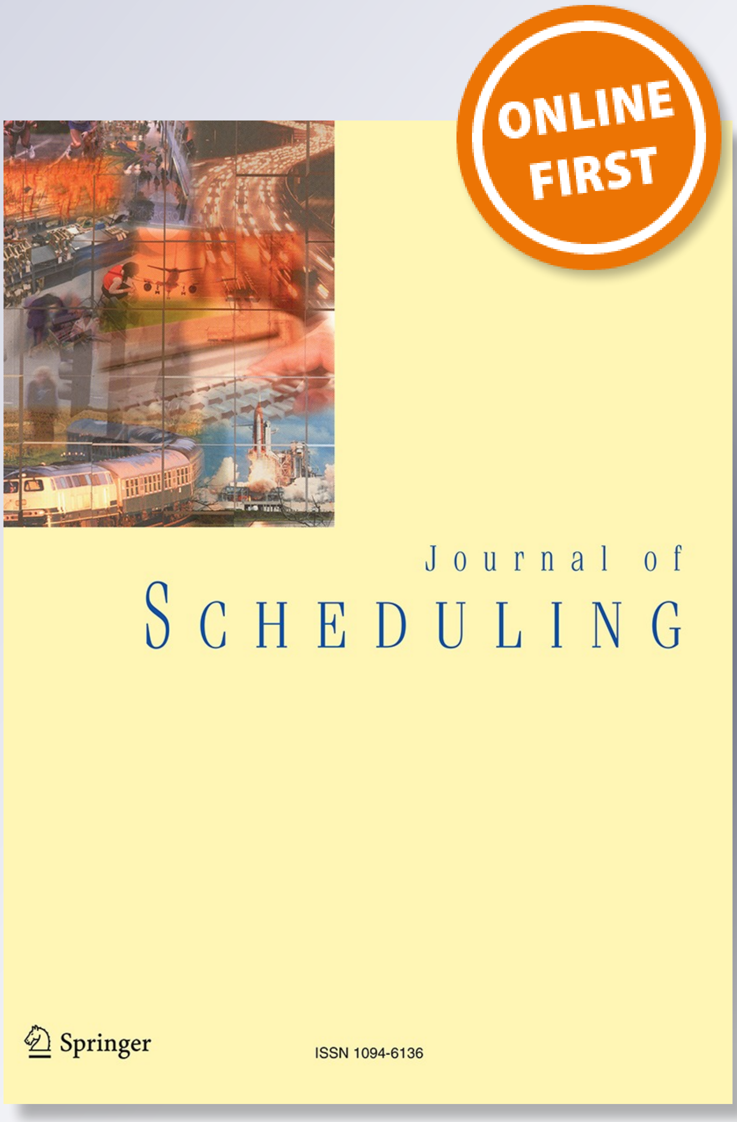
# *Ant colony systems for the single-machine total weighted earliness tardiness scheduling problem*

**Rym M'Hallah & Ali Alhajraf**

**Journal of Scheduling**

ISSN 1094-6136

J Sched
DOI 10.1007/s10951-015-0429-x



J Sched
DOI 10.1007/s10951-015-0429-x

# Ant colony systems for the single-machine total weighted earliness tardiness scheduling problem

**Rym M'Hallah**[1] · **Ali Alhajraf**[2]

**Abstract** Single-machine weighted earliness tardiness scheduling is a prevalent problem in just-in-time production environments. Yet, the case with distinct due dates is strongly NP-hard. Herein, it is approximately solved using ASV, an ant colony-based system with a reduced number of ants and of colonies and with daemon actions that explore the search space around the ants using a variable neighborhood search (VNS). The numerical investigation provides computational proof of the utility of the daemon actions. In addition, it infers that these latter can be applied either to the initial or to subsequent colonies. Furthermore, it highlights the importance of using ant colony optimization as the multiple restart engine of VNS. Finally, it shows that ASV obtains the optimum for most small-sized instances. It has a 0.2 % average deviation from the optimum over all benchmark instances.

**Keywords** Ant colony systems · Weighted earliness tardiness · Single machine · Simulated annealing · Variable neighborhood search

## 1 Introduction

Quick and prompt response to customers' demand is the key to success in many manufacturing and service industries such as nursing, operating theaters, transportation, communication, delivery, apparel, glass, and wood manufacturing. This is particularly the case of evolving markets where competing manufacturers have shifted from mass to just-in-time production. Each strives to satisfy its customers' demand on-time. Missing a due-date may result in the loss of the client's goodwill, a rushed delivery, or a heavy penalty whereas preparing the order too early may cause undue handling, storage, and in some instances product deterioration. For example, in furniture manufacturing, cutting a set of wooden items too early or too late can disrupt the work flow of subsequent stages and affect the on-time delivery of the final products. This is also prevalent in service industries such as health care. For example, the early completion of an elective surgery may result in blocking the operating room and delaying the next surgeries while a tardy completion causes over time costs of surgical staff and post anesthesia care unit nurses [15]. Similarly, a nurse has a predefined set of tasks with each task having a fixed duration and due date. S/he has to complete each task on time and minimize its societal loss (i.e., its total cost to the patient and hospital). These tasks may include administrating medical treatments, observing and monitoring patients' conditions, escorting patients for X-ray or lab work, maintaining records, being present during surgical procedures, etc. [16]. Thus, any schedule of a set of tasks should strive to minimize the total weighted earliness and tardiness.

### 1.1 Problem definition

Scheduling a set $N = \{1, \ldots, n\}$ of $n$ jobs on a single machine with the objective of minimizing their total weighted earliness tardiness is denoted $1|d_j|\sum \underline{w}_j E_j + \overline{w}_j T_j$. Each job is characterized by its processing time $p_j$, due date $d_j$, unit cost of earliness $\underline{w}_j$, and unit cost of tardiness $\overline{w}_j$. All processing times and due dates are known, deterministic, integer, and not necessarily equal. When in-process, a job is

✉ Rym M'Hallah
rymmha@yahoo.com; rym.mhallah@ku.edu.kw

1 Department of Statistics and Operations Research, College of Science, Kuwait University, P.O. Box 5969, 13060 Safat, Kuwait

2 Biomedical Sciences Department, College of Nursing, Public Authority for Applied Education and Training, P.O. Box 23167, 13062 Safat, Kuwait

Published online: 08 April 2015

never preempted. The machine is ready at time zero. Once it starts, the machine is never idle until it completes the $n$ jobs. A wide range of industrial and health care applications prohibit the insertion of idle time for various reasons: exorbitant cost of idle time, infeasibility, unavailable contingent resources, high demand, and expensive operational/setup costs. When the due dates are distinct, the insertion of idle time when technologically feasible can reduce the objective function value [21]. In such a case, the application of a timing algorithm inserts idle time optimally.

In addition to its prevalence as a real-world problem, the $1|d_j|\sum \underline{w}_j E_j + \overline{w}_j T_j$ arises as a relaxation of scheduling problems with more complex manufacturing environments or with additional constraints as in a job shop [3]. Yet, it is strongly NP-hard [25]. That is, no efficient algorithm can solve medium or large-sized instances. Exact techniques vary from branch and bound [9] to a recent dynamic programming-based exact method [22] that solves problems with up to 300 jobs. Muller-Hannemann and Sonnikow [17] highlight the absence of approximation results, and show that the problem is extremely hard to approximate using a polynomial-time algorithm with a guaranteed worst-case performance unless P = NP. A feasible solution to the problem is a permutation of the $n$ jobs. Thus, the search space consists of $n!$ distinct solutions.

This paper approximately solves the $1|d_j|\sum \underline{w}_j E_j + \overline{w}_j T_j$ problem using an ant colony system. This latter uses a limited size population. Its ants ensure the exploration while the external daemon actions guarantee the exploitation. The daemon actions consist of a variable neighborhood search (VNS). The motivation of this design follows.

### 1.2 Motivation of solution approach

M'Hallah [12] describes a positional mathematical programming model for the *unweighted* earliness tardiness single-machine scheduling problem. She tackles the problem using dominance criteria, hill climbing, simulated annealing (SA), and a genetic algorithm (GA), with all the four components being essential to the convergence of the algorithm toward near-global optima. She infers that the most successful heuristics are those dotted with complementary exploration and exploitation mechanisms. This complementarity is particularly essential for this problem. To be on time, many jobs have to compete for the same time slots. In addition, upper bounds obtained via constructive heuristics and lower bounds issued of mathematical programming relaxations have large optimality gaps.

M'Hallah and Alhajraf [14] address the *unweighted* problem using both an ant system and an ant system augmented by a two-opt search around the best ants of each generation. They undertake an extensive computational study to tune the parameters of their heuristics. They conclude that their ant colony system yields better solutions as the numbers of ants and of colonies increase. This of course occurs at the cost of a large runtime. They recommend the use of 5000 ants and 300 generations as a tradeoff between solution quality and runtime. They attribute this unusually large size of the colony to the myopic decisions of the ants when scheduling jobs on the machine. The application of their ant colony system to larger instances is questionable. In fact, their computational investigation is limited to instances with up to 30 jobs. The problem size is confined in part by Cplex's convergence capability toward an exact solution (for assessment) and by the prohibitively large runtime of their heuristics. Their ant colony system yields better solutions than their ant system thanks to the two-opt intensification step that the former offers, hinting to the fundamental role of local search in enhancing the performance of ant colony optimization (ACO) heuristics. In many applications, the initial and updated levels of pheromone can not effectively and efficiently guide the ants' movements unless deduced from local optima that are close to the global one, resulting in the hybridization of ACO with beam search [2], scatter search, tabu search [6], threshold accepting [11], and neighborhood search [1].

Schaller and Valente [20] undertook a comparative study of different heuristics for the *unweighted* earliness tardiness flow shop problem, which is a generalization of the *unweighted* single-machine case. Their GA outperforms deterministic and stochastic neighborhood searches as well as ACO. Yet, M'Hallah [13] showed that a multiple restart VNS outperforms the GA of [20]. VNS investigates different neighborhoods in quest of a (near-) global optimum. It searches immediate neighborhoods via a descent search technique; then, it adopts a more progressive search in neighborhoods that are inaccessible from its current point. It bounces from its current local solution to a new one when it discovers a preferred solution or undertakes a predefined number of successive searches without improvement. Hence, VNS is not a trajectory emulating technique like SA. In addition, it does not risk being trapped in local minima as pairwise exchange local search techniques do. The systematic steepest descent within different neighborhood structures allows VNS to outperform other search heuristics. It provides pertinent knowledge about the problem behavior and characteristics. In fact, VNS stipulates that any local optimum reveals some useful information on the characteristic of the global one. In many instances, the local and global optima share the same values of many variables. However, it is hard to predict which variables they are.

