Computers & Operations Research 44 (2014) 115–136

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A multi-agent system for the weighted earliness tardiness parallel machine problem

S. Polyakovskiy [a], R. M'Hallah [b,*]

[a] School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia
[b] Department of Statistics and Operations Research, College of Science, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait

ARTICLE INFO

Available online 28 October 2013

Keywords:
Parallel machine scheduling
Multi-agent systems
Artificial intelligence
Earliness
Tardiness
Local search

ABSTRACT

This paper studies the weighted earliness tardiness parallel machine problem where jobs have different processing times and distinct due dates. This NP hard problem arises in most just-in-time production environments. It is herein modeled as a mixed integer program, and solved using MAS^H, a deterministic heuristic based on multi-agent systems. MAS^H has three types of agents: I, G, and M. The I-agents are free jobs that need to be scheduled, whereas the G-agents are groups of jobs already assigned to machines. The M-agent acts as the system's manager of the independent intelligent I- and G-agents, which are driven by their own goals, fitness assessments, and context-dependent decision rules. The I- and G-agents employ exact and approximate approaches as part of their decisional process while the M-agent uses local search mechanisms to improve their (partial) solutions. The design of MAS^H is innovative in the way its intelligent agents determine bottleneck clusters and resolve conflicts for time slots. The numerical results provide computational evidence of the efficiency of MAS^H, whose performance on benchmark instances from the literature is superior to that of existing approaches. The success of MAS^H and its modularity make it a viable alternative to more complex manufacturing problems. Most importantly, they demonstrate the benefits of the hybridization of artificial intelligence and operations research.

1. Introduction

Minimizing total earliness and tardiness penalties gained importance with the emergence of Just-In-Time. Companies adhering to this philosophy strive to complete jobs exactly on their due dates. A job that completes before its due date incurs an earliness penalty due to the holding cost whereas a job that completes after its due date generates a tardiness cost due to late charges, express delivery, and lost sales. In such environments, classical measures of performance (e.g., makespan, mean tardiness, mean lateness) are not suitable.

This paper studies the deterministic non-identical parallel machine weighted earliness tardiness problem, $R|d_j|\sum\alpha_j E_j+\beta_j T_j$, denoted hereafter PWET. Every job of a set of n independent jobs, available at time zero, is to be scheduled, without preemption, on one of m non-identical parallel machines. A job j, $j=1,\ldots,n$, is characterized by a known distinct due date d_j, a processing time p_{ij} on machine i, $i=1,\ldots,m$, a per unit cost of earliness α_j, and a per unit cost of tardiness β_j. A machine can process only one job at a time, and can be idle. The objective is to find a schedule that minimizes the sum of weighted earliness and tardiness (WET) of the n jobs.

Herein, PWET is approximately solved using MAS^H, a heuristic based on a multi-agent system (MAS). The superscript H in MAS^H distinguishes between the general MAS framework and the proposed heuristic. Four factors motivate this choice of approach. First, MAS is in general best suited for modular applications [25], and PWET is modular. Its modules are the jobs and machines. Second, MAS is particularly useful when the problem has no alternative optimization technique or when existing ones are highly expensive, not efficient, fast, or easily applicable [16]. This is the case of PWET where large-sized instances can't be solved exactly, and existing approximate approaches for variants of the problem are not fast and efficient. Third, MAS^H adopts a decentralized decisional process in the sense that some decisions are delegated to low-level agents in lieu of emanating from the higher-level system's manager [6]. Differently stated, there is no single intelligent entity which provides a monolithic solution to the scheduling problem. This characteristic allows a higher level of autonomy for lower-level agents in their decision making process. It is applied to enhance the system's performance. On the other hand, existing approximate optimization techniques that tackled variations of PWET are centralized in the sense that the optimization technique decides the starting time of every job and its

* Corresponding author. Tel.: +965 669 14150; fax: +965 248 37332.
E-mail addresses: maxles@yandex.ru (S. Polyakovskiy), rym.mhallah@ku.edu.kw, rymmha@yahoo.com (R. M'Hallah).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.10.013

machine assignment. Fourth and last, PWET involves tradeoffs and bargaining among jobs and competition for resources. These activities cannot be correctly described by a centralized approach, while they can be captured by MAS^H. (cf. [4] for a comparison of agent-based versus classical optimization techniques.)

The efficient adaptation of a search heuristic to any problem requires a thorough preliminary study of the problem so that its specificities are exploited during the design stage of the heuristic. Like genetic algorithms and ant colonies, MASs constitute a search framework which might not yield high performance heuristics if these latter are not dotted with the appropriate control parameters, decoding mechanisms, etc. Therefore, the successful implementation of the proposed MAS^H depends upon the definition of the agents, the elaboration of their competition and cooperation mechanisms, and the construction of their decisional processes.

MAS^H represents a new design of multi-agent based heuristics. It employs an M-agent that acts as the system's manager of two types of low-level rational agents: free jobs that need to be scheduled, and groups of jobs already assigned to machines. MAS^H bases the agents' decisional processes on "if–then" decision rules that are supported by exact and approximate optimization techniques. It federates these decisions via a competition and negotiation model that is partially inspired from [26]. Thus, its solution construction concept is different from existing approaches in the literature [1,8,21]; in particular, in the way its agents determine bottleneck clusters and resolve conflicts among jobs competing for the same time slot. Moreover, it has an architecture that speeds the required computations. Most importantly, MAS^H demonstrates the potential of the hybridization of artificial intelligence and operations research. Embedding exact optimization techniques into the agents' decisional processes and subjecting a (partial) solution to a local search are, respectively, low and high-level hybridizations whose success highlights the "complementary characteristics of MAS and classical heuristics" [4]. In addition to its innovative aspects and scientific merit, MAS^H can solve PWET when it arises as a subproblem in complex industrial settings; for example, for scheduling jobs on parallel cutters for on-line furniture manufacturing [27].

This paper details the proposed MAS^H and tests its performance on two sets of benchmark instances. The results demonstrate the competitiveness of MAS^H in terms of better local optima and reduced run times; thus providing computational proof of the successful implementation of MAS^H. Section 2 surveys the literature on PWET. Section 3 models PWET as a mixed integer program. Section 4 details MAS^H. Section 5 presents the results. Finally, Section 6 is a summary.

2. Literature review

Most of the literature on parallel machine earliness tardiness problems determines a common due date and subsequently schedules the jobs [14,20]. Alidaee and Panwalkar [2] and Kubiak et al. [19] design an $O(n^3)$ algorithm for $R|d_j=d|\sum E_j+T_j$. De et al. [9] show that $P|d_j=d_{unres}|\sum w_j(E_j+T_j)$ is NP-hard for $m=2$ and strongly NP-hard for $m>2$, where the non-restrictive common due date d_{unres} is large enough not to constrain the scheduling process. They apply a pseudo polynomial dynamic program to problems with 30 jobs but whose characteristics are small integers. Federgruen and Mosheiov [11] derive a lower bound and a heuristic for $P|d_j=d_{unres}|\sum w_j(E_j+T_j)$. Chen and Powell [7] model $P|d_j=d_{unres}|\sum w_j(E_j+T_j)$ as an integer program, and decompose it into a set partitioning formulation with side constraints. Biskup and Cheng [5] show that $P|d_j=d,p_j=p|\sum \alpha E_j+\beta T_j+\delta d$ is polynomially solvable, and present a heuristic for the case with distinct processing times. Sun and Wang [30] show that $P|d_j=d,w_j=p_j|\sum w_j(E_j+T_j)$ is NP-hard, and solve it via dynamic programming. Solis and Sourd [29] apply a local search to $P|d_j=d|\sum \alpha_j E_j+\beta_j T_j$. Drobouchevitch and Sidney [10] consider $Q|p_j=p,d_j=d|f(E_j,T_j,d)$ where the decision variable d minimizes a function of earliness, tardiness and due date penalties. Mosheiov and Sarig [24] develop a constant-time algorithm for the case with two uniform machines. Gerstl and Mosheiov [13] study the uniform parallel machine problem where jobs belong to one of k classes and the jobs of a class share a common due-date. They minimize the maximum earliness/tardiness cost and the sum over all k classes of the maximum earliness/tardiness cost of the jobs of each class. Bank and Werner [3] develop a heuristic for the unrelated parallel machine problem with a common due date, release dates, and linear earliness and tardiness penalties.

In contrast to the previous research, Mason et al. [23], Kedad-Sidhoum et al. [18], and M'Hallah and Al-Khamis [22] consider the due dates distinct and known. They focus on the identical machine case. The formers study the unweighed case (i.e. $\alpha_j=\beta_j=1$), whereas the latter considers the general weighed case with the earliness and tardiness penalties not necessarily equal. Mason et al. [23] apply a moving block heuristic and present a mixed integer program based lower bound. Kedad-Sidhoum et al. [18] propose two lower bounds and assess their tightness. M'Hallah and Alkhamis [22] model their problem as a mixed-integer program which exactly solves small-sized instances. For large instances, they design hybrid heuristics which make steepest descent and simulated annealing collaborate with genetic algorithms via different levels and types of hybridization. Variants of PWET are available. For example, Toksari and Güne [31] consider the effects of learning and deterioration. Polyakovskiy and M'Hallah [27] tackle a real life problem that occurs in the furniture industry. Orders are received online. The furniture plant has to decide how to assemble the components of the orders into groups and how to cut them from two-dimensional boards. The objectives of the plant are to maximize the utilization of the boards and minimize the weighted earliness tardiness of the components of the orders as to avoid their temporary storage and blocking the next stations. Thus, PWET was a subproblem to the multiple objective online scheduling problem. It was tackled via a simple heuristic.

This paper addresses $R|d_j|\sum \alpha_j E_j+\beta_j T_j$. It proposes a multi-agent deterministic approach which explores the specificities of the problem to search for a near optimum in lieu of relying on mere enumeration. This heuristic defines its agents in a unique way that distinguishes it from existing MAS implementations for scheduling problems. It is faster and obtains better local minima than the hybrid search H4 of [22], and the search method, denoted hereafter KSS, of [18] for the identical machine case.

3. A mathematical model

There are few mathematical programming models for PWET. Herein, the precedence-based mixed integer linear programming formulation is extended to the non-identical machine case. It uses six types of decision variables. Let x_{ikj}, $i=1,\ldots,m$, $k=1,\ldots,n$, $j=1,\ldots,n$, $k\neq j$, equal 1 if job k *immediately precedes* job j and *both k and j are assigned to the same machine i*, and 0 otherwise. In addition, let ε_{ij}, $i=1,\ldots,m$, $j=1,\ldots,n$, equal 1, if job j is the first job on machine i, and 0 otherwise. Let S_j, $j=1,\ldots,n$, and C_j, $j=1,\ldots,n$, denote the starting time and completion time of job j, respectively. When j is processed on machine $i\in\{1,\ldots,m\}$, $C_j=S_j+p_{ij}$. Finally, let $E_j=\max\{0,d_j-C_j\}$ and $T_j=\max\{0,C_j-d_j\}$ designate the earliness and tardiness of job j, $j=1,\ldots,n$. The first two decision variables are binary whereas

the last four are integer. Using the above decision variables, PWET is modeled as follows:

$$Z = \min \quad \sum_{j=1}^{n} \alpha_j E_j + \beta_j T_j \tag{1}$$

$$C_j - d_j = T_j - E_j, \quad j = 1, \ldots, n \tag{2}$$

$$C_j = S_j + \sum_{i=1}^{m} p_{ij} \left(\varepsilon_{ij} + \sum_{k=1}^{n} x_{ikj} \right), \quad j = 1, \ldots, n \tag{3}$$

$$S_j - S_k \geq \sum_{i=1}^{m} p_{ik} x_{ikj} - M \left(1 - \sum_{i=1}^{m} x_{ikj} \right), \quad k = 1, \ldots, n, \quad j = 1, \ldots, n, \ k \neq j \tag{4}$$

$$\sum_{i=1}^{m} (x_{ikj} + x_{ijk}) \leq 1, \quad k = 1, \ldots, n, \quad j = k+1, \ldots, n \tag{5}$$

$$\sum_{i=1}^{m} \left(\varepsilon_{ij} + \sum_{\substack{k=1 \\ k \neq j}}^{n} x_{ikj} \right) = 1, \quad j = 1, \ldots, n \tag{6}$$

$$\sum_{j=1}^{n} \varepsilon_{ij} = 1, \quad i = 1, \ldots, m \tag{7}$$

$$x_{ikj} \in \{0, 1\}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n, \quad k = 1, \ldots, n, \ k \neq j \tag{8}$$

$$\varepsilon_{ij} \in \{0, 1\}, \ i = 1, \ldots, m, \quad j = 1, \ldots, n \tag{9}$$

$$S_j, \ C_j, \ E_j, \ T_j \text{ integer} \quad j = 1, \ldots, n, \tag{10}$$

where M is a large positive number such that $M \to \infty$. Herein, M is set to $\max_{j=1,\ldots,n}\{d_j\} + \max_{i=1,\ldots,m}\{\sum_{j=1}^{n} p_{ij}\}$.

Eq. (1) defines the objective function value Z as the minimal sum of weighted earliness and tardiness of all jobs. Eq. (2) computes both the tardiness and earliness of job j by setting their difference to the difference between the j's completion time and due date. If j is tardy, then $C_j - d_j > 0$; thus, $T_j > 0$ and $E_j = 0$. On the other hand, if j is early, then $C_j - d_j < 0$; thus, $T_j = 0$ and $E_j > 0$. When j is on time, its $C_j - d_j = 0$; thus, $T_j = 0$ and $E_j = 0$. There are n of these equations. Eq. (3) sets the completion time of j to the sum of its starting and processing time. There are n of these equations. Eq. (4) relates the starting times of two successive jobs on the same machine. If both k and j are scheduled on the same machine i, and k is the immediate predecessor of j, then the starting time of j is greater than or equal to the sum of the starting time of k and its processing time on machine i. The starting time of j and completion time of k coincide when there is no idle time between k and j; otherwise, $S_j > C_k$. The constraint does not establish any relationship between S_k and S_j when k does not immediately precede j, or j and k are not scheduled on the same machine. There are $n(n-1)$ of these constraints. Eq. (5) reinforces the precedence relations between any pair of jobs k and j. Job k immediately precedes j or j immediately precedes k (if both are scheduled on the same machine i), or neither relation holds. These $n(n-1)/2$ constraints reduce the symmetry of the search space, but have no functional utility. Eq. (6) forces a job j with no immediate predecessor to be the first job on one of the m machines. There are n constraints of this type. Eq. (7) imposes that each machine i has a "first" job. There are m constraints of this type. Eqs. (8) and (9) declare x_{ikj} and ε_{ij} binary. Finally, Eq. (10) states that the starting time, completion time, earliness and tardiness of job j are integer. Eq. (10) can be replaced by non-negativity constraints when the processing times, due dates, and weights are all integers. In that case, there exists at least one optimal solution with integer values for all the four variables. PWET is NP hard. It is an extension of the NP-hard identical parallel machine case [22]. Thus, solving the above model using off-the-shelf solvers for moderately sized instances is impossible.

The large number of binary decision variables and of constraints make PWET well adapted to heuristic methods, in general, and to approaches based on the MAS paradigm [4,28], in particular. An MAS partitions a problem into smaller, simpler components that constitute the agents. The agents undertake activities that define the solution's process' dynamics, evolution, and final structure. By striving to achieve their respective goals, the agents improve the system's objective and influence its result. Their decisions emanate from their active interaction and negotiation.

4. A multi-agent system

An agent is a hardware or software-based self-contained problem-solving entity with the key properties of autonomy, social skills, responsiveness and pro-activeness [17]. Autonomy implies that the agent is capable of carrying out actions independently without any external intervention, be it from artificial agents or from human decision makers. Social skills infer that the agent can interact with other agents. Responsiveness suggests that the agent is capable of assessing its environment and adapting to its changes. Finally, pro-activeness indicates that the agent is dotted with a goal-directed initiating behavior. These four properties differentiate an "intelligent" agent from an object by endowing it with a rational behavior and a state.

