili, M.: A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. Future. Gener. Comp. Sy **25**, 654–661 (2009)
11. Karimi, H., Rahmati, S.H.A., Zandieh, M.: An efficient knowledge-based algorithm for the flexible job shop scheduling problem. Knowl. Based. Syst **36**, 236–244 (2012)
12. Zhang, G., Zhang, L., Song, X., Wang, Y., Zhou, C.: A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem. Cluster Comput. **22**, 11561–11572 (2019). https://doi.org/10.1007/s10586-017-1420-4
13. Zhao, F., Qin, S., Yang, G., Ma, W., Zhang, C., Song, H.: A differential-based harmony search algorithm with variable neighborhood search for job shop scheduling problem and its runtime analysis. IEEE Access **6**, 76313–76330 (2018)
14. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness, 236–244. Bell telephone Laboratories Inc, USA (1979)
15. Hurink, J., Knust, S.: A tabu search algorithm for scheduling a single robot in a job-shop environment. Discrete. Appl. Math **119**, 181–203 (2002)
16. Hurink, J., Knust, S.: Tabu search algorithms for job-shop problems with a single transport robot. Eur. J. Oper. Res **162**, 99–111 (2005)
17. Brucker, P., Knust, S.: Lower bounds for scheduling a single robot in a job-shop environment. Ann. Oper. Res **115**, 147–172 (2002)
18. Job-shop scheduling with transportation. [Online]. Available: http://www2.informatik.uni-osnabrueck.de/kombopt/data/robot/. [Accessed: 24-May-2019]
19. Bilge, Ü., Ulusoy, G.: A time window approach to simultaneous scheduling of machines and material handling system in an FMS. Oper. Res. **43**, 1058–1070 (1995)
20. Caumond, A., Lacomme, P., Moukrim, A., Tchernev, N.: An MILP for scheduling problems in an FMS with one vehicle. Eur. J. Oper. Res. **199**, 706–722 (2009)
21. Ham, A.: Transfer-robot task scheduling in job shop. Int. J. Prod. Res. (2020). https://doi.org/10.1080/00207543.2019.1709671
22. Zeng, C., Tang, J., Yan, C.: Scheduling of no buffer job shop cells with blocking constraints and automated guided vehicles. Appl. Soft Comput. **24**, 1033–1046 (2014)
23. Bürgy, R., Gröflin, H.: The blocking job shop with rail-bound transportation. J. Comb. Optim. **31**, 152–181 (2016)
24. Deroussi, L., Gourgand, M., Tchernev, N.: A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. Int. J. Prod. Res **46**, 2143–2164 (2008)
25. Zhang, Q., Manier, H., Manier, M.-A.: A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. Comput. Oper. Res. **39**, 1713–1723 (2012)
26. Deroussi, L.: A Hybrid PSO Applied to the Flexible Job Shop with Transport. In: Siarry, P., Idoumghar, L., Lepagnot, J. (eds.) Swarm Intelligence Based Optimization, pp. 115–122. Springer International Publishing, Cham (2014)
27. Hernández-Gress, E.S., Seck-Tuoh-Mora, J.C., Hernández-Romero, N., Medina-Marín, J., Lagos-Eulogio, P., Ortíz-Perea, J.: The solution of the concurrent layout scheduling problem in the job-shop environment through a local neighborhood search algorithm. Expert Syst. Appl. **144**, 113096 (2020)
28. Ebrahimi, A., Jeon, H.W., Lee, S., Wang, C.: Minimizing total energy cost and tardiness penalty for a scheduling-layout problem in a flexible job shop system: a comparison of four metaheuristic algorithms. Comp. Ind. Eng. **141**, 106295 (2020)
29. Özgüven, C., Özbakır, L., Yavuz, Y.: Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Appl. Math. Model **34**, 1539–1548 (2010)