Despite the robustness of VNS with respect to the starting point and its convergence toward near-optima within the first few iterations of its outer loop, the multiple restart enhances its chances of identifying the global optimum. In lieu of a multiple restart, Tasgetiren et al. [23] opt for a particle swarm optimization heuristic (for the minimal makespan/flowtime permutation flow shop). That is, the multiple restart can be replaced by a population-based heuristic such as GA or ACO.

The choice of GA might turn out to be computationally expensive especially that the single-machine case is a relaxation of more complex problems, and thus, should have relatively reduced run times. When run with a limited size population, GA does not necessarily converge to near-global minima. In addition, it requires a careful design of the crossover operator.

The use of an ACO seems a more plausible alternative as it is naturally adapted to preserving good blocks of the solution, to maintaining relative orderings of the jobs, and to applying a VNS intensification search. In a hybrid ACO–VNS heuristic, VNS reinforces the choices of the surviving ants of the colony. When each ant is subject to VNS, the knowledge yielded by VNS is reflected in the structure of the ants. It becomes the cumulated learning acquired by the ants, and it is transmitted to the next colony via the pheromone matrix. This matrix preserves and reinforces the good parts of the near optima obtained by VNS and indirectly determines the variables that are common to the ant and to the global optimum. The resulting ant colony system can be perceived as a "guided" multiple neighborhood search initiated from a multitude of focal *points*, where the solutions or ants share some common good characteristics while being mildly different from each other. Because VNS converges rapidly toward near-global minima, the number of restarts need not be very large. Thus, the proposed ant colony system uses limited numbers of ants and of generations.

The choice of a hybrid ACO–VNS heuristic is further motivated by Behnamian et al.'s [1] ACO which outperforms a random key GA for the *unweighted* earliness tardiness hybrid flow shop with setup times. Their ACO subjects every ant of the colony to a stochastic VNS. The stochastic component of the search, corresponding to the outer loop of VNS, is an SA. This choice seems contradictory to the purpose of the outer loop, which is supposed to identify an unexplored area of the search space. The inner loop is a variable neighborhood descent. Their ACO performs better than their random key GA for equal runtime with the GA using 100 chromosomes, but the size of the ant colony is not stated. In addition, it is not clear how well either algorithm performs in absolute terms (i.e., with respect to the optimum).

To build neighborhoods and moves, existing search methods use problem-dependent characteristics such as those involving the rolling backward search algorithm [7] and the dynamic heuristic [10], or those applying priority indices [26] and smallest value position [23], or those using different structures and dominance criteria [12]. However, as the problem at hand is a relaxation to numerous scheduling problems, the neighborhoods are built independently of the scheduling environment while being geared toward earliness tardiness objective function types. Moves from the literature are based on job insertion, job swaps, and job order inversion [19]. Herein, the proposed neighborhoods follow the trend in the literature while targeting efficacy and efficiency.

### 1.3 Contribution and outline

This paper proposes an ACO–VNS heuristic for the $1|d_j|\sum \underline{w}_j E_j + \overline{w}_j T_j$. To highlight the effectiveness of the heuristic, the paper compares the performance of three ant colony systems of limited colony size but different external daemon actions. The first daemon action is a pairwise exchange. It consists of deterministic swaps of a pair of jobs. The second is an SA. It involves a stochastic swap of jobs. The third is a VNS that uses three neighborhoods, as recommended by Rocha et al. [19], in increasing order of complexity and of perturbation size, while limiting the structural modifications of the solutions.

In addition to providing an immediate precedence-based mathematical formulation of the problem, the paper's contribution is threefold. First, it provides computational proof of (i) the importance of the exploitation undertaken by the daemon actions and of (ii) the diversification brought up by the ant colony. Second, it indicates that it is possible to have a "successful"ACO with a limited number of ants and colonies. Third, it suggests that ACO–VNS is a good approximate approach since it yields (near-)global minima for all tested instances with a 1.0020 mean ratio of its solution to the optimum.

This paper is organized as follows. Section 2 provides a mathematical formulation of the problem. Section 3 describes a generic ACO. Section 4 details the proposed ACO systems. Section 5 provides computational support for the design of ACO–VNS highlighting the success of VNS as the intensification mechanism and of ACO as the diversification strategy. In addition, it evaluates the performance of ACO–VNS. Finally, Sect. 6 is a summary.

## 2 Problem formulation

Let job 0 be a fictitious job that precedes the first job on the machine, and $N^0 = N \cup \{0\}$. Let $x_{jk},\ j \in N^0,\ k \in N, k \neq j$, equal 1 if job $j$ immediately precedes job $k$, and 0 otherwise. Let $S_j,\ E_j$, and $T_j,\ j \in N$, denote, respectively, the starting time, earliness, and tardiness of job $j$, and $C_j,\ j \in N^0$, denote the completion time of job $j$. $C_j,\ j \in N$, is the sum of the starting and processing times of job $j$, while $C_0$, the completion time of the fictitious job, corresponds to the starting time of the machine. Job $j,\ j \in N$, is early if $C_j \leq d_j$; its earliness $E_j = \max\{0, d_j - C_j\}$. It is tardy if $C_j > d_j$; its tardiness $T_j = \max\{0, C_j - d_j\}$.

$1|d_j|\sum \underline{w}_j E_j + \overline{w}_j T_j$ is modeled as an immediate precedence-based mixed integer program with $n^2$ binary variables, $4n + 1$ positive variables, and $(5n^2 + 7n + 2)/2$ functional constraints:

$$z = \min \sum_{j \in N} \underline{w}_j E_j + \overline{w}_j T_j \quad (1)$$

$$C_j - d_j = T_j - E_j \quad j \in N \quad (2)$$

$$C_j = S_j + p_j \quad j \in N \quad (3)$$

$$S_k + M(1 - x_{jk}) \geq C_j \quad j \in N^0,\ k \in N, k \neq j \quad (4)$$

$$S_k - M(1 - x_{jk}) \leq C_j \quad j \in N^0,\ k \in N, k \neq j \quad (5)$$

$$x_{kj} + x_{jk} \leq 1 \quad j \in N^0,\ k \in N, k < j \quad (6)$$

$$\sum_{\substack{j \in N^0 \\ k \neq j}} x_{jk} = 1 \quad k \in N \quad (7)$$

$$\sum_{\substack{k \in N \\ k \neq j}} x_{jk} \leq 1 \quad j \in N^0 \quad (8)$$

$$x_{jk} \in \{0, 1\} \quad j \in N^0,\ k \in N, k \neq j \quad (9)$$

$$S_j,\ C_j,\ E_j,\ T_j \quad \text{integer} \quad j \in N \quad (10)$$

$$C_0 \quad \text{integer}, \quad (11)$$

where $M$ is a large positive number such that $M \to \infty$.

Equation (1) minimizes the total weighted earliness tardiness. Equation (2) sets the lateness of job $j$ as the difference between its completion time and its due date. The lateness equals the tardiness if the job is tardy, zero if the job is on-time, and the negative of earliness if the job is early. There are $n$ of these equations. Equation (3) sets the completion time of job $j$ equal to the sum of its starting and processing times. There are $n$ of these equations. Equations (4) and (5) set the completion time of $j$ equal to the starting time of $k$ if $j$ immediately precedes $k$ and are redundant otherwise. There are $n^2$ of each of these constraints. Equation (6) reinforces the precedence relations between any pair of jobs $k$ and $j$. Either $k$ immediately precedes $j$ or $j$ immediately precedes $k$, or neither relation holds. These $\frac{n(n+1)}{2}$ constraints reduce the symmetry of the search space, but have no functional utility. Equation (7) forces a job $k \in N$ to have exactly one immediate predecessor. There are $n$ constraints of this type whereas Eq. (8) limits the number of immediate successors of a job $j \in N^0$ to at most one job since the last job scheduled on the machine has no successor. There are $n + 1$ of these constraints. Equation (9) declares $x_{jk}$ as binary variables. Finally, Eq. (10) declares the starting time, completion time, earliness and tardiness of job $j$, $j \in N$, positive integers while Eq. (11) sets the starting time of the machine positive integer. Equations (10) and (11) can be replaced by non-negativity constraints when the processing times and due dates are all integers. In that case, there exists at least one optimal solution with integer values for all the four variables.

When the machine must start at time zero, Eq. (11) is replaced by

$$C_0 = 0 \quad (12)$$

to force the starting time of the machine to be zero. When the jobs have release times, the machine may have to start later than zero to avoid idle time. Subsequently, the above model is augmented by:

$$C_j \geq r_j, \quad (13)$$

$j \in N^0$, where $r_0 = \min_{j \in N}\{r_j\}$. Finally, when idle time is allowed, Eq. (5) is dropped from the model.

## 3 ACO

ACO, which is inspired by the foraging behavior of real ants, is a discrete optimization meta-heuristic that converges to optimality under certain conditions. It has an initialization, an iterative, and a stopping step. The initialization step generates a colony **A** of $m$ ants, where each ant corresponds to a random feasible solution. In the absence of a pheromone trail that guides them, ants move randomly; thus, they follow random paths in the search space.

The colony is updated at every generation $g$ with the new colony reinforcing the "good" traits of the best current solution(s). This reinforcement is guaranteed via the cooperation of the ants. In nature, ants identify the shortest path between their nest and a food source by exchanging information regarding the food source with other ants. The information is channeled amongst the ants via the pheromone each ant deposits during its traveling between the nest and the food source. The more traveled a path is, the more pheromone is deposited on it. Thus, despite the natural evaporation of pheromone, paths with heavy traffic maintain high pheromone levels. They remain attractive to ants in the ants' next ventures for food. Ants might choose to ignore the acquired knowledge (or learned desirability in the form of deposited pheromone) and discover either new paths or paths with low pheromone deposit. When faced with many paths, an ant chooses its path probabilistically, i.e., with a probability that is a function of the amounts of pheromone on the alternative trails and of the worth of each path (according to its own perception). The weighted function value reflects the ant's trade-off between the exploration of new connections and the exploitation of available information.