An agent is characterized by its own parameters, specific decision rules and actions, individual goal and fitness function. It employs the most appropriate paradigm for solving its own problem. At any decision point, it applies a strategy to assess the potential of available actions and chooses one that "optimizes" its reward. When the strategy is greedy, the reward may be a local (i.e., not necessarily a global) optimum. While the specific behavior of an individual agent can be described, the behavior of the entire system cannot be modeled because of all the interactions among agents over time.

This section describes how the general MAS framework is adapted to PWET and motivates the adopted implementation of MAS^{H}. Section 4.1 presents the components of MAS^{H}: the agents, their characteristics, and their decision mechanisms. Section 4.2 describes the architecture of MAS^{H}. Section 4.3 details the construction of a feasible schedule whereas Section 4.4 explains how it is improved.

4.1. The agents of MAS^{H}

MAS^{H} employs three types of agents: G, I, and M. G-agents are groups of jobs already scheduled on specific machines whereas I-agents are free jobs to be scheduled. The features of G- and I-agents are summarized in Table 1. There is a unique M-agent, which plays the role of a system's manager. Among other tasks, it federates the activities of the G- and I-agents, which compete for the resources (i.e., for the free time slots on the machines).

A G-agent $G[i][j]$ is tagged to a machine $i \in \{1, \ldots, m\}$, and initiated by a job $j \in \{1, \ldots, n\}$. It creates and gradually expands a group of jobs $J_{G[i][j]}$ that it schedules on i. Its role is to use i rationally inducing the smallest WET for i. It competes with other G-agents to attract the "best" I-agents to $J_{G[i][j]}$. Iteratively, it identifies the I-agent whose ideal processing period has the largest overlap with the processing period of the group. The ideal processing period of an I-agent associated to a job j' is $[d_{j'} - p_{ij'}, d_{j'}]$ on i. It results in a zero WET for j'. The processing period of the G-agent $G[i][j]$ is $[S_{G[i][j]}, C_{G[i][j]}]$, where $S_{G[i][j]}$ is the starting time of the first job of $J_{G[i][j]}$, $C_{G[i][j]}$ is the completion time of the last job in $J_{G[i][j]}$, and the jobs in $J_{G[i][j]}$ are sorted in non-decreasing order of the

Table 1
Features of the low-level MAS^{H} agents.

	G-agent (group of jobs)	I-agent (job)
Communication	Interaction with I-agents	Interaction with G-agents
Role	To create and expand a group of jobs	To join a group
Competition	With other G-agents to attract the "best" I-agents	With other I-agents to join the "best" G-agent's group
Goal	To use the machine rationally inducing the least possible weighted earliness–tardiness for the machine	To minimize the increment of weighted earliness and tardiness its scheduling would cause to the current schedule
Fitness function	Length of overlap of the ideal processing period of I-agent with the processing period of the group	WET increment of the schedule when it joins the group

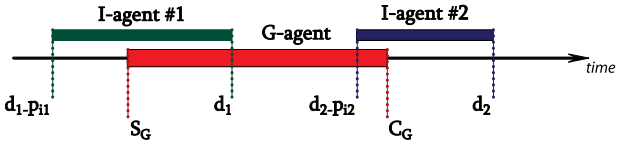

Fig. 1. G-agent makes an attachment offer to I-agent #1.

completion times. The processing period of a G-agent may contain idle time. The I-agents with the *largest* overlap can be viewed as "troublesome" jobs since they are competing for busy time slots. Thus, they are given a higher priority to resolve their conflicts and minimize their WET before the machine becomes loaded. Those jobs with no (respectively with a small) overlap will not increase WET (respectively by much) anyway; so their scheduling can be postponed to a later stage of the heuristic.

In Fig. 1, the G-agent, whose processing interval is $[S_G, C_G]$, is considering attaching I-agent #1 or #2, whose ideal processing times are $[d_1 - p_{i1}, d_1]$ and $[d_2 - p_{i2}, d_2]$, respectively. Because $[d_1 - p_{i1}, d_1]$ has a larger overlap with $[S_G, C_G]$ than $[d_2 - p_{i2}, d_2]$, G-agent prefers attaching I-agent #1.

An I-agent is free if it does not belong to a group, and busy otherwise. A free I-agent f competes with other I-agents to join a G-agent's group. It acts in response to attachment offers made by G-agents. It investigates joining the available groups, and selects the G-agent $G[i][j]$ such that the WET of machine i will be least affected when f joins $G[i][j]$. This is equivalent to finding the smallest increase of WET of the current partial schedule. Hereafter, "job" and "I-agent" are used interchangeably.

4.2. System's architecture

MAS^{H} has a holonic structure. It employs cooperating local agents (i.e., the I- and G-agents) that are managed by a system's managing agent (i.e., the M-agent) [6]. The holonic structure blends the advantages of the hierarchical and heterarchical structures while it avoids their drawbacks. Specifically, the hierarchical architecture involves a command-response structure between high- and low-level entities while the heterarchical one grants more autonomy and decision-making power to low-level entities. While low-level entities may concentrate on a better exploitation of their local neighborhood, high-level entities may explore the entire search space since they benefit from a global view of the system. Their exploration may allow them to adjust the system's behavior more effectively.

In our context, the M-agent is the high-level entity. It ensures the hierarchical aspect. It creates the initial set of G-agents using fixed rules, and runs local search procedures to enhance partial feasible solutions. Furthermore, it coordinates the activities of the G-agents fairly; granting them computational resources to perform their activities. Its coordination of the low-level agents guarantees the convergence to a feasible solution; an important feature of any MAS [17].

On the other hand, I- and G-agents are the low-level entities that represent the heterarchical component of the system. They interact with each other to produce a feasible solution. For instance, they resolve scheduling conflicts such as the assignment of more than one job to the same time slot on the same machine. They choose with whom to communicate, but do not exchange any knowledge about the whole system. They act freely and flexibly within their field of vision, without any explicit external direct control. That is, the interaction of a pair of agents yields a decision that is reached through their mutual agreement. Their activities are based on "if–then" rules.

MAS^{H} has a reactive type architecture [32], where agents interact by directly exchanging information. The high-level M-agent can communicate with any low-level agent, while only low-level agents of different types can communicate with each other. The form of the low-level agents' cooperation may be represented by a bipartite graph, where a subgroup of the nodes constitutes G-agents and the other subgroup consists of I-agents. An arc from a node of the first subgroup to a node from the second subgroup represents a communication channel between the G- and I-agents tagged to the two nodes. Nodes from the same subgroup are unconnected. They indirectly compete among their subgroup for the best assignment to a node from the other subgroup. MAS^{H} does not employ any central symbolic entity or reasoning. Thus, its agents' behavior does not mimic any belief, intention or desire.

4.3. Solution construction

The solution construction adopts the notation of Table 2. Initially, a partial solution is obtained using a set N of free I-agents and a set G of G-agents. This solution is then completed by successively scheduling the free jobs. Section 4.3.1 motivates the choice of the agents, defines them, and constructs a partial feasible solution. Section 4.3.2 details the computation of the marginal increase of WET. Finally, Section 4.3.3 explains how the schedule is completed.

4.3.1. Initialization

The high-level M-agent of MAS^{H} initializes the system by creating the low-level agents. It creates n free I-agents, with each I-agent corresponding to one of the n jobs, and $N = \{1, \ldots, n\}$. It selects a subset $G \subset N$, $G \neq \emptyset$, and declares each element of G a G-agent, as described in Table A1. While the choice of I-agents is straightforward, the selection of G-agents requires determining their number $|G|$, their elements, and their competition mechanism.

A small $|G|$ impends the decentralization of the decision making process. It causes low levels of competition among G-agents. A large $|G|$ makes the G-agents compete for a reduced

Table 2
Notation.

G	is the set of created G-agents
G_i	is the set of G-agents assigned to machine i. $G_i \subseteq G$
$G[i][j]$	is a G-agent initiated by job j and assigned to machine i. $G[i][j] \in G_i$
$J_{G[i][j]}$	is the set of jobs attached to $G[i][j]$ and sorted in non-decreasing order of their completion times
$S_{G[i][j]}$	is the starting time of $G[i][j]$; i.e., of the first job of $J_{G[i][j]}$
$C_{G[i][j]}$	is the completion time of $G[i][j]$; i.e., of the last job in $J_{G[i][j]}$
N	is the set of free I-agents
N_i	is the set of jobs assigned to machine i where $N_i = \cup_{j \in G_i} J_{G[i][j]}$
WET($G[i][j]$)	is the WET of all jobs in $J_{G[i][j]}$
WET(N_i)	is the WET of all jobs of N_i; i.e., the total WET for machine i
WET($f, G[i][j]$)	is the minimal WET of all jobs of $N_i \cup \{f\}$ when f joins $G[i][j]$ (over all possible positions of f in the sequence of $J_{G[i][j]}$)
INC($f, ST_f, G[i][j]$)	is the increment of WET of $N_i \cup \{f\}$ when f, characterized by state ST_f, is attached to $J_{G[i][j]}$

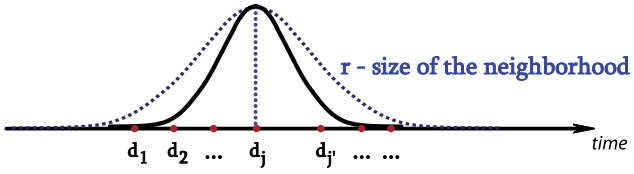

Fig. 2. Influence of r on the peak function.

set of free I-agents; so attaching I-agents becomes difficult. Therefore, the choice of $|G|$ should not hinder competition. It should not be prefixed, but self-adjusting.

To choose the elements of G-agents and set their competition mechanism, the M-agent mimics a decision maker's (DM) thought process. A DM identifies the bottlenecks of the production plan as time intervals requested by many jobs. S/he divides the jobs into independent clusters, and schedules the jobs of every cluster using some priority rule that minimizes WET. S/he does not deal with the clusters simultaneously unless they overlap over time. Subsequently, a G-agent competes for an I-agent if and only if the I-agent falls into its field of vision; i.e., if the I-agent's ideal processing period intersects the processing interval of the G-agent. A small number of I-agents falling into the field of vision of the G-agent leads to low levels of competition with a reduced number of possible sequences of jobs, and vice versa. To ensure an adequate level of competition, the M-agent selects the I-agents closest to the centers of the bottlenecks as G-agents. It identifies them using the peak clustering algorithm [33]. It evaluates a peak function

$$\phi_f^1 = \sum_{f' \in N} \exp\left(-\frac{\| d_f - d_{f'} \|^2}{(.5r)^2}\right)$$

for each job $f \in N$, where r is the size of the neighborhood of the Gauss function. r influences the shape of the peak function; for example, the solid-lined peak function of Fig. 2 changes into the dotted-lined one when r varies. In practice, r is set empirically or graphically from the peak functions whose x-axis correspond to due date values and their y-axis to their frequency. Varying r expands or narrows the width of a bottleneck interval. To remedy any erroneous choice of r, the M-agent applies a local search to every schedule.

A large ϕ_f^1 reflects a dense cluster around d_f; i.e., many jobs are competing for the time slots around d_f. The M-agent sets the job f_1^* having the largest peak function value $\phi_{f_1^*}^1 = \max_{f \in N}\{\phi_f^1\}$ as the center of the first ($k = 1$) cluster. Subsequently, the M-agent identifies machine i' that processes f_1^* fastest: $p_{i'f_1^*} = \min_{i=1,\ldots,m}\{p_{if_1^*}\}$, and associates f_1^* with a new G-agent $G[i'][f_1^*]$ which tags itself to i' by setting $G_{i'} = \{G[i'][f_1^*]\}$ and its group $J_{G[i'][f_1^*]} = \{f_1^*\}$. To have a zero-WET, $G[i'][f_1^*]$ schedules itself on i' during $[d_{f_1^*} - p_{i'f_1^*}, d_{f_1^*}]$. Finally, f_1^* marks itself busy, and leaves N.

The M-agent searches iteratively for other G-agents. At every iteration, it finds $f_{k+1}^* \in N$, the center of the next densest cluster. It evaluates, for $f \in N$, the modified peak function

$$\phi_f^{k+1} = \phi_f^k - \phi_{f_k^*}^k \exp\left(-\frac{\| d_f - d_{f_k^*} \|^2}{(.5r)^2}\right).$$

$\phi_f^{k+1} = 0$ for $f = f_k^*$, and takes small values for jobs with due dates around $d_{f_k^*}$. It chooses f_{k+1}^*, the job whose $\phi_{f_{k+1}^*}^{k+1} = \max_{f \in N}\{\phi_f^{k+1}\}$. If $\phi_{f_{k+1}^*}^{k+1} \geq 1$, MAS$^{\mathrm{H}}$ determines $\mathfrak{M}_{f_{k+1}^*}$, the set of machines that are free during the ideal processing interval of f_{k+1}^*. If $\mathfrak{M}_{f_{k+1}^*} \neq \emptyset$, the M-agent chooses the machine $i' \in \mathfrak{M}_{f_{k+1}^*}$ that processes f_{k+1}^* fastest: $p_{i'f_{k+1}^*} = \min_{i \in \mathfrak{M}_{f_{k+1}^*}}\{p_{if_{k+1}^*}\}$, and creates a new G-agent $G[i'][f_{k+1}^*]$. Subsequently, $G[i'][f_{k+1}^*]$ assigns itself to i' setting $G_{i'} = G_{i'} \cup \{G[i'][f_{k+1}^*]\}$ and $J_{G[i'][f_{k+1}^*]} = \{f_{k+1}^*\}$. Finally, f_{k+1}^* marks itself busy, and leaves N. On the other hand, if $\mathfrak{M}_{f_{k+1}^*} = \emptyset$, the M-agent cannot schedule f_{k+1}^* with a zero WET given the current partial schedule; i.e., f_{k+1}^* belongs to the field of vision of an existing G-agent and cannot serve as the center of a new cluster. In either case, the M-agent increments k and repeats its iterative step. It stops its iterations when $\phi_{f_{k+1}^*}^{k+1} < 1$. That is, when f_{k+1}^* is already part of a cluster; thus, cannot be a new cluster's center.

The partial schedule has a zero WET. The scheduled jobs constitute a preliminary set $G = \{G_1 \cup \cdots \cup G_m\}$ of G-agents. G is dynamically updated as the schedule is completed. This update is dictated by the competition and negotiation mechanisms of the agents, which base their decisions on the marginal increase of WET.

4.3.2. Computing the increment of WET

The goal of free I-agent f is to minimize the additional weighted earliness and tardiness it induces when attached to a group. When it receives an invitation to join a G-agent $G[i][j]$, f determines INC($f, ST_f, G[i][j]$), the additional WET it causes to the schedule of machine i. Let N_i be the ordered set of jobs already scheduled on i, and WET($f, G[i][j]$) the WET of the schedule of $N_i \cup \{f\}$ were f to join $G[i][j]$. Then, INC($f, ST_f, G[i][j]$) is the difference between WET(f, $G[i][j]$) and the current WET of N_i.

f computes WET($f, G[i][j]$) using the function **`ComputeMinimalWET`**($G[i][j], UB(Z_f)$), where $UB(Z_f)$ is an upper bound on WET($f, G[i][j]$). The function solves a mixed integer program (MIP) which inserts f immediately before, between two consecutive jobs of, or immediately after $J_{G[i][j]}$, and determines the completion time of every job of $N_i \cup \{f\}$. Inserting f may alter the completion times not only of jobs of $J_{G[i][j]}$ but of all jobs of N_i. If $N_i \neq J_{G[i][j]}$, then there is at least one more cluster of jobs scheduled on i. This cluster is before or after $J_{G[i][j]}$, and both its starting and completion times may be altered by scheduling f on i. Suppose $J_{G[i][j']}$ precedes $J_{G[i][j]}$ on i, and f is scheduled in $y = 0$, then

the completion time of j^-, the last job on $J_{G[i][j']}$, must be less than the starting time of f. Similarly, suppose $J_{G[i][j'']}$ succeeds $J_{G[i][j]}$ on i, and f is scheduled in $y=|J_{G[i][j]}|$, then the starting time of j^+, the first job on $J_{G[i][j'']}$, must be larger than the completion time of f. Evidently, j^- and j^+ may not exist if $J_{G[i][j]}$ is, respectively, the first or last group of jobs on machine i. In addition, the first job on i must start at or after time zero.