30. Logical constraints for CPLEX. https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.9.0/ilog.odms.ide.help/OPL_Studio/opllangref/topics/opl_langref_constraints_types_logical.html (accessed Aug. 21, 2020)
31. Cheng, R., Gen, M., Tsujimura, Y.: A tutorial survey of job-shop scheduling problems using genetic algorithms-I. Comput. Ind. Eng. Represent. **30**, 983–997 (1996)
32. Dhaenens, C., Jourdan, L.: Optimization and big data. In: Metaheuristics for Big Data. ISTE Ltd and Wiley, Inc., London, UK and Hoboken, USA, pp 1–21 (2016)
33. Sevkli, M., Aydin, M.E.: Parallel variable neighbourhood search algorithms for job shop scheduling problems. IMA J. Manag. Math. **18**, 117–133 (2007)
34. Crainic, T.G., Gendreau, M., Hansen, P., Mladenović, N.: Cooperative parallel variable neighborhood search for the p-median. J. Heuristics. **10**, 293–314 (2004)
35. Djenić, A., Radojičić, N., Marić, M., Mladenović, M.: Parallel VNS for bus terminal location problem. Appl. Soft Comput. **42**, 448–458 (2016)
36. Türkyılmaz, A., Bulkan, S.: A hybrid algorithm for total tardiness minimisation in flexible job shop: genetic algorithm with parallel VNS execution. Int. J. Prod. Res. **53**, 1832–1848 (2015). https://doi.org/10.1080/00207543.2014.962113
37. García-López, F., Melián-Batista, B., Moreno-Pérez, J.A., Moreno-Vega, J.M.: The parallel variable neighborhood search for the p-median problem. J. Heuristics. **8**, 375–388 (2002)
38. Aydin, M.E., Sevkli, M.: Sequential and Parallel Variable Neighborhood Search Algorithms for Job Shop Scheduling. In: Xhafa, F., Abraham, A. (eds.) Metaheuristics for Scheduling in Industrial and Manufacturing Applications, pp. 125–144. Springer, Berlin (2008)
39. Hansen, P., Mladenovic, N.: A Tutorial on Variable Neighborhood Search. Groupe d'études et de recherche en analyse des décisions, HEC Montréal (2003)
40. Hansen, P., Mladenović, N.: First versus best improvement. An emp. study Discrete App. Math. **154**, 802–817 (2006)
41. Hansen, P., Mladenović, N.: Variable Neighborhood Search. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies, 211–238. Springer, US, Boston, MA (2005)
42. Ulusoy, G., Sivrikaya-Şerifoğlu, F., Bilge, Ü.: A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. Comput. Oper. Res. **24**, 335–351 (1997)
43. Zheng, Y., Xiao, Y., Seo, Y.: A tabu search algorithm for simultaneous machine/AGV scheduling problem. Int. J. Prod. Res. **52**, 5748–5763 (2014)
44. Lacomme, P., Larabi, M., Tchernev, N.: Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. Int. J. Prod. Econ. **143**, 24–34 (2013)
45. Bennett, J., Briggs, W.: Chapter 3 Numbers in the real world. In: Using & Understanding Mathematics: A Quantitative Reasoning Approach. Pearson. Boston, MA, p. 160 (2019)
46. Giret, A., Trentesaux, D., Prabhu, V.: Sustainability in manufacturing operations scheduling: a state of the art review. J. Manuf. Syst **37**, 126–140 (2015)
47. Abderrahim, M., Bekrar, A., Trentesaux, D., Aissani, N., Bouamrane, K.: Manufacturing 4.0 operations scheduling with AGV battery management constraints. Energies **13**, 4948 (2020)

Affiliations

Moussa Abderrahim[1] · Abdelghani Bekrar[2] · Damien Trentesaux[2] · Nassima Aissani[3] · Karim Bouamrane[1]

✉ Moussa Abderrahim
abderrahim.moussa@edu.univ-oran1.dz

Abdelghani Bekrar
abdelghani.bekrar@uphf.fr

Damien Trentesaux
damien.trentesaux@uphf.fr

Nassima Aissani
aissani.nassima@univ-oran2.dz

Karim Bouamrane
bouamrane.karim@univ-oran1.dz

1 Laboratoire d'Informatique d'Oran (LIO), Université Oran1, Oran, Algeria

2 LAMIH, UMR, CNRS 8201, UPHF, Valenciennes, France

3 Laboratoire de l'Ingénierie de la Sécurité Industrielle et du Développement Durable, Université Oran2, Oran, Algeria