Once the colony is updated, the algorithm updates the pheromone level of each path of the trail by depositing a pheromone amount that reflects the frequency of usage of that specific path across all ants of the colony. Subsequently, each path is subject to a constant evaporation rate of its pheromone level. Pheromone evaporation limits the chances of stagnation of the algorithm. The aforementioned steps are repeated until the algorithm reaches its stopping criterion: a fixed number of generations $G$, a time limit, or a number of iterations without improvement.

Some variations update the pheromone level at every move of every ant through deposit and/or evaporation (mimicking the effect of time on real life colonies). Others apply daemon actions that are external factors out of the control of ants such as wind, floods, predators, etc. They are similar to environmental phenomena in nature. They vary from local search procedures to fixing approaches to biased pheromone deposit/evaporation [4,8,23].

A survey of ant colony applications to scheduling problems [24] reveals that most approaches (36 out of 54) are based on ant systems, with fewer (10 out 54) using ant colony systems and the rest using Max–Min ant systems. It distinguishes two ways of coding the solution: job to position and job to job, with the first type of coding mostly used in flow shops and the second predominant in all manufacturing environments. Based on this classification, the design of the proposed ACO uses a job to position coding. This choice is prompted by the design of the heuristic which purposely avoids the use of precedence relations between jobs and by the fact that the single-machine environment is a special case of the flow shop. The survey further establishes a set of guidelines for designing ant colony-based approaches. It recommends the use of a dynamic visibility function that is updated as the ant constructs the solution, and emphasizes the importance of the initialization of the pheromone levels. The proposed ACO abides to these recommendations.

## 4 The proposed ant colony systems

An ant $a \in \mathbf{A}$ is constructed as follows. Initially, $a$ has all its positions empty and its jobs free, i.e., its set of already positioned jobs $\overline{N}_a = \emptyset$ and its set of free jobs $N_a = N$. The sets $\overline{N}_a$ and $N_a$ are complementary: $N^a \bigcup \overline{N}_a = N$ and $N_a \cap \overline{N}_a = \emptyset$. When it moves from a position $[i]$ to a position $[i+1]$, ant $a$ assigns a job $j \in N_a$ to its position $[i]$, $i = 1, \ldots, n$. It removes $j$ from $N_a$ and appends it to $\overline{N}_a$. In this sense, $N_a$ constitutes the candidate jobs for position $[i]$. Ant $a$ stops its moves when it schedules all $n$ jobs. Using this definition of ants, the ACO's three steps, summarized in Algorithm 1, proceed as follows.

**Algorithm 1** ACO pseudo code

```
Input
  • $N$, a set of $n$ jobs.
  • $G$, the number of colonies.
  • $m$, the size of the colony (i.e., the number of ants of the colony).
Output
  • A (near-) global optimum $a^*$ of cost $z_{a^*}$.
1. Initialization
  1.1 Set the generation counter $g = 0$.
  1.2 Create an initial colony $\mathbf{A}$ of size $m$.
  1.3 Set the best solution $a^*$ to the ant $a \in \mathbf{A}$ with the least
      weighted earliness tardiness.
  1.4 Initialize the pheromone level $\pi^{(0)}$ using (a subset of) the
      colony $\mathbf{A}$, and set $\pi^{(1)} = \pi^{(0)}$.
2. Iterative Step
  2.1 Set $g = g + 1$.
  2.2 Build a colony of ants while accounting for $\pi^{(g)}$, the acquired
      knowledge, and $\eta^{(g)}$, the attractiveness, and applying a
      dynamic visibility function.
  2.3 Merge the colonies of generations $g$ and $(g-1)$, and retain
      the best $m$ ants.
  2.4 Update the optimal solution $a^*$.
  2.5 Determine the pheromone level $\pi^{(g+1)}$ accounting for
      pheromone evaporation/deposit.
3. Stopping Criterion
  If $g < G$, goto Step 2.
```

*Step 1: Initialization* The initial colony $\mathbf{A}$ (i.e., $g = 0$) consists in $m = |\mathbf{A}|$ random sequences of the $n$ jobs, with each sequence corresponding to an ant $a$ whose weighted earliness tardiness is $z_a$. The best current solution $a^*$ is the ant $a \in \mathbf{A}$ with minimal weighted earliness tardiness: $z_{a^*} = \min_{a \in \mathbf{A}}\{z_a\}$. Building an initial pheromone trail as recommended in [24] uses either $a^*$ or (a portion of) the initial population. This trail constitutes the acquired knowledge that ants will inherit from the experience of previous colonies. Herein, the initial pheromone level of this trail is $\pi^{(0)}$, an $n \times n$ matrix whose entries $\pi_{ij}^{(0)}$ equal the proportion of times job $j$ appears in position $[i]$ in the best $\min\{\lceil m/5 \rceil, 5\}$ ants of the initial colony. $\pi^{(0)}$ constitutes the acquired knowledge at the beginning of generation $g = 1$; therefore, $\pi^{(1)}$ is initialized to $\pi^{(0)}$.

*Step 2: Iterative step* This step consists of five tasks. Task 2.1 of Algorithm 1 increments the generation counter $g$. Task 2.2 builds the $g$th colony, which has $m$ ants. The $k$th ant $a$, $k = 1, \ldots, m$, builds itself by successively filling its positions $[i]$, $i = 1, \ldots, n$. First, it fetches the acquired knowledge $\pi_{ij}^{(g)}$, which is the initial pheromone level corresponding to positioning job $j$ in $[i]$ in colony $g$. Second, it computes the assignment's attractiveness $\eta_{ij}^{(g)}$, which is a relative measure defined as:

$$\eta_{ij}^{(g)} = 1 - \frac{\underline{w}_j E_j + \bar{w}_j T_j}{\max\limits_{j' \in N^a} \{\underline{w}_{j'} E_{j'} + \bar{w}_{j'} T_{j'}\}}. \tag{14}$$

That is, Eq. (14) compares $\underline{w}_j E_j + \bar{w}_j T_j$, the cost of assigning job $j$ to position $[i]$, to $\max_{j' \in N_a}\{\underline{w}_{j'} E_{j'} + \bar{w}_{j'} T_{j'}\}$, the highest possible cost of filling position $[i]$ among all alternative assignments of non-scheduled jobs $j' \in N_a$. The smaller the weighted earliness tardiness of job $j$ when assigned to position $[i]$, the smaller the ratio of the two costs. Ideally, this ratio should be 0, corresponding to scheduling $j$ on time. In any case, the ratio can not exceed 1. Subsequently, $\eta_{ij}^{(g)}$ is bounded by 0 and 1. The closer $\eta_{ij}^{(g)}$ is to 1, the more attractive the assignment of $j$ to $[i]$ is since it corresponds to a near-zero ratio.

Third, ant $a$ fills $[i]$ stochastically as a function of the intensification threshold level $q_0$. It generates a random probability $q$ from the continuous Uniform[0,1] and compares it to $q_0$, where $q_0$ is experimentally set to 0.9.

**When** $q \leq q_0$, it chooses job $j^* \in N_a$ such that $p_{ij^*}^{(g)} = \max_{j \in N_a}\{p_{ij}^{(g)}\}$, where $p_{ij}^{(g)}$ is the learned desirability of filling $[i]$ with $j$. It reflects the degree of suitability of this assignment compared to the sum of the costs associated with filling $[i]$ with the candidate jobs in $N_a$. Its numerator is a pondered product of the acquired knowledge $\pi_{ij}^{(g)}$ and of the local attractiveness $\eta_{ij}^{(g)}$ of such an assignment. The denominator is the sum of the pondered products over all possible assignments. The ratio is therefore bounded between 0 and 1. The higher $p_{ij}^{(g)}$ is, the more desirable the assignment. Specifically,

$$p_{ij}^{(g)} = \frac{[\pi_{ij}^{(g)}]^{\alpha}[\eta_{ij}^{(g)}]^{\beta}}{\sum_{j' \in N_a}[\pi_{ij'}^{(g)}]^{\alpha}[\eta_{ij'}^{(g)}]^{\beta}}, \tag{15}$$

where $\alpha$ and $\beta$ are two parameters that ponder the relative importance of $\pi_{ij}^{(g)}$ versus $\eta_{ij}^{(g)}$. Herein, $\alpha = \beta = 1$.

**When** $q > q_0$, ant $a$ fills $[i]$ with the most locally attractive job $j^*$, i.e., with job $j^*$ such that $\eta_{ij^*}^{(g)} = \max_{j \in N_a}\{\eta_{ij}^{(g)}\}$.

Fourth, ant $a$ updates its set of positioned jobs: $\overline{N}_a = \overline{N}_a \cup \{j^*\}$, and its set of free jobs: $N_a = N_a \setminus \{j^*\}$. Ant $a$ pursues its moves until it assigns all jobs, i.e., until $N_a = \emptyset$. Last, ant $a$ shares its acquired knowledge with the ants of the colony via pheromone evaporation and deposit. Specifically, ant $a$ sets

$$\pi_{ij}^{(g)} = (1-\varphi)\pi_{ij}^{(g)} + \varphi\left(1 - \frac{\underline{w}_j E_j + \bar{w}_j T_j}{z_a}\right), \tag{16}$$

$i \in N$, $j \in N$, where $\varphi \in [0, 1)$ represents both the pheromone evaporation and the pheromone deposit rates. It is herein set to $\varphi = 0.1$. Equation (16) constitutes the dynamic update of the visibility function. It evaporates a portion $\varphi$ of the existing acquired knowledge $\pi_{ij}^{(g)}$, and thus only maintains the portion $(1-\varphi)$ of $\pi_{ij}^{(g)}$. It then augments this knowledge with additional information that is a function of the deposit rate $\varphi$ and of the quality of the assignment of $j$ to $[i]$. In fact, it assesses the contribution of the weighted earliness tardiness of job $j$ with respect to $z_a$, the total weighted earliness tardiness of ant $a$. Ideally, job $j$ is on-time, i.e., has a zero weighted earliness tardiness. When $j$ is not on-time, the smaller its weighted earliness tardiness with respect to $z_a$, the better the assignment is and the higher is the amount of pheromone deposited, and vice versa. This contribution is bounded between 0 and 1; thus, the difference $1 - \frac{\underline{w}_j E_j + \bar{w}_j T_j}{z_a}$ is, in turn, bounded by 0 and 1. Equation (16) illustrates the myopic nature of the ants. Consider the three jobs of Example 1. When in position [2], job 2 has a zero weighted earliness tardiness but causes a high total earliness tardiness (i.e., 260) for ant 1–2–3 (Table 1).