MIP uses three classes of variables. Integer variables T_ℓ, E_ℓ and C_ℓ represent the tardiness, earliness and completion time of the job in position $\ell \in N_i \cup \{f\}$. Integer variable C_0 denotes the completion time of the first job on machine i. Finally, binary variables $x_{[y]}=1$ when f is inserted in position $y \in Y=\{0,\ldots,|J_{G[i][j]}|\}$ within $J_{G[i][j]}$ and 0 otherwise. $y=0$ indicates that f is inserted immediately before $J_{G[i][j]}$ whereas $y=|J_{G[i][j]}|$ signals that f is inserted immediately after $J_{G[i][j]}$. Formally, MIP can be stated as

$$Z_f = \min \sum_{\ell \in N_i \cup \{f\}} (\alpha_\ell E_\ell + \beta_\ell T_\ell) \tag{11}$$

$$\text{s.t.} \quad C_\ell - d_\ell = T_\ell - E_\ell \quad \text{for all } \ell \in N_i \cup \{f\}, \tag{12}$$

$$C_\ell \le C_{\ell+1} - p_{i\ell+1} \quad \text{for all } \ell \in N_i, \tag{13}$$

$$C_f - p_{if} \ge M(x_y - 1) + C_{j^-} \quad \text{for all } y \in Y, \tag{14}$$

$$C_{j^+} - p_{ij^+} \ge M(x_y - 1) + C_f \quad \text{for all } y \in Y, \tag{15}$$

$$\sum_{y \in Y} x_y = 1 \tag{16}$$

$$C_0 \ge p_{i0} \tag{17}$$

$$C_f \ge p_{if} \tag{18}$$

$$x_y \in \{0,1\} \quad \text{for all } y \in Y, \tag{19}$$

$$C_\ell \in \mathbb{Z}^+,\ T_\ell \in \mathbb{Z}^+,\ E_\ell \in \mathbb{Z}^+ \quad \text{for all } \ell \in N_i \cup \{f\}, \tag{20}$$

$$Z_f < UB(Z_f). \tag{21}$$

Eq. (11) defines Z_f as the minimal sum of weighted earliness and tardiness of job f and of all jobs already assigned to machine i. Eq. (12) computes both the tardiness and earliness of job ℓ by setting their difference to the difference between ℓ's completion time and due date. If ℓ is tardy, $C_\ell - d_\ell > 0$; thus, $T_\ell > 0$ and $E_\ell = 0$. On the other hand, if ℓ is early, $C_\ell - d_\ell < 0$; thus, $T_\ell = 0$ and $E_\ell > 0$. When ℓ is on time, its $C_\ell - d_\ell = 0$; thus, $T_\ell = E_\ell = 0$. There are $|N_i \cup \{f\}|$ of these equations. Eq. (13) maintains the precedence relationships between pairs of successive jobs of N_i. The completion time of any job ℓ, $\ell \in N_i$, has to be less than or equal to the starting time of its immediate successor job $\ell+1$, where the starting time of job $\ell+1$ is defined as the difference between its completion time and processing time on machine i. Eq. (14) relates the starting time of job f when inserted in position y of $J_{G[i][j]}$ and the completion time of the job that is in position $y-1$ of $J_{G[i][j]}$; i.e., the job that immediately precedes f. This constraint is omitted when $y=0$ and j^- does not exist. There are $|Y|$ of these constraints with only one of them holding, and the others being redundant. Similarly, Eq. (15) relates the completion time of the job f inserted into position y of $J_{G[i][j]}$ and the starting time of the job that immediately succeeds f. This constraint is omitted when $y=|J_{G[i][j]}|$ and j^+ does not exist. There are $|Y|$ of these constraints with only one of them holding, and the others being redundant. Eq. (16) forces job f to be placed precisely in one of the available positions in $J_{G[i][j]}$. Eq. (17) ensures that the completion time of the first job scheduled on i is greater than or equal to its processing time; thus prevents the starting time of machine i from being negative. Eq. (18) is useful when $J_{G[i][j]}$ is the first cluster of jobs on i and f is positioned in $y=0$ within that sequence. Eq. (19) declares x_y as binary variables. Eq. (20) states that the completion time, earliness and tardiness of job $\ell \in N_i \cup \{f\}$ are all nonnegative integers. This equation can be relaxed so that the variables are non-negative reals since all input data is integer. Finally, Eq. (21) bounds Z_f by $UB(Z_f)$. A tight $UB(Z_f)$ may cause the infeasibility of MIP. This is indeed the purpose of Eq. (21).

Alternatively, **`ComputeMinimalWET`**$(G[i][j], UB(Z_f))$ could use an exact method (`Exact`), which proceeds as follows. `Exact` enumerates all sequences π_y, $y \in Y$, where π_y inserts f in position y within $N_i \cup \{f\}$. It evaluates the minimal WET of each sequence π_y, $y \in Y$, and retains the best WET among the $|Y|$ computed ones. Finally, it sets Z_f to the retained WET if this latter is less than $UB(Z_f)$; otherwise, it returns an infeasible solution.

To find the minimal WET of a sequence π_y, $y \in Y$, `Exact` searches for the optimal idle time between pairs of successive jobs of π_y. The optimal forced idleness is the solution of the timing algorithm `HS` [15]. The kth iteration, $k=1,\ldots,|Y|$, of `HS` left-shifts the starting time of the kth job of π_y from infinity until one of two cases occurs: either `HS` succeeds in scheduling job k to start at $d_k - p_{ik}$ and complete on-time at d_k, or `HS` stumbles against a block of jobs that are already scheduled during parts of $[d_k - p_{ik}, d_k]$. A block is a subset of jobs that are processed consecutively without any inserted idle time between any pair of successive jobs. In the second case, `HS` cannot schedule job k during $[d_k - p_{ik}, d_k]$. Therefore, it appends k to the block. At the end of the kth iteration, the schedule of the first k jobs of π_y is optimal. To optimize storage, retrieval, and minimize the number of operations, `HS` uses reversed stored lists of the breakpoints of the cost function.

The argument $UB(Z_f)$ of **`ComputeMinimalWET`**$(G[i][j], UB(Z_f))$ is made part of the decisional process of the I-agent of MAS$^{\mathrm{H}}$ so that it blocks unprofitable decisions, as explained below. The I-agent uses the states of the G-agents in its field of vision to avoid solving MIP or applying `Exact`; thus, to save computation time. A G-agent $G[i][j]$ is characterized by its state $ST_{G[i][j]}$. Similarly, an I-agent f is characterized by its state ST_f and its variable $\mathrm{INC}(f, ST_f, G[i][j])$. Both ST_f and $ST_{G[i][j]}$ are integer counters. Initially, ST_f is set to zero whereas $\mathrm{INC}(f, ST_f, G[i][j])$ is undefined. The state $ST_{G[i][j]}$ is initialized to the lowest level of a range of integer numbers that differ from those reserved to any other G-agent. Upon getting an attachment offer from $G[i][j]$, f computes $\mathrm{INC}(f, ST_f, G[i][j])$, stores its value, and sets $ST_f = ST_{G[i][j]}$. The value $\mathrm{INC}(f, ST_f, G[i][j])$ remains valid as long as the schedule of i is not altered. When a job is scheduled on i, the M-agent changes the state of every G-agent incrementing it by one: $ST_{G[i][j]} = ST_{G[i][j]} + 1$. Therefore, when considering an attachment offer, f recomputes $\mathrm{INC}(f, ST_f, G[i][j])$ if $ST_f \neq ST_{G[i][j]}$, and uses the current value otherwise. Most importantly, when $ST_f = ST_{G[i][j]}$, f uses $\mathrm{INC}(f, ST_f, G[i][j])$ as a guideline for group joining actions. f rejects joining $G[i][j]$ as soon as it identifies a G-agent $G[i'][j']$ such that $\mathrm{INC}(f, ST_f, G[i'][j']) < \mathrm{INC}(f, ST_f, G[i][j])$. Subsequently, the I-agent uses $\mathrm{INC}(f, ST_f, G[i][j])$ as part of $UB(Z_f)$ and injects it in MIP.

4.3.3. Completing the schedule's construction

MAS$^{\mathrm{H}}$ finishes the solution construction as detailed in Table A2 and Fig. 3. The communication diagram [12] of Fig. 3 is read from top to bottom with arrows corresponding to communication channels between agents. Longitudinal rectangles are states or internal decisional processes of the agents. Dotted lines express time intervals. A bifurcation indicates that the agent can send two types of information depending on its decision.

The M-agent sequentially asks the G-agents to undertake *group formation actions*, and stops its requests when all I-agents become busy; i.e., when $N=\varnothing$. Initially, it sets $i=1$ and $j=1$.

The M-agent activates G-agent $G[i][j]$ by giving $G[i][j]$ a CPU time to perform group formation. When active, $G[i][j]$ extends an

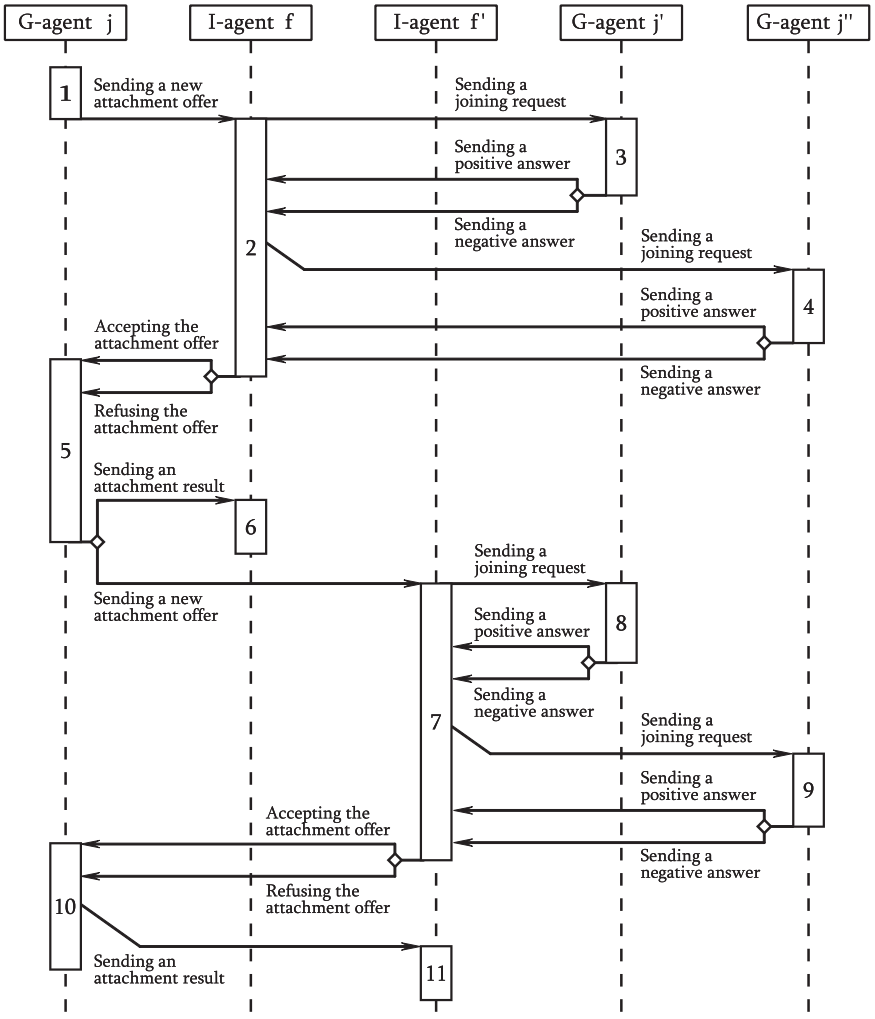

Fig. 3. Negotiation process within low-level MAS^{H} agents.

invitation to a subset of free I-agents to join its group $J_{G[i][j]}$ (cf. state 1 of Fig. 3). $G[i][j]$ calls the function **GroupFormation** (a, b) whose pseudocode is given in Table A3. This function proceeds as follows. *First*, $G[i][j]$ determines $V_{G[i][j]}$, the set of free I-agents that fall into its field of vision. $V_{G[i][j]}$ consists of jobs whose ideal processing periods overlap the group's processing period $[S_{G[i][j]}, C_{G[i][j]}]$, where $S_{G[i][j]}$ and $C_{G[i][j]}$ are the current starting and completion times of $G[i][j]$. *Second*, $G[i][j]$ sorts the elements of $V_{G[i][j]}$ in non-increasing order of the length of their overlap with its processing period $[S_{G[i][j]}, C_{G[i][j]}]$; that is, in terms of the size of their conflict with $G[i][j]$. It then selects the first I-agent f of $V_{G[i][j]}$, rather than choosing the job causing the least WET. Preliminary empirical results suggest that scheduling the "hardest" jobs of the bottleneck period first maintains their WET low during subsequent stages of the solution construction. *Third*, $G[i][j]$ sends f an attachment offer.

When invited to join $G[i][j]$, free I-agent f decides whether or not to accept the attachment offer (cf. state 2 of Fig. 3). To reach its decision, f undertakes a *group-joining action* by calling the function **GroupJoining** (a, b) whose pseudocode is given in Table A4.

If this is its first attachment offer, f determines the subset $\mathfrak{M}_f$ of machines i', $i' = 1, \ldots, m$, $i' \neq i$, that can schedule f during $[d_f - p_{i'f}, d_f]$ without overlapping the processing interval of any already scheduled job on i'. If $\mathfrak{M}_f \neq \emptyset$, f chooses machine i' that processes it fastest: $p_{i'f} = \min_{i \in \mathfrak{M}_f}\{p_{if}\}$. Subsequently, f asks the M-agent to create $G[i'][f]$, a new G-agent. $G[i'][f]$ tags itself to i', enters $G_{i'}$, and adds f to its group $J_{G[i'][f]}$ while f leaves N_i marks itself "busy", declines the attachment offer of $G[i][j]$ returning *false* as a result of the **GroupJoining** (a, b) function, and informs $G[i][j]$ of its decision.