**Table 1** Example 1

| $j$ | 1 | 2 | 3 |
|---|---|---|---|
| $p_j$ | 1 | 10 | 4 |
| $d_j$ | 11 | 11 | 12 |
| $\underline{w}_j$ | 20 | 1 | 30 |
| $\bar{w}_j$ | 30 | 1 | 20 |

Task 2.3 of Algorithm 1 merges the current and the previous colonies, ranks their ants in a non-descending order of their objective function values, and retains the best $m$ ants as the current colony. In fact, it applies a natural selection mechanism where only the fittest survive.

Task 2.4 updates the best solution $a^*$ and $z_{a^*}$ if the best of the retained $m$ ants of the current colony has a lower weighted earliness tardiness than the current local minimum. Task 2.5 applies a global pheromone update for all trails. For every entry $(i, j)$ of matrix $\pi^{(g)}$, it evaporates an amount of pheromone that is proportional to the acquired knowledge $\pi_{ij}^{(g)}$ after the construction of the colony $g$, thus leaving only a portion $1 - \varphi$ of this knowledge. It then deposits another pheromone amount $\Delta_{ij}^{(g)}$, which corresponds to the proportion of times $j$ has been assigned to $i$ in the best $\min\{\lceil m/5 \rceil, 5\}$ ants of the current colony $g$. The resulting knowledge serves as the pheromone matrix of the next colony $g + 1$. The entries of this pheromone matrix are:

$$\pi_{ij}^{(g+1)} = (1-\varphi)\pi_{ij}^{(g)} + \Delta_{ij}^{(g)}. \tag{17}$$

Equation (17) constitutes a simple rank-based update mechanism, where each pheromone value $\pi_{ij}^{(g)}$ is first decreased via evaporation, then increased proportionally by $\Delta_{ij}^{(g)}$ to reinforce the "good" traits of the colony.

*Step 3: Stopping Criterion* If the maximal number of generations $G$ is reached, the algorithm stops and returns $a^*$ as the near-global optimum and $z_{a^*}$ as its value. Otherwise, the algorithm returns to the iterative step.

In summary, Task 2.2 generates a colony of $m$ ants by balancing acquired knowledge with new discoveries. It then merges the ants of colonies $g$ and $g-1$, ranks them, and retains the best $m$ ants. Even though the retained ants are the best ants identified during the last $g$ colonies, they may be suboptimal. Enhancing the quality of the retained $m$ ants may speed the convergence of the ant system toward a near-global optimum. Therefore, at the end of Task 2.3, the ant colony system applies a local search. Sections 4.1–4.3 detail three local searches: pairwise exchange, SA, and VNS.

### 4.1 Pairwise exchange

The pairwise exchange described in Algorithm 2 is a steepest descent. The neighbor $a'$ of $a,\ a \in \mathbf{A}$, is obtained by swapping the jobs in positions $[i_1]$ and $[i_2]$ such that $i_2 - i_1 \leq 4$. It replaces $a$ as the focal point of the search if $z_{a'} < z_a$. The returned ant $a$ has the smallest weighted earliness tardiness among $10(n-2)$ investigated neighbors.

**Algorithm 2** Pseudo code of the pairwise exchange

```
Input
  • $a$, an ant, and its weighted earliness tardiness $z_a$.
Output
  • A local optimum $a$ of cost $z_a$.
Algorithm
  For $\kappa = 1$ to 4
    For $i_1 = 1$ to $n - \kappa$
      For $i_2 = i_1 + 1, i_1 + \kappa$
          1. Generate ant $a'$ by swapping the jobs in positions $[i_1]$ and $[i_2]$.
          2. Compute its weighted earliness tardiness $z_{a'}$.
          3. If $z_{a'} < z_a$, set $a = a'$ and $z_a = z_{a'}$.
      End For
    End For
  End For
```

This pairwise exchange investigates a limited number of neighbors. To be on time, jobs with close due dates have to compete for the same time slot. Changing their order will change their respective completion times and penalties. Consequently, it might reduce the total weighted earliness tardiness. Thus, inverting the order of two jobs with conflicting interests is more useful than inverting any pair of jobs as in [5,18] or inserting a job between a pair of jobs as in [14]. It is possible to determine the jobs with conflicting time windows, but this requires the computation of the overlap window, a costly computation in comparison to the calculation of $\kappa$ deviations. In fact, inverting the order of the jobs in positions $[i_1]$ and $[i_2] = [i_1]+\kappa$ requires the calculation of $\kappa - 1$ completion times and $\kappa$ penalties. It preserves $n - \kappa$ penalties fixed.

This pairwise exchange purposely avoids using sequencing knowledge [26] or dominance properties [12] to make the approach applicable to all types of scheduling environments. It can be implemented by inducing several swaps of jobs as long as the time windows of the concerned pairs do not overlap. Reaching the global optimum may require more than a series of pairwise exchanges.

### 4.2 SA

The SA of Algorithm 3 maintains the computational ease of the pairwise exchange. It substitutes ant $a,\ a \in \mathbf{A}$, by the local minimum $a^*$ in the colony. It uses a $T_0 = 1.1$ initial temperature of the annealing process, yielding a $0.4 = \exp^{-1/T_0}$ initial probability of acceptance of a non-improving solution. The outer loop determines the neighborhood size $\kappa$, and serves as the stopping criterion (prefixed to 4 plateaus). The inner loop investigates the neighbors of $a$. Step 1 generates a neighbor $a'$ of ant $a$ by swapping pairs of jobs in positions $[i_1]$ and $[i_2]$ of $a$ such that $[i_2] = [i_1] + 1, \ldots, [i_1] + \kappa$, and $i_1 = 1, \ldots, n - \kappa$. If $a'$ improves $a$, Step 2 moves the focal point of the search to $a'$, and updates the local minimum $a^*$ setting it equal to $a'$, if need be. Otherwise, Step 2 selects a random real $u$ from the uniform(0,1), and compares it to $\exp^{-1/T_\kappa}$, the probability of acceptance of a non-improving solution at temperate $T_\kappa$ of the current plateau $\kappa$. The threshold probability of acceptance decreases at every new plateau $\kappa + 1:\ T_{\kappa+1} = 0.95T_\kappa$.

**Algorithm 3** Pseudo code of SA

```
Input
  • $a$, an ant, and its weighted earliness tardiness $z_a$.
  • $T_0$, the initial temperature of the annealing process.
Output
  • A local optimum $a^*$ of cost $z_{a^*}$.
Algorithm
Set the current best solution $a^* = a$, and its cost $z_{a^*} = z_a$.
For $\kappa = 1$ to 4
  Calculate the temperature of plateau $\kappa :\ T_{\kappa+1} = 0.95 * T_\kappa$.
  For $i_1 = 1$ to $n - \kappa$
    For $i_2 = i_1 + 1$ to $i_1 + \kappa$
      1. Generate neighbor $a'$ of ant $a$ by swapping the jobs in positions $[i_1]$ and $[i_2]$.
      2. Compute $z_{a'}$, the weighted earliness tardiness of $a'$.
      2. If $z_{a'} < z_a$,
            • set $a = a'$ and $z_a = z_{a'}$;
            • if $z_{a'} < z_{a^*}$, set $a^* = a'$ and $z_{a^*} = z_{a'}$;
        Else
            • draw a random number $u$ from the uniform (0,1);
            • If $u < \exp^{-1/T_\kappa}$, set $a = a'$ and $z_a = z_{a'}$.
        End If
    End For
  End For
End For
```

### 4.3 VNS

The VNS of Algorithm 4 applies a more extensive intensification to ant $a,\ a \in \mathbf{A}$, than the deterministic and stochastic pairwise exchange. It considers very large neighborhoods obtained by swapping every pair of jobs, inserting a job between every pair of jobs, and inserting a pair of jobs after every job. Unlike the deterministic and stochastic intensifications where non-improving moves are automatically discarded or adopted probabilistically, VNS accepts such moves when the search stagnates. In such a case, it changes the size or the structure of its neighborhood. It presumes that a global minimum is a local minimum over all neighborhood types, that all local minima are close to each other, and that a local minimum for one specific neighborhood is not necessarily so for a differently structured neighborhood.

**Algorithm 4** Pseudo code of VNS

```
Input
  • $a$, an ant and its weighted earliness tardiness $z_a$.
Output
  • $a^*$, a local optimum of cost $z_{a^*}$.
Algorithm
  Set $a^* = a$, and its cost $z_{a^*} = z_a$.
  Set $a'$, the current focal point of VNS to $a$, and its cost $z_{a'} = z_a$.
  For $\ell = 1$ to 4
    For $\kappa = 1$ to 3
      1. Find the neighbor $\dot{a} \in \mathcal{N}_\kappa(a')$ with the minimal weighted
         earliness tardiness $z_{\dot{a}}$.
      2. If $z_{\dot{a}} < z_{a'}$,
           • set $a' = \dot{a}$, $z_{a'} = z_{\dot{a}}$ and $\kappa = 1$;
           • if $z_{\dot{a}} < z_{a^*}$, set $a^* = \dot{a}$ and $z_{a^*} = z_{\dot{a}}$.
         Else
           • set $\kappa = \kappa + 1$.
         End If
    End For
    If $\ell < 4$, shake $a'$.
  End For
```

VNS consists of two loops. The inner loop starts from the current focal point $a'$. It sets $\kappa = 1$, and chooses $\dot{a}$, the best neighbor in its neighborhood $\mathcal{N}_1(a')$. When $\dot{a}$ improves $a'$, the inner loop changes its focal point to $\dot{a}$ by setting $a' = \dot{a}$, and sets its neighborhood type $\kappa = 1$. In addition, it checks whether $\dot{a}$ enhances the best solution $a^*$. On the other hand, when $\dot{a}$ does not improve $a'$, the inner loop maintains its current focal point $a'$ and increments its neighborhood counter setting $\kappa = \kappa + 1$. It repeats this iterative step as long as $\kappa \leq 3$. In fact, it uses three neighborhood types.

When $\kappa = 1$, the search considers every neighbor $\dot{a}$ obtained by swapping any pair of jobs in positions $[i_1]$ and $[i_2]$, $i_1 = 1, \ldots, n-1$, $i_2 = i_1+1, \ldots, n$, of $a$. There are $n(n-1)/2$ such neighbors.

When $\kappa = 2$, the search is a 2-opt. It enumerates each neighbor $\dot{a}$ obtained by reversing the order of the jobs between any pair of jobs $j$ and $j'$, $j \in N$, $j' \in N \setminus \{j\}$. There are $(n-2)^2$ such neighbors.

When $\kappa = 3$, the search is an OR-opt. It generates every neighbor $\dot{a}$ obtained by inserting the pair of successive jobs $(j, j+1)$ in reverse order after every job $j'$ of $a$. There are $(n-2)^2$ neighbors of this kind.

These neighborhood types preserve a certain portion of the sequence, and induce relatively small perturbations. However, they require the re-evaluation of the weighted earliness tardiness of a large number of jobs. The "size" of the perturbation increases as $\kappa$ increases.

When the inner loop uses the three neighborhoods, it returns control to the outer loop, which changes the current focal point of the search. The outer loop serves both as a shaking procedure and as a stopping condition. Its first iteration sets ant $a$ as the focal point. Its succeeding iterations apply a shaking procedure to the best current local minimum $a'$, re-centering the search around the perturbed solution. The shaking procedure randomly selects four pairs of jobs of $a'$ and swaps them simultaneously. These moves may deteriorate the weighted earliness tardiness of $a'$.

## 5 Computational results

The computational investigation uses the benchmark instances of [22] available from http://turbine.kuee.kyoto-u.ac.jp/~tanaka/SiPS/. They correspond to sets of five instances with $n = 40, 50, 100, 150, 200, 250, 300$, $\tau = 0.2, 0.4, 0.6, 0.8, 1.0$, and $\rho = 0.2, 0.4, 0.6, 0.8, 1.0$, where $\tau$ is the tardiness factor and $\rho$ is the range of due dates. These instances have know optima $z^*$, obtained within a runtime $t^*$. The exact algorithm of [22] was ran on a Pentium IV 524 Mb. For consistency, all heuristics are run on a similar platform. Section 5.1 computationally justifies the design of ACO–VNS while Sect. 5.2 assesses its performance.

### 5.1 Importance of intensification and diversification

This section investigates the role of daemon actions, of the initial pheromone level, and of the ACO diversification.

#### *5.1.1 Role of daemon actions*

Consider AS, an ant system without daemon actions, and ASD, the ant system with pairwise exchange. For $n = 40$ and 50, we run AS with various population sizes: $m = 5, 10, 20, 50, 100, 1000$, and number of generations $G = 5, 10, 20, 50, 100, 1000$, and ASD with $m' = 5$ and $G' = 5$. Let $z$ and $z_1$ denote the weighted earliness tardiness of AS and ASD, respectively, and $t$ and $t_1$ their respective run times. Table 2 summarizes the results.

$\frac{z}{z^*}$ improves as $G$ and $m$ increase, but remains very large even for $m = 1000$ and $G = 1000$. On the other hand, $\frac{z_1}{z^*}$ averages 1.05 over all 250 instances with the minimum and the first quartile both equaling 1.00. The median is also low.

**Table 2** Mean $\frac{z}{z^*}$ as a function of colony size, number of generations, and problem size

| $n$ | $G$ | $m$ | | | | | |
|---|---|---|---|---|---|---|---|
| | | 5 | 10 | 20 | 50 | 100 | 1000 |
| 40 | 5 | 3.26 | 3.19 | 3.12 | 3.00 | 2.91 | 2.67 |
| | 10 | 3.20 | 3.12 | 3.05 | 2.93 | 2.84 | 2.62 |
| | 20 | 3.14 | 3.05 | 2.98 | 2.86 | 2.78 | 2.58 |
| | 50 | 3.04 | 2.95 | 2.86 | 2.78 | 2.70 | 2.51 |
| | 100 | 2.96 | 2.87 | 2.80 | 2.70 | 2.64 | 2.45 |
| | 1000 | 2.73 | 2.64 | 2.60 | 2.51 | 2.45 | 2.31 |
| 50 | 5 | 3.63 | 3.58 | 3.45 | 3.34 | 3.25 | 3.01 |
| | 10 | 3.59 | 3.49 | 3.40 | 3.26 | 3.17 | 2.97 |
| | 20 | 3.51 | 3.40 | 3.31 | 3.18 | 3.10 | 2.91 |
| | 50 | 3.40 | 3.30 | 3.19 | 3.09 | 3.01 | 2.86 |
| | 100 | 3.32 | 3.20 | 3.12 | 3.03 | 2.99 | 2.82 |
| | 1000 | 3.05 | 3.00 | 2.92 | 2.86 | 2.81 | 2.66 |

It equals 1.02 while the third quartile does not exceed 1.07. A paired statistical hypothesis test infers that there is statistical proof that the mean ratio $\frac{z_1}{z^*}$ is less than the mean ratio $\frac{z}{z^*}$ at any level of significance. This inference highlights the role of the pairwise interchange intensification in driving the best solution value toward the neighborhood of $z^*$ with a limited number of ants and generations, a non-achievable task for AS with $m = 1000$ and $G = 1000$.

Figure 1 displays the 95 % confidence intervals of the mean ratio $\frac{t}{t^*}$, as a function of $G$, $m$, and $n$, where the ratio adopts a base 10 logarithmic scale, and both $t$ and $t^*$ are expressed in the same time unit (s). The runtime of AS is less than $t^*$ for reasonable numbers of ants $m$ and generations $G$, but can be much larger than $t^*$, for very large $m$ and $G$. In fact, the runtime of AS increases as a function of $n$, $G$, and $m$ with the positive coefficient correlations being significant at all levels. Therefore, any ant colony-based heuristic has to use small numbers of ants and of generations to be competitive with the exact method of [22] in terms of runtime. ASD abides to this runtime constraint. It uses only 5 ants and 5 generations. Consequently, it has a very reduced runtime with a 0.11 point estimate of the ratio $\frac{t_1}{t^*}$ and a limited standard error of 0.05, i.e., with a (0.10, 0.12) estimate of the 95 % confidence interval of the mean $\frac{t_1}{t^*}$. Yet, its solution values are in the neighborhood of $z^*$.

The pairwise exchange intensification makes the ant colony system competitive with the exact approach in terms of runtime and solution quality. It can be argued that the superiority of ASD relative to AS is due to its larger search space: $m'[10(n-2)G'+1]$ ants for ASD versus $m(10G+1)$ ants for AS. Table 3 reports the average $\frac{z}{z^*}$ and $\frac{z_1}{z^*}$ for equal numbers of ants evaluated, and the resulting average run times $t$ and $t_1$ of AS and ASD.

Paired statistical hypothesis testing confirms that the mean of $z_1$ is less than the mean of $z$ at any level of significance for equal total number of ants and that there is statistical evidence that the mean of ASD's runtime $t_1$ is less than the mean of AS' run time $t$ at any level of significance. That is, the pairwise exchange enhances the search of the ants for good paths, and requires less evaluations and sorting than AS with a larger number of ants. However, a statistical $t$ test shows that the mean $\frac{z_1}{z^*}$ is still larger than 1.00 at any level of significance even when $G' = m' = 25$. This suggests that ASD requires