If this is not its first attachment offer or $\mathfrak{M}_f$ *is empty*, f analyzes the current value $\mathrm{INC}(f, ST_f, G[i^*][j^*])$. $\mathrm{INC}(f, ST_f, G[i^*][j^*])$, which is non-negative, is the best result of the attachment offer that f examined last. f computes $\mathrm{INC}(f, ST_f, G[i][j])$, the least increment of WET that would result from f accepting to join $G[i][j]$. Two cases are distinguished. The first case occurs when $ST_f \neq ST_{G[i^*][j^*]}$; that is when machine i^* has scheduled one or more jobs since $G[i^*][j^*]$ had sent an attachment offer to f. Thus, the value of $\mathrm{INC}(f, ST_f, G[i^*][j^*])$ is no longer valid and cannot be used as part of an upper bound for $\mathrm{WET}(f, G[i][j])$. Therefore, f runs `ComputeMinimalWET` $(G[i][j], UB(Z_f))$ where $UB(Z_f) = \infty$. The second case occurs when $ST_f = ST_{G[i^*][j^*]}$; that is, when machine i^* had not scheduled any job since $\mathrm{INC}(f, ST_f, G[i^*][j^*])$ was computed. Thus, the value of $\mathrm{INC}(f, ST_f, G[i^*][j^*])$ is still valid. If, in addition, $G[i^*][j^*] = G[i][j]$, f sets $\mathrm{WET}(f, G[i][j]) = \mathrm{WET}(N_i) + \mathrm{INC}(f, ST_f, G[i^*][j^*])$. Otherwise (i.e., $G[i^*][j^*] \neq G[i][j]$), f considers joining $G[i][j]$ if and only if $\mathrm{INC}(f, ST_f, G[i][j]) < \mathrm{INC}(f, ST_f, G[i^*][j^*])$. Thus, f uses $\mathrm{INC}(f, ST_f, G[i^*][j^*])$ to compute an upper bound on the WET of machine i were f to join $G[i][j]$. It sets $UB(Z_f) = \mathrm{WET}(N_i) + \mathrm{INC}(f, ST_f, G[i^*][j^*])$, and calls `ComputeMinimalWET`$(G[i][j], UB(Z_f))$.

When run with $UB(Z_f) = \infty$, `ComputeMinimalWET` always returns a strictly positive value. On the other hand, when run with $UB(Z_f) < \infty$, `ComputeMinimalWET` may return a -1 value; indicating that MIP cannot find a feasible solution whose $Z_f <$

$UB(Z_f)$. Therefore, f declines the attachment offer and conveys its decision to $G[i][j]$.

A positive output of `ComputeMinimalWET` indicates that f may join $G[i][j]$ at a smaller cost than joining $G[i^*][j^*]$. Therefore, f updates its variable $\mathrm{INC}(f, ST_f, G[i][j]) = \mathrm{WET}(f, G[i][j]) - \mathrm{WET}(N_i)$, and considers all its alternatives. It sends every G-agent $G[i'][j'] \in G \backslash \{G[i][j]\}$ a joining request. Upon receipt of a request from f, $G[i'][j']$ processes it and returns a positive response to f if $[S_{G[i'][j']}, C_{G[i'][j']}] \cap [d_f - p_{i'f}, d_f] \neq \emptyset$, and a negative response otherwise (cf. state 3 of Fig. 3). When the answer of $G[i'][j']$ is positive, f computes $\mathrm{WET}(f, G[i'][j'])$ by running `ComputeMinimalWET` $(G[i'][j'], UB(Z_f))$ with $UB(Z_f) = \mathrm{WET}(N_{i'}) + \mathrm{INC}(f, ST_f, G[i][j])$.

A $\mathrm{WET}(f, G[i'][j']) \geq 0$ infers that $G[i'][j']$ can host f at a smaller cost than $G[i][j]$. Thus, f updates its state: $ST_f = ST_{G[i'][j']}$, saves $\mathrm{INC}(f, ST_f, G[i'][j']) = \mathrm{WET}(f, G[i'][j']) - \mathrm{WET}(N_{i'})$, and stores the arguments of its best potential G-agent: $a = i'$ and $b = j'$. It declines the attachment offer of $G[i][j]$ and conveys its decision to $G[i][j]$.

A $\mathrm{WET}(f, G[i'][j']) = -1$ indicates that $G[i'][j']$ cannot provide a better offer to f. So, f considers the next G-agent that sent f a positive reply, say $G[i''][j'']$ (cf. state 4 of Fig. 3). In case f does not find a better alternative after processing all G-agents that sent it a positive answer, it accepts the attachment offer of $G[i][j]$ and relays its decision to $G[i][j]$. In addition, it sends $G[i][j]$ the completion times of all jobs in $N_i \cup \{f\}$ and its position $y \in Y$ in $J_{G[i][j]}$ as obtained by MIP or `Exact`.

$G[i][j]$ reacts to the response of f (cf. state 5 of Fig. 3). *If f declines the invitation*, $G[i][j]$ initiates a new attachment offer and sends it to the next I-agent $f' \in V_{G[i][j]}$ (cf. states 7 of Fig. 3). On the other hand, *if f accepts the invitation*, $G[i][j]$ inserts it into $J_{G[i][j]}$ in position y. It asks the M-agent to update G_i, N_i, and the schedule of machine i according to the optimal solution obtained by MIP/`Exact` and received from f. Subsequently, the M-agent updates the completion times of the jobs of N_i and recomputes WET for machine i. $G[i][j]$ returns an attachment confirmation to f which declares itself "busy" (cf. state 6 of Fig. 3). To further reduce $\mathrm{WET}(G[i][j])$, $G[i][j]$ applies `LocalSearch`$(G[i][j])$ to every pair (ℓ, k) of jobs in $J_{G[i][j]}$ with $C_\ell < C_k$. Let $J_{\ell k} \subseteq J_{G[i][j]}$ be the sequence of jobs positioned between ℓ and k. For pair (ℓ, k), the search considers the sequences $(k, J_{\ell k}, \ell)$, $(J_{\ell k}, k, \ell)$ and $(k, \ell, J_{\ell k})$, and retains the one with the least WET. The search stops when no further reduction of $\mathrm{WET}(G[i][j])$ is possible. It marks the successful end of the group-formation action of $G[i][j]$, and returns a *true* value.

If no I-agent of $V_{G[i][j]}$ is attached during the group formation action, $G[i][j]$ ends its action unsuccessfully with an outcome *false*. If, on the other hand, the group-formation action of $G[i][j]$ has been successful, the M-agent tries to merge G-agents of G_i. It considers each pair of successive G-agents, say $G[i][j']$ and $G[i][j'']$, $j' = 1, \ldots, |G_i| - 1$, $j'' = j' + 1$. If there is no idle time between their processing intervals (i.e., $C_{G[i][j']} = S_{G[i][j'']}$), the M-agent merges the elements of $G[i][j'']$ into $G[i][j']$, concatenates the elements of $J_{G[i][j'']}$ to $J_{G[i][j']}$, sets $C_{G[i][j']} = C_{G[i][j'']}$, and deletes $G[i][j'']$ from G_i. Then, the M-agent asks the next G-agent in G to initiate a group formation.

It is possible that no G-agent of G attaches a job during a complete cycle of group-formation despite the existence of free jobs. This occurs when none of the processing intervals of the G-agents overlaps the ideal processing interval of any free job. It is signaled via a *false* value of the variable *Flag*. In this case, the M-agent creates a new G-agent $G[i'][\ell]$, where ℓ is the first free job in N and i' is the machine that processes it fastest. $G[i'][\ell]$ tags itself to i', enters $G_{i'}$, and sets its group $J_{G[i'][\ell]} = \{\ell\}$. Meanwhile, ℓ marks itself "busy" and leaves N. Next, the M-agent returns to asking G-agents to initiate group-formation actions.

Because MAS$^{\mathrm{H}}$ is decentralized, the M-agent gives the G-agents of G equal opportunities for group formation. It iterates sequentially through the elements of G allocating each element approximately equal access to CPU time. This equal-share strategy gives G-agents equal chances to attach I-agents; thus, to grow equally in size. However, MAS$^{\mathrm{H}}$ would gain were some G-agents granted a higher priority. Subsequently, the M-agent identifies G-agent $G[a][b]$ determined within the last group-joining action launched by f when it received an attachment offer from $G[i][j]$ but declined it. It redirects the solution construction process to $G[a][b]$ rather than to the next G-agent of G and requests from $G[a][b]$ to run its group-formation action. In this way, the M-agent decreases the number of group-formation/group-joining actions; thus avoiding numerous exact resolutions of MIP and/or `Exact` by skipping unpromising cooperations.

MAS$^{\mathrm{H}}$ is constructed such that a G-agent spanning a large time interval grows faster than a G-agent with a reduced time span. The former is more likely to find I-agents that overlap its processing interval and attach them than the latter which, after few iterations, can no longer extend attachment offers. Thus, an enlarged group is more likely to dominate a new group than a small group with few agents. This results in an unbalanced workload among machines and an inflated WET. Therefore, the M-agent exploits the redirection mechanism by turning on the redirect variable only when two successive G-agents are merged. This event is chosen as a kind of system's maturity indicator and shows that G-agents are large enough; thus redirection will reduce computations while granting no higher priority to one G-agent versus another. The M-agent turns redirection off each time a new G-agent is created; thus giving the "new" G-agent a chance to compete with existing ones.

4.4. Solution's enhancement

When a G-agent $G[i][j]$ attaches an I-agent, the M-agent may apply a steepest descent (SD) search which exchanges every pair $(G[i][j], G[i'][j'])$, $G[i'][j'] \in G_{i'}$, $i' = 1, \ldots, m$, $i' \neq i$, of G-agents whose $[S_{G[i][j]}, C_{G[i][j]}] \cap [S_{G[i'][j']}, C_{G[i'][j']}] \neq \emptyset$. Procedure `JobExchanging` $(G[i][j], G[i'][j'])$ is detailed in Table A5. Initially, it computes $\omega_{ii'}$, the current best WET of machines i and i' as $\omega_{ii'} = \mathrm{WET}(N_i) + \mathrm{WET}(N_{i'})$ and sets counters k and ℓ to zero. For its iterative step, it declares temporary variables $J^k_{G[i][j]} = J_{G[i][j]}$ and $J^\ell_{G[i'][j']} = J_{G[i'][j']}$ and increments the counters so that all pairs of positions (k, ℓ), $k = 0, \ldots, |G[i][j]|$, and $\ell = 0, \ldots, |G[i'][j']|$ are considered. Four cases arise.

- $k = 0$ and $\ell = 0$
  This corresponds to the current schedule of i and i' and is not further examined.
- $k = 0$ and $\ell > 0$
  No job is removed from $J_{G[i][j]}$ but j_ℓ, the job in position ℓ in $J_{G[i'][j']}$, is moved to the right of j_ρ, where j_ρ, the job in position ρ in $J_{G[i][j]}$, has the closest due date to the due date of j_ℓ. The exchange procedure updates the temporary variables setting $J^k_{G[i][j]} = J^k_{G[i][j]} \cup \{j_\ell\}$ and $J^\ell_{G[i'][j']} = J^\ell_{G[i'][j']} \backslash \{j_\ell\}$.
- $k > 0$ and $\ell = 0$
  No job is removed from $J_{G[i'][j']}$ but j_k, the job in position k in $J_{G[i][j]}$, is moved to the right of j_γ, where j_γ, the job in position γ in $J_{G[i'][j']}$, has the closest due date to the due date of j_k. The procedure sets $J^\ell_{G[i'][j']} = J^\ell_{G[i'][j']} \cup \{j_k\}$, $J^k_{G[i][j]} = J^k_{G[i][j]} \backslash \{j_k\}$.
- $k > 0$ and $\ell > 0$
  If $[d_\ell - (1+\xi)p_{i'\ell}, d_\ell + \xi p_{i'\ell}] \cap [d_k - (1+\xi)p_{ik}, d_k + \xi p_{ik}] \neq \emptyset$, where ξ is a positive real, jobs j_k and j_ℓ are swapped. In fact, j_k is removed from $J_{G[i][j]}$ and inserted in $J_{G[i'][j']}$ whereas j_ℓ is removed from $J_{G[i'][j']}$ and placed in $J_{G[i][j]}$. The procedure sets $J^\ell_{G[i'][j']} = J^\ell_{G[i'][j']} \cup \{j_k\} \backslash \{j_\ell\}$ and $J^k_{G[i][j]} = J^k_{G[i][j]} \cup \{j_\ell\} \backslash \{j_k\}$.

When j_ℓ is inserted in $J^k_{G[i][j]}$, the M-agent searches for a better position for j_ℓ within $J^k_{G[i][j]}$. It successively swaps j_ℓ with its neighbor and finds the jobs' completion times that minimize $\mathrm{WET}(J^k_{G[i][j]})$ using the algorithm `Timing`$(J^k_{G[i][j]})$ which applies HS of [15]. It stops its job swaps when `Repositioning` finds a position for j_ℓ that reduces $\mathrm{WET}(J^k_{G[i][j]})$. In this case, it creates a temporary variable $N^k_i = N_i \backslash J_{G[i][j]} \cup J^k_{G[i][j]}$ where $J^k_{G[i][j]}$ substitutes $J_{G[i][j]}$, and calls `Timing`(N^k_i) to optimize the completion times of all jobs of N^k_i. If $\omega_{ii'} \le \mathrm{WET}(N^k_i)$, it interrupts the job swaps and considers the next pair of positions (k,ℓ). When j_k is inserted in $J^\ell_{G[i'][j']}$, the above steps are similarly applied with the simple exchange of the indices k and ℓ. Finally, if $\omega_{ii'} < \mathrm{WET}(N^k_i) + \mathrm{WET}(N^l_{i'})$, the schedules of i and i' are updated by setting $N_i = N^k_i$ and $N_{i'} = N^\ell_{i'}$, respectively. The procedure stops when all possible pairs (k,ℓ) of all appropriate groups are examined.

5. Computational investigation

The computational investigation evaluates the performance of MAS$^{\mathrm{H}}$ in terms of solution quality and runtime; the two main measures of performance in combinatorial optimization.

Generally, an agent is an independent entity that controls its own decision making process [25]. For example, in a car assembly flow-line where substations are independent, each agent represents a substation with its own computing mechanism. This modeling approach is not applicable to PWET because it requires a large number of agents. MAS$^{\mathrm{H}}$ cannot provide each agent with an independent processor, computer or thread since the number of program threads is bounded by hardware capabilities. Alternatively, MAS$^{\mathrm{H}}$ simulates the independent tasks of agents by making its M-agent alternately allocate to each low-level agent the central processing unit (CPU) time it requires. This simulation ensures fairness among agents which get approximately equal access to the CPU for decision making purposes. Thus, MAS$^{\mathrm{H}}$ is a pseudo-parallel working environment where simultaneous tasks of multiple threads are emulated rationally to yield high quality results.

MAS$^{\mathrm{H}}$ is tested on two sets of benchmark instances: (i) those of [22] but adapted to the non-identical machine case, and (ii) those used in [18] for the identical machine case. For the first set, the results are compared to those obtained by the state-of-the-art hybrid approach H4, which was compared in [22] to existing approaches and to several variations of heuristics based on genetic algorithms, simulated annealing, and integer programming. For the second set, the results are compared to the lower bounds of [18] and to the best known upper bounds. In addition, the computational analysis assesses the effectiveness of MAS$^{\mathrm{H}}$ for different problem types and the impact of the hybridization. All the test instances used for the computational investigation along with corresponding upper bound values can be found on http://cs.adelaide.edu.au/~sergey/benchmarks/. MAS$^{\mathrm{H}}$ is coded in C# and run on a PC with an Intel Pentium Dual-Core processor 3.16 GHz and 4.0 Gb RAM. The MIP-based routines are computed using CPLEX 12.2 with a single thread. Sections 5.1 and 5.2 present the computational results for the first and second set of instances, respectively.

5.1. Results for the weighted case

Sections 5.1.1. and 5.1.2 present the experimental setup and results.

5.1.1. Experimental setup

An instance consists of n integer vectors $(d_j, p_{ij}, \alpha_j, \beta_j)$ corresponding to the due date, processing times on the m machines, and unit earliness and tardiness penalties of job j, $j = 1,\ldots,n$, where all data is rounded to the nearest integer. As in [22,23], the processing times p_{ij}, $i = 1,\ldots,m$, $j = 1,\ldots,n$, are normally distributed with mean 100 and standard deviation 10. Five levels are considered for the number of machines: $m = 2, 3, 4, 5, 10$. The penalties α_j and β_j, $j = 1,\ldots,n$, follow a uniform$[1, \overline{W}]$, where $\overline{W} =$ 6, 100. Two different scenarios are considered for the due dates: d_j, $j = 1,\ldots,n$, follow a normal distribution with mean $\mu_d = 50n/m$ and standard deviation $\sigma_d = \mu_d\, CV_d$, where the coefficient of variation $CV_d =$ 0.1, 0.2; and d_j, $j = 1,\ldots,n$, follow a uniform $[\underline{d}_{CV_d}, \overline{d}_{CV_d}]$, where $\underline{d}_{CV_d}$ and $\overline{d}_{CV_d}$ denote, respectively, the minimum and maximum due date of the n jobs when the due dates were generated normally with a coefficient of variation CV_d.