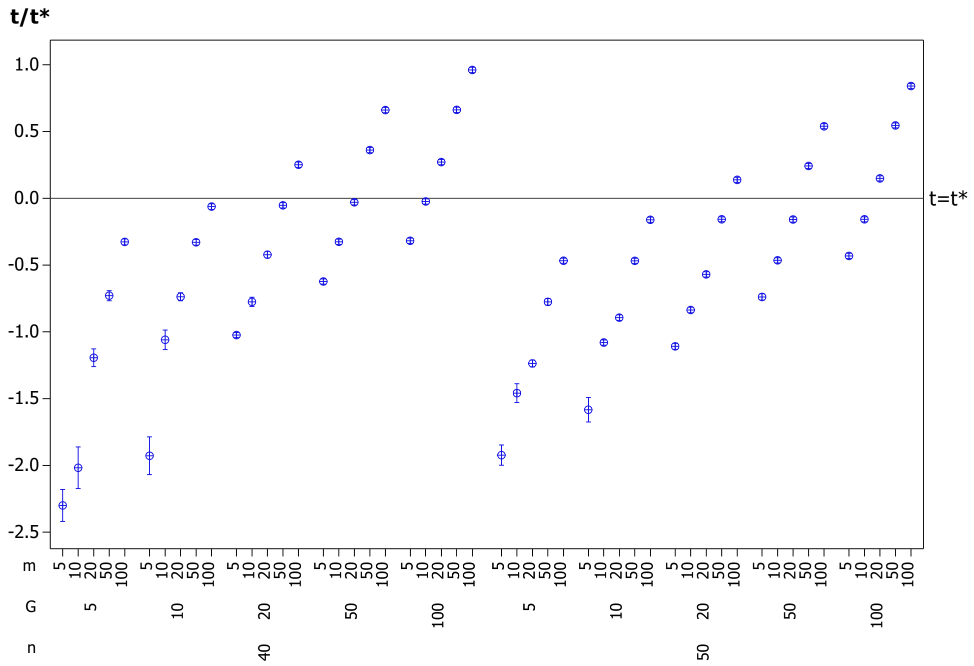


**Fig. 1** 95 % Confidence interval of the mean of $\frac{t}{t^*}$ as a function of $m$, $G$, and $n$

**Table 3** Relative performance of AS and ASD for equal total number of ants

| $n$ | AS |  |  |  | ASD |  |  |  |
|---|---|---|---|---|---|---|---|---|
|  | $G$ | $m$ | $\frac{z}{z^*}$ | $t$ (s) | $G'$ | $m'$ | $\frac{z_1}{z^*}$ | $t_1$ (s) |
| 40 | 5 | 186 | 2.83 | 0.10 | 5 | 5 | 1.05 | 0.01 |
|  |  | 373 | 2.73 | 0.18 |  | 10 | 1.04 | 0.03 |
|  |  | 932 | 2.67 | 0.44 |  | 25 | 1.03 | 0.07 |
|  | 10 | 188 | 2.78 | 0.19 | 10 | 5 | 1.04 | 0.03 |
|  |  | 376 | 2.66 | 0.36 |  | 10 | 1.03 | 0.05 |
|  |  | 941 | 2.62 | 0.88 |  | 25 | 1.03 | 0.13 |
|  | 25 | 189 | 2.70 | 0.48 | 25 | 5 | 1.04 | 0.07 |
|  |  | 379 | 2.60 | 0.90 |  | 10 | 1.03 | 0.13 |
|  |  | 946 | 2.56 | 2.21 |  | 25 | 1.02 | 0.33 |
| 50 | 5 | 235 | 3.12 | 0.20 | 5 | 5 | 1.04 | 0.02 |
|  |  | 471 | 3.04 | 0.37 |  | 10 | 1.03 | 0.05 |
|  |  | 1177 | 3.01 | 0.90 |  | 25 | 1.02 | 0.12 |
|  | 10 | 238 | 3.06 | 0.39 | 10 | 5 | 1.03 | 0.05 |
|  |  | 475 | 2.99 | 0.74 |  | 10 | 1.02 | 0.10 |
|  |  | 1188 | 2.96 | 1.80 |  | 25 | 1.02 | 0.24 |
|  | 25 | 239 | 3.00 | 0.99 | 25 | 5 | 1.02 | 0.12 |
|  |  | 478 | 2.93 | 1.85 |  | 10 | 1.02 | 0.24 |
|  |  | 1195 | 2.89 | 4.58 |  | 25 | 1.02 | 0.59 |

a more aggressive intensification mechanism. This is most likely due to the myopic nature of the move decisions of the ants in presence of a high level of competition of jobs for the same time slots on the machine. This is particularly obvious when a large portion of the jobs are tardy or early. An analysis of variance of $\frac{z_1}{z^*}$ indicates that the tardiness factor $\tau$ is the only significant factor on $\frac{z_1}{z^*}$ at any significance level and that $n$ is significant at the 7.3 % level. This can further be observed in Fig. 2, which displays the 95 % confidence intervals of the mean $\frac{z_1}{z^*}$ as a function of $\rho$, $\tau$, and $n$. When $\tau$ is large (resp. small), the proportion of tardy (resp. early) jobs is very large, making the pairwise exchange effective in finding better solutions. For the middle values of $\tau$, the local pairwise interchange is myopic and can not lead to global optima.

### *5.1.2 Role of initialization of the pheromone trail*

Next, consider AS1 and ASD1, which are variations of AS and ASD with additional daemon actions undertaken on the initial colony. That is, the ants of this colony are subject to intensification. This is equivalent to implementing the recommendation of [24] of initializing the pheromone trail. For $G = m = 10$, paired comparisons of the mean ratios $\frac{z(H)}{z^*}$, $H =$ AS1, ASD, ASD1, suggest that there is no statistical evidence that any pair of mean ratios are different at a 5 % level of significance. However, there is statistical proof that any of the three mean ratios is different than the mean ratio of AS at the 2 % level. This is clearly evidenced by Fig. 3, which displays the 95 % confidence intervals of the mean ratios $\frac{z(H)}{z^*}$, $H =$ AS, AS1, ASD, ASD1. Differently stated, enhancing the quality of ants (be them from colony $g = 0$, $g = 1, \ldots, G$, or $g = 0, \ldots, G$) helps the ants identify good paths that lead to near-global minima and overcome their myopic decisions caused by the sub-optimality of their local moves. Because the mean runtime of ASD is smaller than the mean runtime of ASD1 at

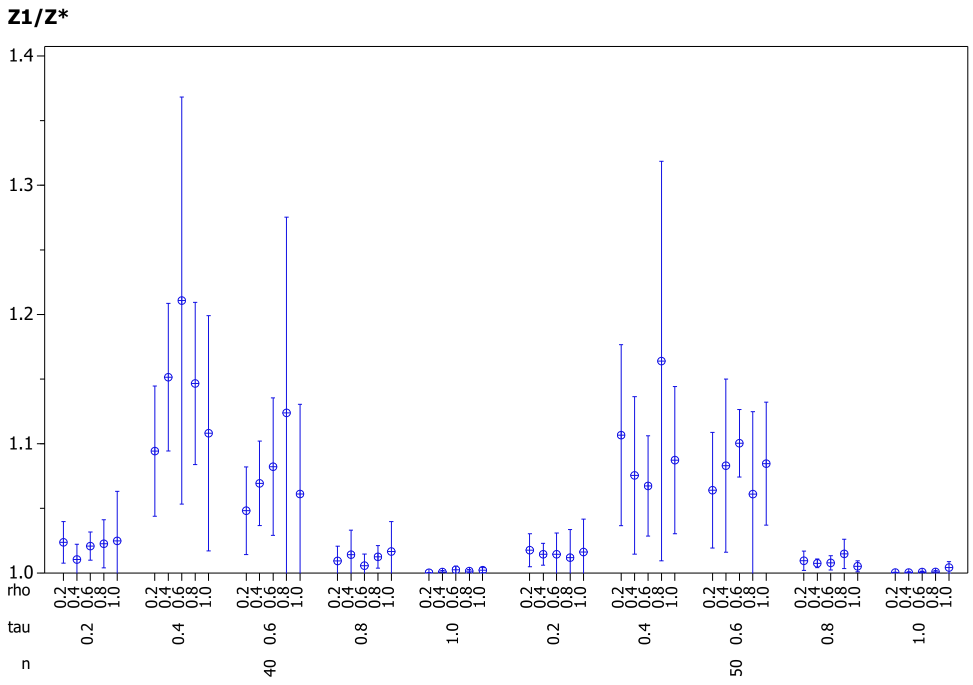


**Fig. 2** 95 % Confidence interval of the mean of $\frac{z_1}{z^*}$ as a function of $\tau$, $\rho$, and $n$

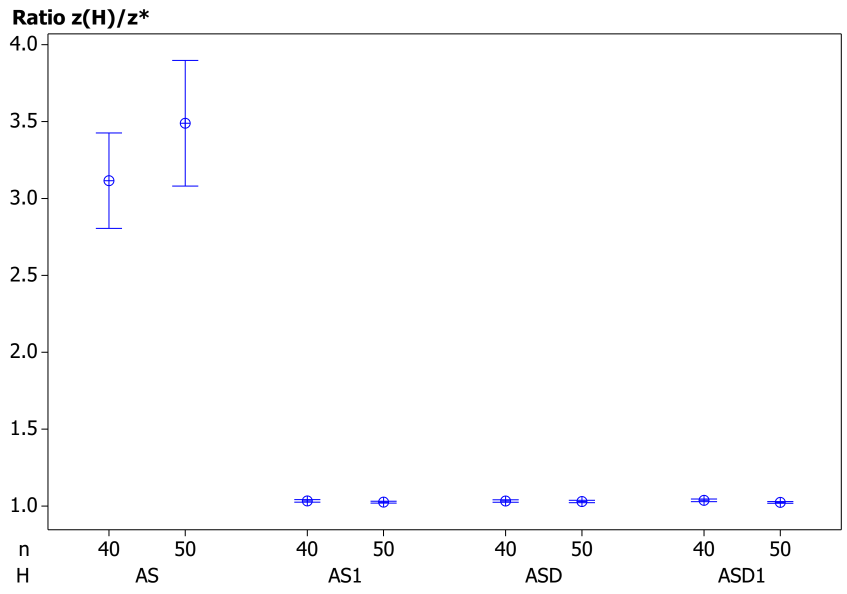


**Fig. 3** 95 % Confidence interval of the mean of $\frac{z(H)}{z^*}$, $H$ = AS, AS1, ASD, and ASD1

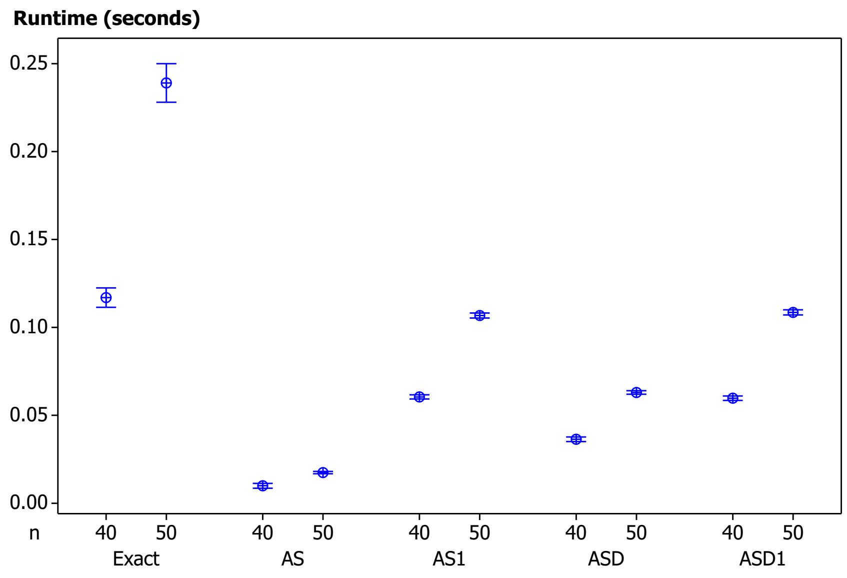


**Fig. 4** 95 % Confidence interval of the mean run times

any level of significance, it seems reasonable to limit the intensification to colonies $g$, $g = 1, \ldots, G$. Figure 4 compares the 95 % confidence intervals of the mean runtime of $H$, $H = AS, AS1, ASD, ASD1$, to the mean runtime of the dynamic programming-based approach of [22].

### 5.1.3 *Importance of ACO as the diversification mechanism*

Let ASS and ASV denote AS with SA and VNS daemon actions, respectively. The following compares a multiple restart SA (MSA) and a hybrid GA-SA (GASA) to ASS, and a multiple restart VNS (MVNS) to ASV. The heuristics use equal run times, set equal to one eighth the median of $t^*$ for each problem size. Figure 5 displays the 95 % confidence intervals for the mean $\frac{z(H)}{z^*}$, $H$ = ASD, ASS, ASV, MSA, MVNS, and GASA.

Figure 5 suggests that ASV and MVNS outperform the other heuristics, confirming the importance of the VNS daemon actions. In addition, it infers that ACO is more effective than GA as the diversification mechanism when the daemon action is an SA. A paired $t$ test shows that the mean of $\frac{z(\text{ASS})}{z^*}$ is less than the mean of $\frac{z(\text{GASA})}{z^*}$ at any level of significance with the mean difference equaling −0.01008. Finally, it shows that the diversification using ACO is more successful than a random multiple restart. A paired $t$ test shows that the mean of $\frac{z(\text{ASS})}{z^*}$ is less than the mean of $\frac{z(\text{MSA})}{z^*}$ at any level of significance with the mean difference equaling −0.02173. Similarly, a paired $t$ test shows that the mean of $\frac{z(\text{ASV})}{z^*}$ is

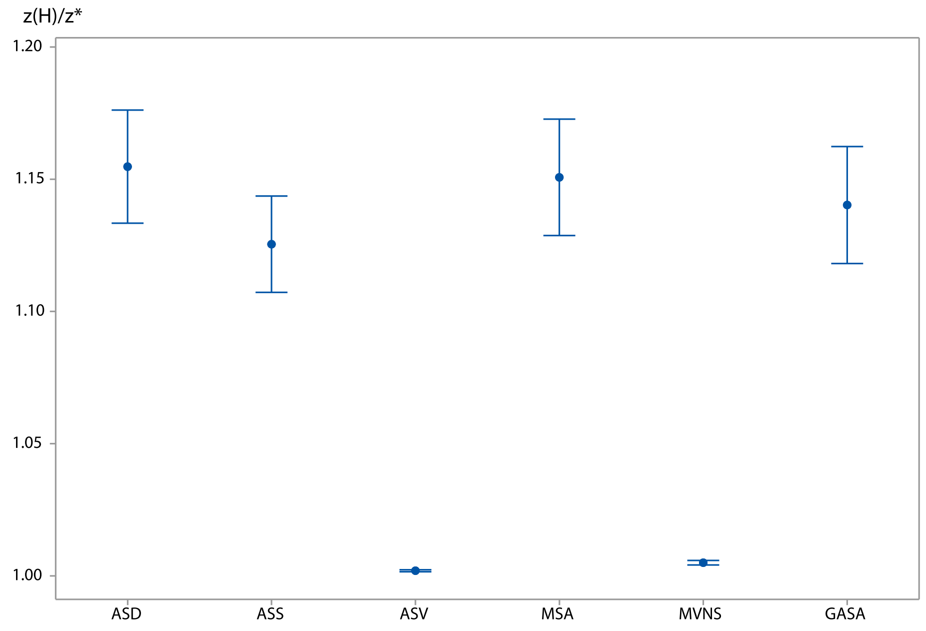


**Fig. 5** 95 % Confidence interval of the mean $\frac{z(H)}{z*}$, $H$ = ASD, ASS, ASV, MSA, MVNS, and GASA

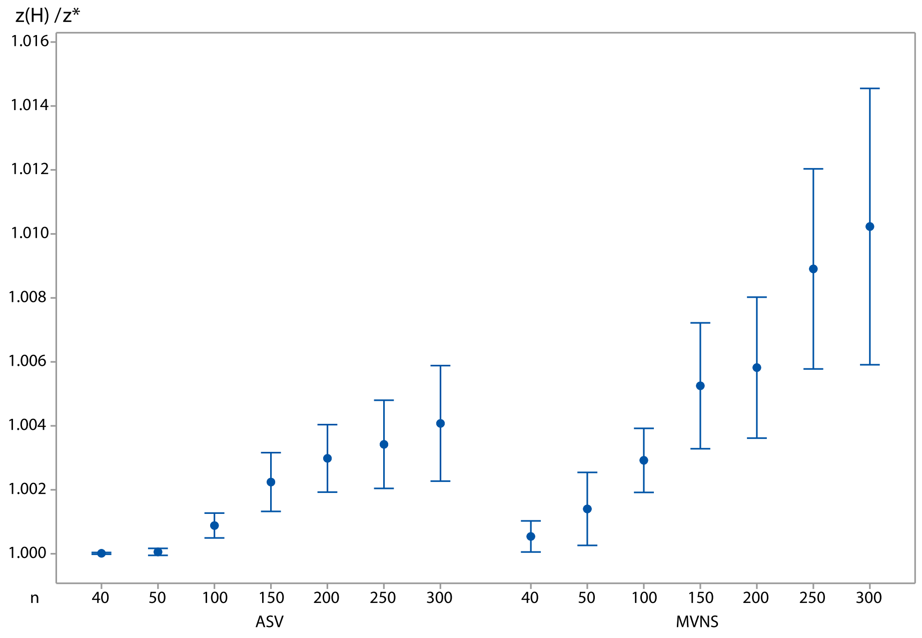


**Fig. 6** 95 % Confidence interval of the mean $\frac{z(H)}{z*}$, $H$ = ASV and MVNS, as a function of problem size

less than the mean of $\frac{z(\text{MVNS})}{z*}$ at any level of significance with the mean difference equaling −0.02719. Figure 6 clarifies this superiority as a function of the problem size. In fact, the effect of using ACO as the diversification mechanism of VNS is amplified as the problem size increases.

## 5.2 ACO's performance

This subsection investigates the relative and absolute performance of ASD, ASS, and ASV. Based on the results of Sect. 5.1.3, paired comparisons of $\frac{z(H)}{z*}$ for $H$ = ASD, ASS, and ASV, show that there is sufficient statistical proof to claim that the mean $\frac{z(ASV)}{z*}$ is less than both mean ratios for ASS and ASD at any level of significance, with the mean difference being respectively −0.13840 and −0.11231. The mean $\frac{z(ASV)}{z*}$ is 1.0041 with a standard error of 0.0003 with the three quartiles being 1.0001, 1.0012, and 1.0047 while the maximum is 1.0904.

The same inference is valid when ASD, ASS and ASV are run with $m = G = 10$, as evidenced by Fig. 7 which compares the 95 % confidence intervals of the mean $\frac{z(H)}{z*}$ for $H$ = ASD, ASS, and ASV. However, the statistics of $\frac{z(ASV)}{z*}$ are enhanced. The mean $\frac{z(ASV)}{z*}$ becomes 1.0020 with a standard error of 0.0001 with the three quartiles becoming 1.0000, 1.0005, and 1.0023 and the maximum 1.0513. This maximum (which is an outlier) is registered for $n = 300$,

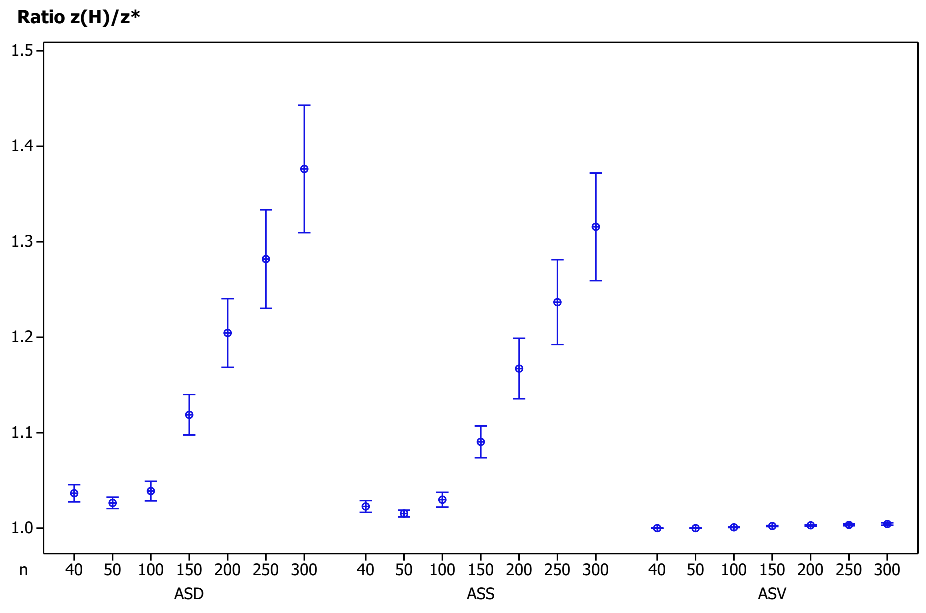


**Fig. 7** 95 % Confidence interval of the mean $z(ASD)/z*$, $z(ASS)/z*$ and $z(ASV)/z*$ as a function of problem size $n$

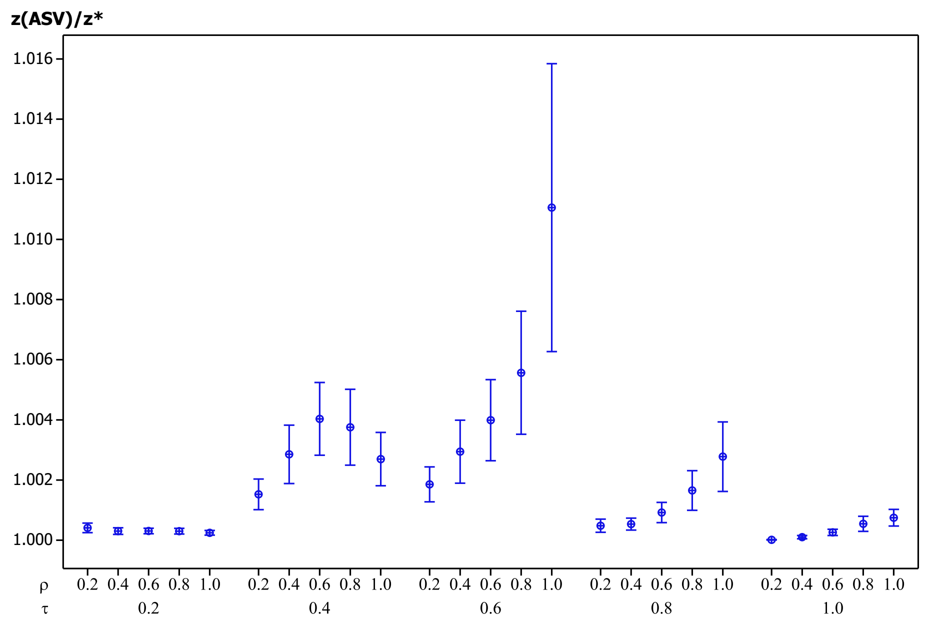


**Fig. 8** 95 % Confidence interval of the mean $z(ASD)/z*$, $z(ASS)/z*$ and $z(ASV)/z*$ as a function of the tardiness factor $\tau$ and relative range of due dates $\rho$

$\tau = 0.6$ and $\rho = 1.0$. The mean for this category is 1.0039. It is the highest among all classes of $n, \tau, \rho$ instances. The values of the first and second quartiles are due to ASV matching the optimum in 122 and 118 out of 125 instances for $n = 40$ and 50. However, the number of times $z(ASV) = z^*$ decreases as $n$ increases. This is expected since the algorithm is using a reduced number of ants and of generations.