Two sizes of instances are considered: small-sized where $n =$ 10, 15, 20, 50, 100, 150, and large-sized where $n =$ 50, 100, 150, 200, 250, 300, 350. The large-sized ones are run using MAS$^{\mathrm{H}-}$ which consists of MAS$^{\mathrm{H}}$ without SD of Section 4.4. Thus, the duplicated instances for $n = 50$, 100, 150 are to assess the utility of the enhancement step.

SD swaps the current job with jobs in its immediate neighborhood, where the size of the neighborhood is reflected by ξ. Were runtime not an issue, all job swaps would be considered. In general, the larger ξ is, the higher SD's chances of improving the solution are; but at the cost of increased runtime. Different values of $\xi \in [0.5, 2.0]$ were investigated. For values of $\xi \in [1, 2]$, the solution quality was stable. Therefore, ξ is set to 1.0 for all experiments. This is in concordance with the conclusion of [22]: setting $\xi = 1$ yields, on average, the best overall performance for their heuristics.

For any set generated as described above, every instance κ, $\kappa \in \mathcal{K} = \{1,\ldots,10\}$, is run with different neighborhood sizes $r \in \Re = \{4, 10, 25, 40, 60, 70, 80, 100, 200, 300, 400, 1000, 2000,$ 4000$\}$. Let z^κ_r, $r \in \Re$, $\kappa \in \mathcal{K}$, denote the solution value obtained by MAS$^{\mathrm{H}}$ for the instance κ when MAS$^{\mathrm{H}}$ is run with a neighborhood size r, and let $\underline{z}^\kappa = \min_{r\in\Re}\{z^\kappa_r\}$ be the minimal solution value over all neighborhood sizes. In addition, let $\overline{\rho}_r = (1/|\mathcal{K}|)\sum_{\kappa\in\mathcal{K}}\rho^\kappa_r$, where $\rho^\kappa_r = z^\kappa_r/\underline{z}^\kappa$, $\kappa \in \mathcal{K}$, $r \in \Re$. Therefore, $\tilde{r}$ is the neighborhood size that yields the minimum average ratio over all neighborhood sizes: $\overline{\rho}_{\tilde{r}} = \min_{r\in\Re}\{\overline{\rho}_r\}$.

Subsequently, three solutions are reported for instance κ, $\kappa \in \mathcal{K}$: $\underline{z}^\kappa$, $\tilde{z}^\kappa = z^\kappa_{\tilde{r}}$ the solution value obtained by MAS$^{\mathrm{H}}$ when κ is run with $\tilde{r}$, and $\overline{z}^\kappa = \max_{r\in\Re}\{z^\kappa_r\}$. $\underline{z}^\kappa$ and $\overline{z}^\kappa$ are the best and worst solution values whereas $\tilde{z}^\kappa$ reflects the average behavior of MAS$^{\mathrm{H}}$.

In addition, t^κ_r, the run time of MAS$^{\mathrm{H}}$ when applied to instance κ with a neighborhood size r is used to calculate: the shortest runtime $\underline{t}^\kappa = \min_{r\in\Re}\{t^\kappa_r\}$, the average runtime $\tilde{t}^\kappa$, and $\overline{t}^\kappa = \max_{r\in\Re}\{t^\kappa_r\}$, the maximal observed runtime of MAS$^{\mathrm{H}}$ when applied to κ.

Finally, for each κ, $\kappa \in \mathcal{K}$, we compute z_{H^κ}, the solution value obtained by the hybrid heuristic H4 of [22] when applied with a population size of 350 and 500 generations. We record t_{H^κ}, the runtime of H4. Then, we compute $\overline{z}_{H^\kappa}$ by applying H4 to instance κ while constraining its runtime to $\overline{t}$.

5.1.2. Results

Tables 3–6 report the results. Specifically, Tables 3 and 5 report for each set the following averages: $\tilde{\rho} = (1/|\mathcal{K}|)\sum_{\kappa\in\mathcal{K}} z_{H^\kappa}/\tilde{z}^\kappa$, $\overline{\rho} = (1/|\mathcal{K}|)\sum_{\kappa\in\mathcal{K}} z_{H^\kappa}/\overline{z}^\kappa$, $\underline{\rho} = (1/|\mathcal{K}|)\sum_{\kappa\in\mathcal{K}} z_{H^\kappa}/\underline{z}^\kappa$, and $\hat{\rho} = (1/|\mathcal{K}|)\sum_{\kappa\in\mathcal{K}} z_{H^\kappa}/\overline{z}_{H^\kappa}$. The first three average ratios reflect the relative solution quality of MAS$^{\mathrm{H}}$ with respect to H4. $\underline{\rho}$ and $\overline{\rho}$ reflect the best and worst behavior of MAS$^{\mathrm{H}}$ whereas $\tilde{\rho}$ considers its average behavior. Finally, $\hat{\rho}$ reveals the quality of the solutions that H4 gets when constrained by the time limit $\overline{t}$. Every ratio is measured with respect to the best solution obtained by H4. A larger than 1.00

Table 3
MAS^{H} average relative solution quality for small-sized instances.

$CV_d = 0.1$		Uniform, $\overline{W} = 6$				Normal, $\overline{W} = 6$				Normal, $\overline{W} = 100$			
m	n	$\tilde{\rho}$	$\underline{\rho}$	$\overline{\rho}$	$\hat{\rho}$	$\tilde{\rho}$	$\underline{\rho}$	$\overline{\rho}$	$\hat{\rho}$	$\tilde{\rho}$	$\underline{\rho}$	$\overline{\rho}$	$\hat{\rho}$
2	10	1.03	1.03	1.00	0.99	1.02	1.03	1.00	0.98	1.04	1.06	1.01	0.97
	15	1.01	1.02	0.99	0.96	1.02	1.02	0.99	0.94	1.04	1.05	1.02	0.93
	20	1.04	1.04	1.01	0.94	1.01	1.02	1.00	0.94	1.06	1.06	1.02	0.91
	50	1.12	1.13	1.10	0.85	1.09	1.10	1.08	0.89	1.23	1.24	1.20	0.84
	100	1.14	1.15	1.12	0.76	1.14	1.15	1.13	0.80	1.30	1.31	1.30	0.72
	150	1.18	1.19	1.17	0.71	1.18	1.19	1.18	0.75	1.39	1.40	1.38	0.67
Average		1.09	1.09	1.07	0.87	1.08	1.08	1.06	0.88	1.18	1.19	1.15	0.84
3	10	1.06	1.07	1.01	0.99	1.05	1.08	1.01	0.98	1.09	1.12	1.00	0.97
	15	1.02	1.06	0.97	0.93	1.01	1.03	0.96	0.95	1.01	1.03	0.94	0.93
	20	1.02	1.03	0.98	0.93	1.01	1.04	0.99	0.95	1.03	1.05	0.97	0.92
	50	1.11	1.12	1.09	0.84	1.09	1.10	1.07	0.84	1.21	1.22	1.18	0.84
	100	1.19	1.19	1.16	0.75	1.14	1.15	1.13	0.82	1.27	1.28	1.25	0.75
	150	1.20	1.21	1.18	0.71	1.17	1.17	1.16	0.76	1.35	1.36	1.34	0.71
Average		1.10	1.11	1.06	0.86	1.08	1.09	1.05	0.88	1.16	1.18	1.11	0.85
4	10	1.03	1.07	0.94	0.98	0.98	1.02	0.90	0.97	1.03	1.06	0.90	0.97
	15	0.99	1.04	0.94	0.95	1.03	1.04	0.96	0.95	1.00	1.05	0.92	0.91
	20	1.04	1.06	0.99	0.93	1.01	1.03	0.97	0.95	1.03	1.06	0.98	0.92
	50	1.10	1.12	1.07	0.87	1.09	1.10	1.07	0.86	1.19	1.21	1.17	0.84
	100	1.19	1.20	1.17	0.77	1.16	1.17	1.15	0.82	1.26	1.27	1.24	0.75
	150	1.25	1.26	1.23	0.74	1.17	1.18	1.16	0.77	1.36	1.37	1.35	0.71
Average		1.10	1.12	1.06	0.87	1.07	1.09	1.03	0.89	1.15	1.17	1.09	0.85
5	10	1.01	1.05	0.89	0.99	1.01	1.06	0.87	0.98	1.05	1.13	0.80	0.96
	15	1.02	1.05	0.94	0.96	1.02	1.06	0.97	0.94	1.00	1.07	0.92	0.93
	20	1.00	1.03	0.93	0.91	1.01	1.06	0.98	0.93	1.04	1.10	0.98	0.93
	50	1.07	1.09	1.05	0.87	1.09	1.10	1.05	0.87	1.15	1.16	1.10	0.83
	100	1.18	1.19	1.16	0.79	1.15	1.16	1.14	0.82	1.26	1.28	1.24	0.77
	150	1.27	1.29	1.25	0.75	1.19	1.20	1.18	0.79	1.39	1.40	1.38	0.73
Average		1.09	1.12	1.04	0.88	1.08	1.11	1.03	0.89	1.15	1.19	1.07	0.86
10	10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	15	1.01	1.10	0.89	0.90	0.99	1.05	0.90	0.93	0.96	1.07	0.84	0.87
	20	0.94	1.01	0.90	0.91	0.96	1.03	0.89	0.89	1.00	1.05	0.86	0.85
	50	1.08	1.10	1.03	0.90	1.04	1.06	1.01	0.90	1.08	1.12	1.04	0.87
	100	1.14	1.18	1.11	0.83	1.12	1.13	1.10	0.85	1.20	1.22	1.17	0.79
	150	1.26	1.28	1.23	0.79	1.18	1.19	1.17	0.82	1.32	1.34	1.29	0.77
Average		1.09	1.13	1.03	0.89	1.06	1.09	1.01	0.90	1.11	1.16	1.04	0.86
Summary		1.09	1.12	1.05	0.87	1.07	1.09	1.04	0.88	1.15	1.18	1.09	0.85

$CV_d = 0.2$													
2	10	1.02	1.04	0.92	0.95	1.01	1.03	0.90	0.97	1.07	1.10	0.96	0.92
	15	1.05	1.08	0.94	0.97	1.01	1.03	0.94	0.93	1.11	1.13	0.98	0.89
	20	1.01	1.04	0.92	0.94	1.02	1.03	0.94	0.92	1.08	1.11	0.96	0.88
	50	1.05	1.12	0.99	0.82	1.13	1.14	1.08	0.83	1.29	1.32	1.21	0.75
	100	1.22	1.24	1.14	0.76	1.19	1.21	1.16	0.74	1.48	1.51	1.41	0.66
	150	1.27	1.30	1.22	0.79	1.28	1.30	1.26	0.71	1.65	1.71	1.61	0.63
Average		1.10	1.14	1.02	0.87	1.11	1.13	1.05	0.85	1.28	1.31	1.19	0.79
3	10	0.98	1.03	0.90	0.96	0.99	1.04	0.87	0.98	0.99	1.08	0.81	0.93
	15	1.00	1.05	0.87	0.94	0.98	1.02	0.87	0.94	1.02	1.10	0.81	0.91
	20	0.99	1.04	0.89	0.94	0.99	1.04	0.91	0.92	1.02	1.07	0.88	0.88
	50	1.06	1.12	1.00	0.85	1.11	1.15	1.06	0.85	1.24	1.32	1.18	0.80
	100	1.15	1.21	1.05	0.77	1.19	1.22	1.17	0.77	1.40	1.44	1.34	0.66
	150	1.26	1.31	1.17	0.75	1.27	1.30	1.24	0.72	1.64	1.68	1.58	0.65
Average		1.07	1.13	0.98	0.87	1.09	1.13	1.02	0.86	1.22	1.28	1.10	0.80
4	10	0.89	0.95	0.74	0.93	0.89	0.99	0.75	0.94	0.90	1.02	0.60	0.97
	15	0.98	1.03	0.81	0.91	0.98	1.03	0.79	0.95	0.94	1.02	0.67	0.86
	20	0.96	1.02	0.82	0.92	0.94	1.01	0.87	0.91	0.97	1.05	0.84	0.87
	50	1.04	1.10	0.92	0.85	1.11	1.16	1.07	0.83	1.28	1.33	1.17	0.82
	100	1.00	1.08	0.94	0.73	1.19	1.22	1.16	0.76	1.36	1.40	1.30	0.68
	150	1.24	1.32	1.18	0.75	1.31	1.33	1.28	0.73	1.64	1.70	1.59	0.63
Average		1.02	1.08	0.90	0.85	1.07	1.13	0.99	0.85	1.18	1.25	1.03	0.81
5	10	0.91	0.96	0.74	0.98	0.82	0.90	0.61	0.95	0.86	1.09	0.52	0.94
	15	0.82	0.92	0.73	0.91	0.88	0.96	0.74	0.92	0.85	1.00	0.68	0.88
	20	0.88	0.94	0.75	0.92	0.95	1.01	0.85	0.91	1.00	1.09	0.82	0.88
	50	0.96	1.01	0.85	0.80	1.09	1.11	1.01	0.86	1.17	1.22	1.06	0.76
	100	1.02	1.07	0.93	0.77	1.21	1.23	1.16	0.81	1.44	1.48	1.36	0.73
	150	1.07	1.15	0.98	0.73	1.30	1.32	1.27	0.73	1.68	1.73	1.63	0.68
Average		0.94	1.01	0.83	0.85	1.04	1.09	0.94	0.86	1.17	1.27	1.01	0.81
10	10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	15	0.89	1.02	0.72	0.90	1.06	1.12	0.87	0.91	0.97	1.08	0.82	0.89
	20	0.82	0.90	0.73	0.85	0.82	0.95	0.72	0.82	0.78	0.88	0.61	0.81
	50	0.88	0.94	0.75	0.74	1.01	1.04	0.92	0.85	0.96	1.04	0.87	0.77
	100	0.91	1.02	0.81	0.65	1.13	1.17	1.08	0.79	1.32	1.37	1.19	0.74
	150	1.00	1.10	0.90	0.63	1.25	1.28	1.21	0.81	1.45	1.53	1.41	0.73
Average		0.90	1.00	0.78	0.79	1.05	1.11	0.96	0.86	1.09	1.18	0.98	0.82
Summary		1.01	1.07	0.90	0.84	1.07	1.12	0.99	0.85	1.19	1.26	1.06	0.80

Table 4
MAS^{H} average relative run times for small-sized instances.