The effect of $\tau$ and $\rho$ is clearly illustrated by the 95 % confidence intervals of the mean $z(ASV)/z^*$ displayed in Fig. 8. ASV performs best for low or high $\tau$ and worst for $\tau = 0.6$. Its mean ratio increases as $\rho$ and $n$ increase. There is statistical proof of the prevalence of these relationships at any level of significance as supported by analysis of variance tests.

Paired comparisons of the runtime of $H$ = ASD, ASS, and ASV with $m = G = 10$ to the runtime of the exact method show that there is sufficient statistical proof to claim that the mean runtime of the exact method is larger than the mean runtime of $H$ = ASD, ASS and ASV at any level of significance. This is further evidenced by the comparison of the 95 % confidence intervals of the mean run times displayed in Fig. 9. For the largest instances with $n = 300$, the median runtime of the exact approach, ASD, ASS and ASV are 288.70, 15.36, 29.39, and 92.16, respectively. That is, ASV is on average three times faster than the exact method with a 0.2 % average deviation of its solution values from the optimal ones. Thus, ASV's near-global minima can be used as upper bounds for the dynamic programming-based

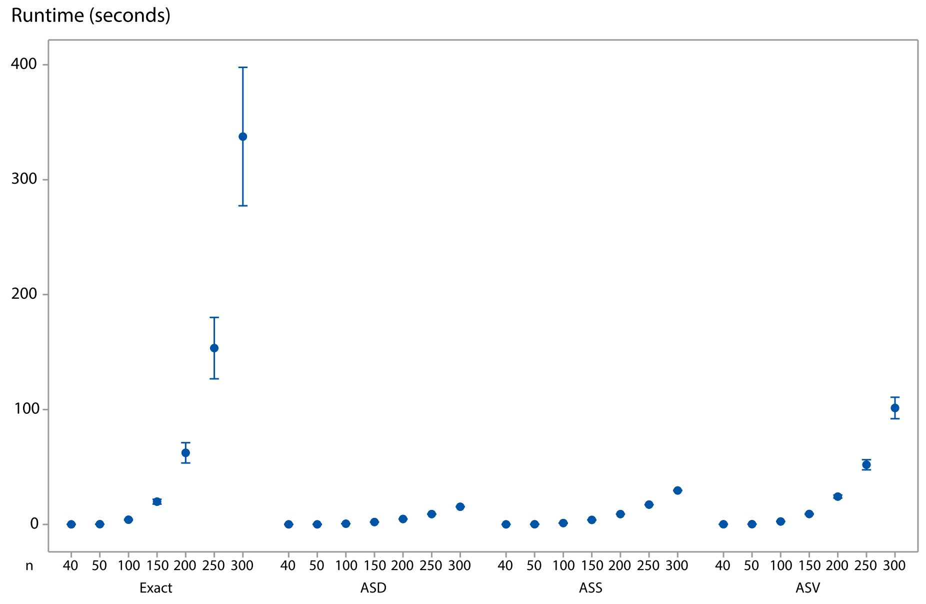


**Fig. 9** 95 % Confidence interval of the mean run times as a function of problem size

approach of [22]. The most important competitive advantage of ASV in comparison to the exact method of [22] is its adaptability. ASV can be applied to any machine environment or to scheduling constraints such as precedence, set up, deteriorating jobs, inserted idleness, or objective criterion. This application would involve changing the assessment mechanism of the ants. Evidently, this adaptability is not straightforward for the exact method of [22].

## 6 Conclusion

This paper addresses the minimum weighted earliness tardiness single machine scheduling problem with distinct deterministic known due dates and no idle time. It approximately solves the problem via an ant colony optimization variable neighborhood search system. It shows that dotting an ant system with daemon actions that intensify the search around the ants of each colony enhances the efficacy of the ant system. In addition, it improves the efficiency of the ant system which converges to near-global optima with a reduced number of ants and generations (thus a smaller runtime). It further shows that daemon actions that consist in a variable neighborhood search yield (near-)global optima in most instances. Finally, it highlights the utility of diversification via ant colony optimization. The proposed approach can be easily extended to more complex scheduling problems with different job environments and constraints, and to other combinatorial problems where the short-term optimal decisions of the ants regarding their immediate moves from one state to the next are not necessarily globally optimal.

## References

1. Behnamian, J., Fatemi Ghomi, S. M. T., & Zandieh, M. (2010). Development of a hybrid meta heuristic to minimise earliness and tardiness in a hybrid flow shop with sequence-dependent setup times. *International Journal of Production Research*, *48*(5), 1415–1438.
2. Blum, C. (2005). Beam-ACO-hybridizing ant colony optimization with beam search: An application to open shop scheduling. *Computers & Operations Research*, *32*, 1565–1591.
3. Bulbul, K., & Kaminsky, P. (2013). A linear programming-based method for job shop scheduling. *Journal of Scheduling*, *16*(2), 161–183.
4. Gagné, C., Price, W. L., & Gravel, M. (2002). Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. *Journal of the Operational Research Society*, *53*, 895–906.
5. Holthaus, O., & Rajendran, C. (2005). A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs. *Journal of the Operational Research Society*, *56*, 947–953.
6. Huang, K. L., & Liao, C. J. (2008). Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers & Operations Research*, *35*, 1030–1046.
7. Kim, Y., Lim, H. G., & Park, M. W. (1996). Search heuristics for a flow shop scheduling problem in a printed circuit board assembly process. *European Journal of Operational Research*, *91*(1), 124–143.
8. Li, H., & Zhang, H. (2013). Ant colony optimization-based multimode scheduling under renewable and nonrenewable resource constraints. *Automation in Construction*, *35*, 431–438.
9. Liaw, C. F. (1999). A branch and bound algorithm for the single machine earliness and tardiness scheduling problem. *Computers & Operations Research*, *26*, 679–693.
10. Lo, S. T., Chen, R. M., Huang, Y. M., & Wu, C. L. (2008). Multiprocessor system scheduling with precedence and resource constraints using an enhanced ant colony system. *Expert Systems with Applications*, *34*(3), 2071–2081.
11. Marimuthu, S., Ponnambalam, S. G., & Jawahar, N. (2009). Threshold accepting and ant-colony optimization algorithms for

scheduling m-machine flow shops with lot streaming. *Journal of Materials Processing Technology*, *209*, 1026–1041.

12. M'Hallah, R. (2007). Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. *Computers & Operations Research*, *34*(10), 3126–3142.
13. M'Hallah, R. (2014). An iterated local search variable neighborhood descent hybrid heuristic for the total earliness tardiness permutation flow shop. *International Journal of Production Research*, *52*(13), 3802–3819.
14. M'Hallah, R., & Alhajraf, A. (2008). Ant colony optimization for the single machine total earliness tardiness scheduling problem. *Lecture Notes in Computer Science*, *5027*, 397–407.
15. M'Hallah, R., & Alkhabbaz, A. (2013). Scheduling of nurses: A case study of a Kuwaiti health care unit. *Operational Research for Health Care*, *2*(1–2), 1–19.
16. M'Hallah, R., & Al-Roomi, A. (2014). The planning and scheduling of operating rooms: A simulation based approach. *Computers & Industrial Engineering*, *78*, 235–248.
17. Muller-Hannemann, M., & Sonnikow, A. (2009). Non-approximability of just-in-time scheduling. *Journal of Scheduling*, *12*(5), 555–562.
18. Parthasarathy, S., & Rajendran, C. (1997). A simulated annealing heuristic for scheduling to minimize mean weighted tardiness in a flow shop with sequence-dependent setup times of jobs–A case study. *Production Planning and Control*, *8*(5), 475–483.
19. Rocha de Paula, M., Ravetti, M. G., Mateus, G. R., & Pardalos, P. M. (2007). Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search. *IMA Journal of Management Mathematics*, *18*(2), 101–115.
20. Schaller, J., & Valente, J. (2013). A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness. *International Journal of Production Research*, *51*(3), 772–779.
21. Tanaka, S., & Fujikuma, S. (2012). A dynamic-programming-based exact algorithm for general single-machine scheduling with machine idle time. *Journal of Scheduling*, *15*(3), 347–361.
22. Tanaka, S., Fujikuma, S., & Araki, M. (2009). An exact algorithm for single-machine scheduling without machine idle time. *Journal of Scheduling*, *12*, 575–593.
23. Tasgetiren, M. F., Liang, Y., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research*, *177*, 1930–1947.
24. Tavares Neto, R. F., & Godinho, Filho M. (2013). Literature review regarding ant colony optimization applied to scheduling problems: Guidelines for implementation and directions for future research. *Engineering Applications of Artificial Intelligence*, *26*, 150–161.
25. Wan, L., & Yuan, J. (2013). Single-machine scheduling to minimize the total earliness and tardiness is strongly NP-hard. *Operations Research Letters*, *41*, 363–365.
26. Zegordi, S. H., Itoh, K., & Enkawa, T. (1995). A knowledgeable simulated annealing scheme for the early/tardy flow shop scheduling problem. *International Journal of Production Research*, *33*(5), 1449–1466.