$CV_d = 0.1$		Uniform, $\overline{W} = 6$				Normal, $\overline{W} = 6$				Normal, $\overline{W} = 100$			
m	n	$\underline{\tau}$	$\tilde{\tau}$	$\overline{\tau}$	$\hat{t}$	$\underline{\tau}$	$\tilde{\tau}$	$\overline{\tau}$	$\hat{t}$	$\underline{\tau}$	$\tilde{\tau}$	$\overline{\tau}$	$\hat{t}$
2	50	89 | 201	75 | 184	57 | 163	2 | 1	75 | 157	64 | 146	52 | 131	2 | 1	77 | 161	65 | 148	52 | 132	2 | 1
	100	23 | 34	21 | 32	19 | 30	22 | 14	17 | 26	16 | 24	14 | 21	29 | 19	18 | 27	17 | 26	15 | 23	28 | 19
	150	10 | 14	10 | 13	9 | 13	95 | 70	7 | 9	6 | 9	5 | 7	151 | 111	7 | 9	7 | 9	6 | 8	144 | 105
3	50	105 | 386	82 | 359	67 | 329	2 | 0	101 | 337	77 | 316	62 | 292	2 | 1	95 | 341	73 | 319	54 | 298	2 | 1
	100	37 | 67	31 | 63	27 | 58	15 | 8	28 | 48	24 | 46	21 | 42	20 | 11	29 | 52	26 | 49	22 | 45	20 | 10
	150	17 | 26	15 | 24	14 | 23	60 | 38	12 | 17	11 | 16	9 | 15	89 | 59	12 | 18	11 | 17	10 | 15	86 | 57
4	50	109 | 766	81 | 682	62 | 622	2 | 0	104 | 652	78 | 614	63 | 581	2 | 0	100 | 659	75 | 625	57 | 587	2 | 0
	100	49 | 125	41 | 115	34 | 106	13 | 4	38 | 89	32 | 85	27 | 79	17 | 6	40 | 93	34 | 89	29 | 82	16 | 6
	150	23 | 42	20 | 39	18 | 36	48 | 25	18 | 31	16 | 29	14 | 27	62 | 35	19 | 32	17 | 30	15 | 28	60 | 33
5	50	116 | 1289	84 | 1134	66 | 1057	2 | 0	109 | 1216	80 | 1118	64 | 1062	3 | 0	106 | 1172	77 | 1081	63 | 1006	3 | 0
	100	56 | 198	47 | 184	38 | 170	12 | 3	46 | 145	38 | 139	31 | 131	15 | 4	47 | 150	38 | 141	32 | 132	15 | 4
	150	31 | 71	26 | 67	22 | 62	38 | 15	23 | 50	20 | 48	18 | 45	53 | 23	25 | 52	21 | 49	18 | 46	50 | 22
10	50	118 | 6405	92 | 6026	72 | 4932	3 | 0	100 | 6024	78 | 5643	62 | 4831	3 | 0	101 | 5872	79 | 5683	65 | 4658	3 | 0
	100	67 | 1012	52 | 916	43 | 833	13 | 1	61 | 898	46 | 844	38 | 790	16 | 1	62 | 935	48 | 873	39 | 822	15 | 1
	150	46 | 340	38 | 314	31 | 290	32 | 4	41 | 272	33 | 260	28 | 248	41 | 5	42 | 286	35 | 271	28 | 256	38 | 5
$CV_d = 0.2$													
2	50	251 | 1055	187 | 668	147 | 426	1 | 0	124 | 327	98 | 278	70 | 210	2 | 1	121 | 327	97 | 278	71 | 210	2 | 1
	100	146 | 334	102 | 232	76 | 154	6 | 4	34 | 50	29 | 45	22 | 35	15 | 9	34 | 52	29 | 46	22 | 35	15 | 9
	150	183 | 482	170 | 355	125 | 249	13 | 9	14 | 19	12 | 17	9 | 13	71 | 52	14 | 19	12 | 17	9 | 13	69 | 50
3	50	236 | 2054	176 | 1305	126 | 858	1 | 0	162 | 628	117 | 561	80 | 470	1 | 0	144 | 646	106 | 545	73 | 449	1 | 0
	100	204 | 754	160 | 498	119 | 329	3 | 1	49 | 95	43 | 87	34 | 73	11 | 5	49 | 97	41 | 88	32 | 72	11 | 5
	150	170 | 526	144 | 379	107 | 277	8 | 4	22 | 34	19 | 31	15 | 24	44 | 28	22 | 35	19 | 31	15 | 25	44 | 27
4	50	263 | 4077	189 | 2698	150 | 1713	1 | 0	157 | 1103	113 | 955	80 | 809	1 | 0	146 | 1103	106 | 952	78 | 826	1 | 0
	100	230 | 2406	204 | 1781	155 | 1413	3 | 1	66 | 169	52 | 151	42 | 133	9 | 3	65 | 175	53 | 154	40 | 128	9 | 3
	150	163 | 617	127 | 421	107 | 305	7 | 3	32 | 59	28 | 54	24 | 46	31 | 16	34 | 64	29 | 57	23 | 47	30 | 15
5	50	238 | 5074	178 | 4272	130 | 3419	1 | 0	158 | 1829	115 | 1547	84 | 1346	2 | 0	152 | 1983	110 | 1603	83 | 1387	2 | 0
	100	250 | 2842	199 | 2063	152 | 1495	3 | 0	73 | 261	58 | 237	49 | 212	8 | 2	75 | 303	61 | 254	48 | 223	8 | 2
	150	272 | 2182	215 | 1541	168 | 1261	5 | 1	40 | 93	34 | 86	30 | 78	26 | 10	43 | 98	36 | 90	29 | 79	25 | 10
10	50	235 | 24890	169 | 9850	125 | 9300	1 | 0	168 | 10665	127 | 8445	107 | 6404	2 | 0	157 | 9303	119 | 7210	96 | 6034	2 | 0
	100	244 | 6498	163 | 4376	125 | 3166	2 | 0	95 | 1436	73 | 1261	59 | 1103	8 | 0	104 | 1647	81 | 1389	65 | 1152	7 | 0
	150	226 | 4098	173 | 2917	145 | 2310	5 | 0	68 | 488	57 | 439	46 | 399	20 | 3	72 | 535	59 | 471	46 | 427	18 | 2

Table 5
MAS^{H-} average relative solution quality for large-sized instances.

$CV_d = 0.1$		Uniform, $\overline{W} = 6$				Normal, $\overline{W} = 6$				Normal, $\overline{W} = 100$			
m	n	$\tilde{\rho}$	$\underline{\rho}$	$\overline{\rho}$	$\hat{\rho}$	$\tilde{\rho}$	$\underline{\rho}$	$\overline{\rho}$	$\hat{\rho}$	$\tilde{\rho}$	$\underline{\rho}$	$\overline{\rho}$	$\hat{\rho}$
2	50	1.04	1.08	0.97	0.85	1.03	1.06	1.00	0.89	1.14	1.18	1.08	0.84
	100	1.05	1.08	1.01	0.76	1.08	1.10	1.06	0.80	1.21	1.26	1.17	0.72
	150	1.11	1.13	1.06	0.71	1.13	1.15	1.11	0.75	1.32	1.34	1.27	0.67
	200	1.18	1.20	1.14	0.71	1.15	1.16	1.12	0.74	1.31	1.35	1.29	0.66
	250	1.19	1.22	1.16	0.70	1.16	1.17	1.14	0.74	1.35	1.37	1.31	0.67
	300	1.22	1.24	1.18	0.69	1.19	1.20	1.17	0.73	1.41	1.43	1.36	0.65
	350	1.25	1.28	1.22	0.69	1.20	1.21	1.17	0.73	1.43	1.44	1.40	0.65
Average		1.15	1.17	1.10	0.73	1.13	1.15	1.11	0.77	1.31	1.34	1.27	0.70
3	50	0.98	1.04	0.93	0.84	1.00	1.04	0.96	0.84	1.08	1.12	1.00	0.84
	100	1.08	1.10	1.01	0.75	1.06	1.08	1.03	0.82	1.16	1.19	1.10	0.75
	150	1.07	1.10	1.02	0.71	1.10	1.12	1.07	0.76	1.25	1.28	1.22	0.71
	200	1.20	1.23	1.15	0.72	1.15	1.16	1.11	0.76	1.32	1.35	1.28	0.69
	250	1.22	1.24	1.16	0.72	1.16	1.17	1.14	0.75	1.39	1.42	1.36	0.68
	300	1.26	1.29	1.23	0.72	1.19	1.20	1.16	0.73	1.42	1.44	1.37	0.66
	350	1.33	1.35	1.28	0.74	1.21	1.22	1.17	0.74	1.44	1.46	1.40	0.68
Average		1.16	1.19	1.11	0.74	1.12	1.14	1.09	0.77	1.29	1.32	1.25	0.72
4	50	0.98	1.01	0.91	0.87	0.99	1.02	0.94	0.86	1.02	1.10	0.98	0.84
	100	1.02	1.07	0.98	0.77	1.07	1.09	1.03	0.82	1.11	1.17	1.08	0.75
	150	1.11	1.13	1.05	0.74	1.09	1.11	1.05	0.77	1.24	1.27	1.18	0.71
	200	1.17	1.21	1.12	0.73	1.13	1.15	1.11	0.76	1.31	1.33	1.26	0.70
	250	1.22	1.24	1.17	0.74	1.17	1.18	1.14	0.75	1.39	1.41	1.35	0.68
	300	1.31	1.34	1.26	0.75	1.20	1.21	1.17	0.75	1.44	1.47	1.40	0.70
	350	1.32	1.34	1.27	0.75	1.23	1.24	1.20	0.76	1.49	1.51	1.44	0.71
Average		1.16	1.19	1.11	0.76	1.13	1.14	1.09	0.78	1.29	1.32	1.24	0.73
5	50	0.94	0.98	0.88	0.87	0.97	1.01	0.93	0.87	1.00	1.05	0.91	0.83
	100	1.00	1.05	0.96	0.79	1.04	1.08	1.01	0.82	1.11	1.16	1.07	0.77
	150	1.09	1.13	1.03	0.75	1.11	1.12	1.06	0.79	1.24	1.28	1.19	0.73
	200	1.19	1.21	1.13	0.76	1.14	1.16	1.11	0.77	1.31	1.34	1.27	0.71
	250	1.25	1.27	1.20	0.76	1.16	1.18	1.13	0.77	1.36	1.40	1.33	0.72
	300	1.28	1.31	1.23	0.74	1.20	1.21	1.16	0.76	1.42	1.45	1.38	0.72
	350	1.35	1.37	1.29	0.75	1.23	1.24	1.19	0.76	1.48	1.50	1.44	0.72
Average		1.16	1.19	1.10	0.77	1.12	1.14	1.09	0.79	1.27	1.31	1.23	0.74
10	50	0.92	0.99	0.88	0.90	0.93	0.97	0.88	0.90	0.95	0.99	0.85	0.87
	100	0.98	1.01	0.92	0.83	1.00	1.02	0.95	0.85	1.03	1.07	0.97	0.79
	150	1.03	1.08	0.99	0.79	1.06	1.08	1.03	0.82	1.14	1.17	1.09	0.77
	200	1.10	1.13	1.05	0.79	1.10	1.13	1.08	0.81	1.24	1.26	1.19	0.76
	250	1.17	1.20	1.11	0.78	1.15	1.17	1.13	0.80	1.30	1.32	1.25	0.74
	300	1.26	1.30	1.21	0.80	1.17	1.19	1.14	0.79	1.37	1.39	1.33	0.76
	350	1.34	1.37	1.28	0.80	1.22	1.23	1.18	0.79	1.41	1.43	1.37	0.75
Average		1.12	1.15	1.06	0.81	1.09	1.11	1.05	0.82	1.20	1.23	1.15	0.78
Summary		1.15	1.18	1.10	0.77	1.12	1.14	1.09	0.79	1.27	1.30	1.23	0.73

Table 5 (*continued*)

$CV_d = 0.1$		Uniform, $\overline{W} = 6$				Normal, $\overline{W} = 6$				Normal, $\overline{W} = 100$			
m	n	$\tilde{\rho}$	$\underline{\rho}$	$\overline{\rho}$	$\hat{\rho}$	$\tilde{\rho}$	$\underline{\rho}$	$\overline{\rho}$	$\hat{\rho}$	$\tilde{\rho}$	$\underline{\rho}$	$\overline{\rho}$	$\hat{\rho}$
$CV_d = 0.2$													
2	50	0.92	0.99	0.82	0.82	1.01	1.07	0.93	0.83	1.11	1.20	1.00	0.75
	100	1.00	1.06	0.92	0.76	1.09	1.13	1.02	0.74	1.29	1.36	1.20	0.66
	150	1.05	1.10	0.90	0.80	1.19	1.22	1.13	0.71	1.51	1.57	1.41	0.63
	200	1.03	1.08	0.95	0.79	1.23	1.25	1.15	0.70	1.57	1.61	1.45	0.62
	250	1.08	1.15	0.98	0.78	1.27	1.29	1.20	0.71	1.57	1.61	1.47	0.61
	300	1.08	1.12	0.97	0.76	1.32	1.34	1.24	0.69	1.76	1.78	1.62	0.62
	350	1.04	1.07	0.95	0.76	1.35	1.37	1.28	0.71	1.85	1.89	1.73	0.63
Average		1.03	1.08	0.93	0.78	1.21	1.24	1.13	0.73	1.52	1.57	1.41	0.65
3	50	0.88	0.94	0.78	0.85	0.98	1.02	0.87	0.85	1.01	1.11	0.91	0.80
	100	0.88	0.94	0.78	0.77	1.05	1.10	0.98	0.77	1.18	1.23	1.06	0.66
	150	0.95	1.01	0.85	0.75	1.13	1.17	1.04	0.72	1.39	1.46	1.28	0.65
	200	0.85	0.91	0.75	0.76	1.22	1.24	1.13	0.71	1.53	1.60	1.41	0.66
	250	0.94	0.98	0.81	0.76	1.27	1.30	1.19	0.71	1.72	1.77	1.56	0.66
	300	0.89	0.94	0.79	0.75	1.35	1.38	1.23	0.73	1.87	1.92	1.71	0.67
	350	0.98	1.03	0.87	0.72	1.39	1.42	1.32	0.73	1.89	1.93	1.74	0.67
Average		0.91	0.96	0.80	0.76	1.20	1.23	1.11	0.75	1.51	1.58	1.38	0.68
4	50	0.79	0.89	0.70	0.85	0.93	1.00	0.85	0.83	0.98	1.06	0.88	0.82
	100	0.77	0.84	0.67	0.73	1.05	1.07	0.96	0.76	1.08	1.16	0.99	0.68
	150	0.96	1.01	0.80	0.75	1.12	1.16	1.04	0.73	1.31	1.39	1.21	0.63
	200	0.88	0.94	0.77	0.71	1.23	1.27	1.14	0.75	1.49	1.55	1.36	0.66
	250	0.96	1.02	0.82	0.72	1.29	1.32	1.19	0.74	1.66	1.72	1.49	0.67
	300	0.80	0.85	0.72	0.71	1.33	1.37	1.25	0.74	1.80	1.85	1.67	0.69
	350	0.97	1.07	0.88	0.69	1.41	1.44	1.29	0.76	1.88	1.94	1.75	0.70
Average		0.87	0.95	0.77	0.74	1.19	1.23	1.10	0.76	1.46	1.52	1.34	0.69
5	50	0.78	0.82	0.65	0.80	0.90	0.95	0.82	0.86	0.89	0.97	0.74	0.76
	100	0.76	0.84	0.68	0.77	0.99	1.06	0.93	0.81	1.08	1.16	0.99	0.73
	150	0.80	0.86	0.70	0.74	1.09	1.13	1.02	0.73	1.29	1.37	1.18	0.68
	200	0.80	0.87	0.71	0.67	1.21	1.25	1.12	0.74	1.50	1.57	1.38	0.69
	250	0.80	0.85	0.72	0.70	1.27	1.30	1.17	0.76	1.61	1.66	1.48	0.69
	300	0.85	0.90	0.74	0.67	1.37	1.39	1.25	0.76	1.79	1.84	1.63	0.70
	350	0.92	0.96	0.81	0.69	1.41	1.44	1.30	0.78	1.82	1.90	1.70	0.71
Average		0.81	0.87	0.71	0.72	1.18	1.22	1.09	0.78	1.42	1.49	1.30	0.71
10	50	0.69	0.77	0.59	0.74	0.83	0.88	0.77	0.85	0.77	0.84	0.69	0.77
	100	0.68	0.77	0.60	0.65	0.89	0.95	0.83	0.79	0.92	1.00	0.83	0.74
	150	0.73	0.80	0.64	0.63	0.99	1.04	0.95	0.81	1.05	1.13	0.98	0.73
	200	0.76	0.84	0.66	0.64	1.10	1.13	1.02	0.76	1.23	1.31	1.13	0.71
	250	0.82	0.89	0.74	0.63	1.18	1.22	1.12	0.77	1.35	1.41	1.25	0.71
	300	0.73	0.81	0.65	0.59	1.29	1.32	1.21	0.80	1.45	1.53	1.37	0.72
	350	0.78	0.84	0.68	0.62	1.35	1.37	1.27	0.79	1.64	1.67	1.51	0.73
Average		0.74	0.82	0.65	0.64	1.09	1.13	1.02	0.80	1.20	1.27	1.11	0.73
Summary		0.87	0.94	0.77	0.73	1.17	1.21	1.09	0.76	1.42	1.49	1.31	0.69

Table 6
MAS^{H-} average relative run times for large-sized instances.

$CV_d = 0.1$		Uniform, $\overline{W} = 6$				Normal, $\overline{W} = 6$				Normal, $\overline{W} = 100$			
m	n	$\underline{\tau}$	$\tilde{\tau}$	$\overline{\tau}$	$\hat{t}$	$\underline{\tau}$	$\tilde{\tau}$	$\overline{\tau}$	$\hat{t}$	$\underline{\tau}$	$\tilde{\tau}$	$\overline{\tau}$	$\hat{t}$
2	50	211|2064	147|1490	104|1075	1|0	218|2029	151|1501	102|1090	1|0	188|1912	134|1486	93|1124	1|0
	100	141|346	105|260	75|199	4|2	141|352	103|268	71|178	5|2	129|434	92|241	62|204	5|2
	150	106|126	85|99	65|75	11|9	101|133	78|101	55|59	12|9	100|133	68|100	40|67	14|10
	200	87|65	68|51	50|38	23|30	78|65	61|52	37|35	26|31	75|67	57|50	43|32	28|32
	250	70|39	56|31	43|24	41|74	60|39	52|32	44|25	46|75	58|40	47|31	36|22	51|77
	300	55|25	46|21	36|17	72|157	49|25	43|22	36|19	79|150	47|26	40|21	32|17	84|161
	350	44|18	39|15	31|11	113|300	39|18	34|16	30|13	132|291	37|18	32|15	27|12	140|301
3	50	159|3629	118|2819	87|2145	1|0	175|3979	118|2923	84|2149	1|0	158|3838	108|2728	78|1998	2|0
	100	124|706	90|462	65|339	5|1	117|670	80|480	49|322	6|1	106|705	73|490	51|351	7|1
	150	99|252	70|179	50|124	13|5	87|250	65|183	45|128	15|5	84|242	58|165	37|112	17|6
	200	88|115	62|82	45|63	24|18	74|134	54|91	39|57	29|18	69|121	49|88	36|60	32|18
	250	70|66	52|50	38|36	44|45	65|70	48|53	37|35	50|45	58|70	42|51	31|36	57|47
	300	61|45	45|32	34|24	68|97	55|47	40|34	27|18	83|98	52|44	37|32	26|20	89|102
	350	56|30	41|23	30|16	101|183	48|34	38|27	29|21	118|164	49|32	35|25	25|19	128|179
4	50	147|5963	105|4406	79|3147	2|0	134|6287	100|4642	75|3353	2|0	134|5728	92|4362	68|3110	2|0
	100	114|1028	81|751	58|541	6|1	98|1023	67|756	50|553	8|1	90|1128	69|773	52|564	8|1
	150	88|358	64|250	49|182	15|4	76|392	54|276	37|197	19|4	75|333	51|250	36|198	20|4
	200	75|170	54|126	39|96	28|12	61|183	45|128	33|89	36|13	56|170	42|124	31|91	39|13
	250	61|104	46|71	35|52	51|33	54|113	40|82	31|60	61|30	50|104	35|73	25|56	70|33
	300	57|64	41|44	30|32	76|71	50|69	34|50	24|37	98|67	44|64	32|44	23|32	105|75
	350	49|44	35|31	26|23	120|134	44|46	30|33	21|23	148|135	40|44	28|31	20|22	156|141
5	50	139|7662	101|6163	80|4956	2|0	128|8935	92|6606	67|4605	2|0	124|8607	88|6292	70|4571	2|0
	100	94|1513	77|1104	59|872	7|0	86|1473	62|1113	45|908	9|1	80|1382	60|1081	44|863	9|1
	150	79|513	61|374	49|296	16|3	68|512	49|387	38|301	22|3	69|531	48|354	37|282	22|3
	200	71|254	53|177	41|141	31|9	57|258	41|187	30|144	42|9	52|251	39|181	30|142	44|9
	250	59|132	44|97	36|75	53|24	46|145	35|104	26|82	73|24	46|130	33|100	25|79	76|25
	300	52|84	38|62	29|46	85|51	44|91	32|66	25|51	107|52	40|87	30|64	23|48	113|54
	350	43|52	33|40	25|32	128|104	36|62	27|43	20|32	170|105	34|56	25|41	19|32	179|109
10	50	125|23 380	94|17 515	73|11 690	3|0	105|24 590	79|13 540	63|10 642	3|0	114|22 730	88|11 365	69|10 985	3|0
	100	77|4869	63|3704	50|2851	10|0	67|4847	55|3855	44|3056	13|0	72|4872	59|3904	49|3187	12|0
	150	67|1844	55|1420	46|1133	22|1	57|1740	45|1461	37|1186	30|1	60|1857	47|1472	39|1222	28|1
	200	62|867	51|685	42|547	37|3	45|831	38|672	32|554	56|3	49|831	40|681	33|557	53|3
	250	53|459	44|366	36|303	62|7	40|445	32|362	26|295	94|8	41|446	32|361	26|289	92|8
	300	50|278	42|228	35|185	85|16	38|292	30|233	24|192	135|17	37|277	30|226	26|187	133|18
	350	48|187	38|149	30|124	120|30	31|176	25|146	21|122	203|35	31|180	25|142	21|120	202|36
$CV_d = 0.2$													
2	50	353|3825	233|2867	138|2032	1|0	252|2242	177|1593	115|1170	1|0	244|2173	168|1572	112|1136	1|0
	100	297|958	183|703	97|484	3|1	163|345	113|265	84|192	4|2	147|348	103|248	66|181	4|2
	150	303|1155	178|846	100|535	6|3	122|133	85|101	60|74	10|9	107|133	73|91	50|67	12|9
	200	356|870	239|613	157|442	8|4	82|67	67|53	51|43	22|27	73|63	56|47	40|34	25|30
	250	319|478	201|362	114|267	14|10	63|38	51|31	39|23	42|71	58|36	45|28	34|19	48|77
	300	328|359	188|281	100|199	21|23	46|24	39|20	32|15	78|155	42|23	35|18	26|14	88|168
	350	343|480	238|362	159|280	24|32	37|17	31|14	25|12	135|294	32|16	26|13	21|9	154|323
3	50	264|7152	180|5143	129|3740	1|0	208|4119	147|3128	103|2283	1|0	192|4091	133|2859	93|1933	1|0
	100	255|2483	164|1687	86|1116	3|0	143|725	106|528	75|381	4|1	131|723	94|501	69|376	5|1
	150	257|1391	137|973	70|719	6|1	106|251	75|183	53|130	11|5	105|244	71|169	51|119	12|5
	200	336|2139	205|1475	130|1175	10|1	93|126	63|89	48|69	22|16	87|126	57|85	37|58	24|16
	250	249|843	154|673	94|510	14|4	76|73	53|53	37|39	40|40	70|67	50|49	34|36	42|43
	300	269|984	173|752	118|585	24|7	59|45	44|33	34|24	66|88	56|44	38|30	25|22	76|96
	350	245|589	154|438	106|337	32|16	51|33	40|24	29|18	97|161	47|30	35|22	24|15	112|174

Table 6 (*continued*)

$CV_d = 0.1$		Uniform, $\overline{W} = 6$				Normal, $\overline{W} = 6$				Normal, $\overline{W} = 100$			
m	n	$\underline{\tau}$	$\tilde{\tau}$	$\overline{\tau}$	$\hat{t}$	$\underline{\tau}$	$\tilde{\tau}$	$\overline{\tau}$	$\hat{t}$	$\underline{\tau}$	$\tilde{\tau}$	$\overline{\tau}$	$\hat{t}$
4	50	223|11 235	160|8237	118|6649	1|0	169|6910	117|4978	84|3542	1|0	183|7035	120|4688	86|3213	1|0
	100	253|4730	173|3852	108|2999	3|0	124|1154	87|848	64|666	5|1	116|1153	86|825	63|625	5|1
	150	178|1741	116|1271	76|951	7|1	95|395	70|292	54|226	13|3	92|391	69|292	49|223	13|3
	200	249|2309	166|1525	111|1180	9|2	83|197	60|145	45|111	24|10	75|181	56|135	40|101	25|10
	250	201|1176	134|869	83|642	16|4	72|114	50|84	37|64	42|25	64|105	46|77	35|62	46|27
	300	238|1543	184|1263	131|1020	15|2	58|71	42|50	32|37	69|58	49|63	38|47	29|35	76|61
	350	203|680	113|534	71|405	38|13	54|44	37|33	28|25	105|117	49|43	34|32	27|24	111|119
5	50	210|12 270	151|12 270	116|8118	1|0	152|10 380	109|7832	80|5375	2|0	151|9070	111|7457	88|4831	2|0
	100	250|5853	160|4519	124|3639	3|0	115|1647	82|1291	64|1066	6|0	115|1803	83|1284	61|983	6|0
	150	203|4137	152|3190	108|2500	6|0	91|588	68|454	52|346	13|2	91|584	68|436	51|344	13|2
	200	210|2786	137|1936	93|1509	10|1	82|280	61|207	48|159	24|7	74|287	58|200	44|146	25|7
	250	212|2195	158|1752	122|1429	12|2	63|149	49|111	37|88	44|19	62|147	46|108	35|83	47|20
	300	181|1564	127|1123	88|875	23|3	58|95	44|73	36|56	66|40	56|90	41|70	30|54	70|41
	350	204|1371	156|1025	122|793	26|6	50|66	38|50	31|39	101|78	47|61	35|47	26|36	108|82
10	50	182|15 170	142|14 099	104|14 490	1|0	155|20 520	114|20 520	91|17 335	2|0	134|17 580	97|17 580	83|14 830	2|0
	100	173|12 467	124|8922	95|6742	3|0	97|6016	76|4609	60|3655	8|0	104|5834	75|4765	65|3977	8|0
	150	144|7570	119|5696	99|4428	6|0	85|2345	70|1859	57|1467	16|1	84|2429	68|1906	55|1538	16|1
	200	187|6888	149|4489	116|3355	9|0	73|1121	59|874	47|718	29|2	72|1119	57|860	45|684	29|2
	250	159|4290	121|3100	98|2423	17|1	64|583	53|455	43|360	46|5	63|582	53|463	40|377	46|5
	300	197|4632	147|3253	118|2539	17|1	64|392	50|299	40|238	64|11	62|388	50|299	39|242	64|11
	350	222|4492	172|2956	140|2350	20|1	54|231	43|186	35|152	98|23	53|226	43|186	34|153	98|22

average ratio confirms the superiority of MAS^{H} in terms of solution quality or rapid convergence.

Tables 4 and 6 report $\underline{\tau} = (1/|\mathcal{K}|)\sum_{\kappa \in \mathcal{K}} t_{H^\kappa}/\underline{t}^\kappa$, $\overline{\tau} = (1/|\mathcal{K}|)\sum_{\kappa \in \mathcal{K}} t_{H^\kappa}/\overline{t}^\kappa$, $\tilde{\tau} = (1/|\mathcal{K}|)\sum_{\kappa \in \mathcal{K}} t_{H^\kappa}/\tilde{t}^\kappa$, and $\hat{t} = (1/|\mathcal{K}|)\sum_{\kappa \in \mathcal{K}} \tilde{t}^\kappa$. The first three average ratios reflect the relative runtime of MAS^{H} with respect to H4. $\underline{\tau}$ and $\overline{\tau}$ are based on the shortest and longest runtime of MAS^{H} whereas $\tilde{\tau}$ considers the typical behavior of MAS^{H}. Every ratio is measured with respect to the runtime of H4. A larger than 1 average ratio confirms that MAS^{H} is faster than H4. Finally, $\hat{t}$ provides an average running time measured in seconds. Each cell of Tables 4 and 6 contains two MAS^{H} solution values separated by a line "|". The first corresponds to MAS^{H} applying MIP to compute the increment of WET while the second corresponds to MAS^{H} employing `Exact`. MIP and `Exact` may produce alternative optima for a given subset of jobs; eventually leading MAS^{H} to different local optima for the instance. However, the computational results suggest that the values of the local optima are too close to warrant including them in Tables 3 and 5. The ratios of the two solutions are identical to 10^{-4}.

The analysis of Tables 3–6 discerns the impact of the following factors on the solution quality and runtime of MAS^{H}: (i) MIP versus `Exact`, (ii) the rule of thumb for the choice of I-agents, (iii) SD, (iv) the problem size, (v) the weights, and (vi) the spread/distribution of the due dates.

Impact of the exact approach used to compute WET: MAS^{H} computes the increment of WET by solving MIP or by employing `Exact`. For instances with a large n and small m, MIP is faster whereas `Exact` performs better for instances with a small n or large m. When using MIP, MAS^{H} devotes some of the runtime to interface the solver with the code; i.e., to input the data to the mathematical model and to read the output results. However, this interface consumes at most five percents of the total run time; thus, does not have a sizeable impact on the runtime of MAS^{H}. That is, MIP and `Exact` behave differently for a given problem size. The analysis of the solution structure and its construction process suggest that MIP is generally preferred to `Exact` when MAS^{H} deals with very dense clusters of jobs.

Impact of the rule of thumb for the choice of I-agents: In the implementation of MAS^{H}, G-agent $G[i][j]$, tagged to machine $i \in \{1, \ldots, m\}$ and initiated by a job $j \in \{1, \ldots, n\}$, identifies the I-agent whose ideal processing period $[d_{j'} - p_{ij'}, d_{j'}]$ on i has the largest overlap with the processing period $[S_{G[i][j]}, C_{G[i][j]}]$ of $G[i][j]$. Consider alternatively the rule of thumb where G-agent $G[i][j]$ chooses the I-agent with the minimal WET on machine i. Let MAS$^{H}_1$ denote MAS^{H} when applying this alternative rule of thumb. To assess the impact of the selection rule applied by the G-agents, we apply both MAS^{H} and MAS$^{H}_1$ on the set of large sized instances with normally distributed due dates, uniformly distributed penalties in the range [1, 6], and CV=0.1, and record for each instance the best solution values $Z_{\text{MAS}^{H}}$ and $Z_{\text{MAS}^{H}_1}$, obtained by MAS^{H} and MAS$^{H}_1$. A paired t-test indicates that the mean of $Z_{\text{MAS}^{H}_1} - Z_{\text{MAS}^{H}}$ is larger than 0 at any significance level while the run times are on average equal. This substantiates that the current selection rule is better than the alternative one (which happens to be more intuitive). Table 7 displays the average $Z_{\text{MAS}^{H}_1}/Z_{\text{MAS}^{H}}$ obtained for different problem sizes and machine environments.

Impact of steepest descent: To discern the impact of SD, we consider the instances with n=50, 100, 150, run with MAS^{H} and MAS^{H-}. The results were respectively reported as parts of Tables 3 and 5 for solution quality and of Tables 4 and 6 for runtime. Since the instances are identical, we apply paired t-tests and conclude that there is sufficient statistical evidence to claim that the mean improvement that SD induces on $\tilde{\rho}, \underline{\rho}$, and $\overline{\rho}$ is strictly positive at any significance level. It is larger than 0.16, 0.14 and 0.19 for $\tilde{\rho}, \underline{\rho}$, and $\overline{\rho}$ at the 95% confidence level. It is largest on the worst solutions obtained by MAS^{H-}.

Table 7
Average $Z_{MAS_1^H}/Z_{MAS^H}$ for different problem sizes.

n	$m=2$	$m=3$	$m=4$	$m=5$	$m=10$	Overall
50	1.0096	1.0080	1.0070	1.0068	1.0219	1.0107
100	0.9999	1.0044	1.0070	1.0172	1.0151	1.0087
150	1.0124	1.0139	1.0161	1.0137	1.0193	1.0151
200	1.0115	1.0103	1.0106	1.0177	1.0146	1.0129
250	1.0152	1.0136	1.0119	1.0183	1.0138	1.0146
300	1.0117	1.0200	1.0147	1.0150	1.0165	1.0156
350	1.0143	1.0175	1.0196	1.0177	1.0136	1.0166
Overall	1.0107	1.0125	1.0124	1.0152	1.0164	1.0134

These improvements occur at the cost of runtime. The mean increase of run time caused by SD is at least 9 s at the 95% level and is strictly positive at any level. However, this additional time is negligible when compared to the runtime of H4. There is no statistical evidence that the ratios $\tilde{\tau}, \underline{\tau}$, and $\overline{\tau}$ for MAS^H differ from their counterparts for MAS^{H-}. The growth rate of SD's runtime is largest for $m=2$ and $n=150$. It decreases as m increases and n decreases. This suggests that applying SD for large-sized instances could slow MAS^H.

Effect of problem size: The ratios $\tilde{\rho}$, $\underline{\rho}$, and $\overline{\rho}$ are weakly negatively correlated to m, with -0.202, -0.179, and -0.196 respective correlation coefficients at any level of significance. The ratios are moderately positively correlated to n with 0.562, 0.553, and 0.557 respective correlation coefficients at any level of significance. This suggests that H4 obtains slightly better solutions than MAS^H as m increases, but MAS^H gets better solutions than H4 as n increases. There is no statistical evidence that the run time of MAS^H is correlated to m with a correlation coefficient of 0.014 (with a 0.786 type I risk of error). However, the runtime of MAS^H is strongly positively correlated to n with the 0.803 correlation coefficient being significant at all levels. As n increases, $\tilde{\tau}$, $\underline{\tau}$, and $\overline{\tau}$ decrease with the correlation level being moderate but significant at all levels; i.e., their respective correlation coefficients are -0.562, -0.575, and -0.588. This suggests that the runtime of MAS^H grows faster than the runtime of H4 as n increases. Even though a similar trend seems to exist for $\tilde{\tau}$, $\underline{\tau}$, and $\overline{\tau}$ as m increases, there is not sufficient statistical support for the claim. The correlation coefficients between $\tilde{\tau}$, $\underline{\tau}$, $\overline{\tau}$ and m are -0.123, -0.095, -0.073 at respective significance levels of 0.015, 0.061, and 0.152.

To further assess the behavior of MAS^H on large-scale problems, we consider instances with a larger number of machines and jobs (i.e., $m=10$, 20, 30, 50, 75, 100 and $n=500$, 750, 1000, 1250, 1500). Fig. 4 illustrates the growth of the runtime of MAS^H as m increases for different n whereas Fig. 5 illustrates the growth of the runtime of MAS^H as n increases for different m.

Effect of distribution of due dates: For identical ranges, uniformly distributed due dates have a higher variation than those generated using a truncated normal. Similarly, as CV_d increases, the range of due dates increases and their variation becomes larger. As the variation of due dates increases, MAS^H becomes faster with the mean reduction of runtime being statistically significant at all levels and being larger than 13.09 s at the 95% confidence level. Furthermore, as the dispersion level of due dates increases, $\tilde{\tau}$, $\underline{\tau}$, and $\overline{\tau}$ increase; that is, the runtime of MAS^H decreases at a faster rate than the runtime of H4. The mean difference of the growth rates is statistically significant at all levels. This is expected since MAS^H is designed upon the idea of few dense clusters of due dates. So, the more spread the due dates are, the larger the number of clusters is and the smaller the number of jobs per cluster. This results in (i) a smaller size of the subproblems whose solutions indicate the best position of the job being inserted within the group, and (ii) in a fewer group-formation and group-joining actions. These two factors speed up MAS^H.

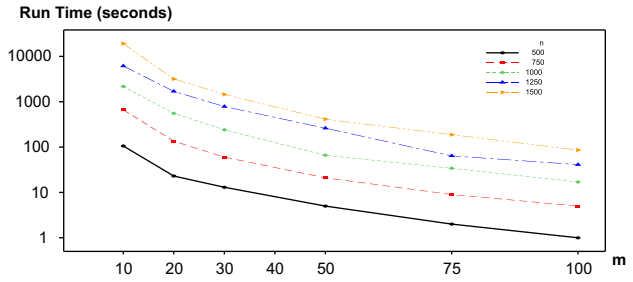

Fig. 4. Runtime of MAS^H for large numbers of machines.

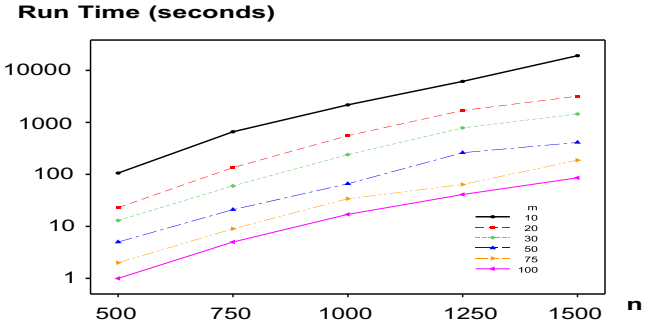

Fig. 5. Runtime of MAS^H for large numbers of jobs.

Table 8
MAS^H average relative solution quality and run time for uniform $[\underline{d}_{0.2}, \overline{d}_{0.2}]$ due dates and $\overline{W}=6$.

m	n	$\tilde{\rho}$	$\underline{\rho}$	$\overline{\rho}$	$\underline{\tau}$	$\tilde{\tau}$	$\overline{\tau}$	$\hat{t}$
2	200	1.24	1.27	1.17	207.13	167.04	146.96	14.30
	250	1.35	1.39	1.30	155.45	124.50	99.97	32.50
	300	1.35	1.37	1.29	137.04	115.24	99.67	66.30
	350	1.30	1.33	1.25	189.88	142.87	113.26	82.80
Average		1.31	1.34	1.25				
3	200	1.09	1.14	1.02	314.70	257.50	198.41	7.20
	250	1.23	1.29	1.17	186.41	156.64	131.91	19.70
	300	1.19	1.23	1.13	230.16	183.52	153.18	27.90
	350	1.39	1.41	1.30	152.64	120.02	103.40	53.70
Average		1.23	1.26	1.15				
4	200	1.23	1.30	1.15	205.07	185.56	152.87	12.00
	250	1.36	1.39	1.26	169.74	140.91	119.06	24.80
	300	1.07	1.12	1.01	292.50	228.80	197.30	12.50
	350	1.39	1.46	1.34	146.91	122.16	104.36	54.10
Average		1.26	1.32	1.19				
5	200	1.16	1.23	1.06	210.74	185.21	147.22	9.40
	250	1.10	1.16	1.02	274.17	206.94	178.02	11.60
	300	1.25	1.31	1.16	208.67	171.94	145.64	19.50
	350	1.35	1.40	1.24	229.68	177.40	151.41	36.10
Average		1.21	1.28	1.12				
10	200	1.06	1.14	0.95	258.75	188.61	152.18	8.10
	250	1.18	1.27	1.08	223.20	170.12	136.79	12.40
	300	1.02	1.10	0.93	275.44	227.03	189.52	10.50
	350	1.08	1.17	0.97	335.63	243.77	200.49	12.80
Average		1.08	1.17	0.98				
Summary		1.22	1.27	1.14				

The statistical analysis of the relationship between the ratios $\tilde{\rho}$, $\underline{\rho}$, and $\overline{\rho}$ and the spread of due dates suggests that MAS^H gets, on average, better solutions than H4 as the dispersion of the due

dates increases. However, when CV_d=0.1 and the due dates are normally distributed, MAS^{H} misses few near-global minima obtained by H4 and improves a few with the size of improvement being very moderate; i.e., most of the ratios and their average are in the neighborhood of 1. As the spread of due dates increases, the deviations of MAS^{H} solutions from the local minima obtained by H4 become more sizeable regardless of the direction of the deviation.

Effect of weights: As $\overline{W}$ increases, the absolute and relative performance of MAS^{H} with respect to H4 improves. The larger diversity of the weights offers MAS^{H} more discrimination power in choosing the neighborhoods of local minima and in converging to good quality ones rapidly. The ratios $\tilde{\rho}$, $\underline{\rho}$, and $\overline{\rho}$ are positively weakly correlated to $\overline{W}$; suggesting that MAS^{H} gets better solutions than H4 as $\overline{W}$ increases. This is further confirmed by the mean difference of $\tilde{\rho}$, $\overline{\rho}$, $\underline{\rho}$ when $\overline{W}=100$ and $\overline{W}=6$ being at least 0.13, 0.15, and 0.10 at the 95% level. In fact, MAS becomes more capable of finding near global minima than H4 as $\overline{W}$ increases. The mean difference of runtime is significant at all levels with the mean time reduction induced by the larger diversity of the weights exceeding 0.75 s at the 95% level. However, the improvement of runtime is not unique to MAS^{H}, but is similarly observed for H4. In fact, there is no difference between the means of $\tilde{\tau}$, $\overline{\tau}$, $\underline{\tau}$ when $\overline{W}=6$ and $\overline{W}=100$.

Summary of results: MAS^{H-} is a very competitive approach for PWET providing near-global minima in reduced run times. It is particularly suited for instances with groups of jobs clustered around the due dates. When the due dates are widespread, MAS^{H-} can be augmented with SD. Its results become comparable and often better than those obtained by H4. Table 8 presents the relative solution quality and runtime of MAS^{H} applying MIP with respect to H4 when $\overline{W}=6$ and the due dates follow a uniform $[\underline{d}_{CV_d}, \overline{d}_{CV_d}]$, with CV_d=0.2. It confirms that, when the due dates are widespread, SD can be applied even to large-sized instances while maintaining reasonably small run times. Indeed, the comparison of the results displayed in Table 8 to those of Tables 5 and 6 provides computational proof of the sizeable improvements brought up by SD without much impending the runtime for this class of instances.

5.2. Results for the unweighted case

The set of Keddad-Sidhoum et al. [18] has five instances for each combination of n, m, tardiness factor, and relative range of due dates, where $n=30, 60, 90$, $m=2, 3, 4$, the tardiness factor $\in\{0.2, 0.5, 0.8\}$, and the relative range of due dates $\in\{0.2, 0.5, 0.8\}$. Each instance $\kappa\in\mathcal{K}=\{1,\ldots,5\}$ has a known lower bound LB$^{\kappa}$ and a known upper bound UB$^{\kappa}$. For each instance $\kappa\in\mathcal{K}$, we apply MAS^{H} and report its best solution value $\underline{z}^{\kappa}$, and its smallest runtime $\underline{t}^{\kappa}$. Subsequently, we compute two ratios that assess the solution quality of MAS^{H} and two ratios that assess its relative runtime.

The first ratio $UB^{\kappa}/\underline{z}^{\kappa}$ compares the solution value $\underline{z}^{\kappa}$ obtained by MAS^{H} to the best known upper bound UB$^{\kappa}$ for the instance. When larger than 1.00, the ratio indicates that MAS^{H} improves UB$^{\kappa}$. The second ratio $\underline{z}^{\kappa}/LB^{\kappa}$ reflects the tightness of the lower bound LB$^{\kappa}$. This ratio is necessarily larger than or equal to 1 with $\underline{z}^{\kappa}=LB^{\kappa}$ when the solution corresponding to $\underline{z}^{\kappa}$ is optimal. The closer the ratio is to one, the tighter the optimality gap for the instance is; however, a large ratio does not necessarily translate into a bad performance of MAS^{H} since LB$^{\kappa}$ could be loose. The third and fourth ratios $t_{KSS}/\underline{t}^{\kappa}_{\text{MIP}}$ and $t_{KSS}/\underline{t}^{\kappa}_{\texttt{Exact}}$ compare t_{KSS}, the runtime reported by [18], to the run times $\underline{t}^{\kappa}_{\text{MIP}}$ and $\underline{t}^{\kappa}_{\texttt{Exact}}$ of MAS^{H} which applies, respectively, MIP and `Exact`.

Tables 9 and 10 display the median of each of the four ratios by problem size n and m. It shows that MAS^{H} matches or improves the known upper bound for about half of the instances with a small median deviation from the lower bounds; thus, MAS^{H} reduces the optimality gap reported in [18]. When $\rho=0.2$, $\underline{z}^{\kappa}$ and UB$^{\kappa}$ coincide with the optimality gap $\underline{z}^{\kappa}-$LB$^{\kappa}$ being zero in most instances; suggesting the optimality of the obtained solutions for those cases. MAS^{H} is faster than the heuristic of [18] with a speed up factor ranging between 2.61 and 40.58 when MIP is used and between 4.45 and 206.38 when `Exact` is applied. The speed up factors are largest for n=30 and m=6.

Table 9
MAS^{H} versus the heuristic of Keddad-Sidhom et al. [18] as a function of the number of jobs.

n	$\frac{UB^{\kappa}}{\underline{z}^{\kappa}}$	$\frac{\underline{z}^{\kappa}}{LB^{\kappa}}$	$\frac{t_{KSS}}{\underline{t}^{\kappa}_{MIP}}$	$\frac{t_{KSS}}{\underline{t}^{\kappa}_{\texttt{Exact}}}$
30	1.00	1.01	6.49	43.90
60	0.99	1.02	9.41	24.78
90	1.00	1.02	11.73	19.71

Table 10
MAS^{H} versus the heuristic of Keddad-Sidhom et al. [18] as a function of the number of machines.

m	$\frac{UB^{\kappa}}{\underline{z}^{\kappa}}$	$\frac{\underline{z}^{\kappa}}{LB^{\kappa}}$	$\frac{t_{KSS}}{\underline{t}^{\kappa}_{MIP}}$	$\frac{t_{KSS}}{\underline{t}^{\kappa}_{\texttt{Exact}}}$
2	1.00	1.01	10.02	16.73
4	1.00	1.02	8.51	27.95
6	0.99	1.02	8.85	46.16

6. Conclusion

This paper addresses the minimum weighted earliness tardiness parallel machine scheduling problem. It proposes a precedence based mixed integer linear programming model, and approximately solves it using a decentralized multi-agent system. MAS^{H} decomposes the problem into smaller components mimicking the decisional process of a decision maker. It avoids dealing with all the jobs simultaneously unless absolutely necessary, and iteratively handles subsets of bottleneck jobs. It uses a local search to swap pairs of jobs among clusters, and incorporates exact approaches that simultaneously insert a new job optimally within a specific cluster on a given machine and determine the optimal starting times of the jobs of the machine. Finally, it applies a steepest descent heuristic on its constructed solution to swap a pair of jobs between two machines. The results provide computational proof of the efficiency of MAS^{H}. Its computational speed and performance make it a good solution approach for stochastic scheduling problems. MAS^{H} can be extended to multiple objective online scheduling problems encountered in real life manufacturing settings, where the scheduling aspect is a subproblem or optimizing earliness and tardiness is a second objective. In addition, MAS^{H} can handle special events that occur in these environments such as machine breakdown, random processing times, and priority scheduling.

Acknowledgements

The authors thank Dr. S. Kedad-Sidhoum and Dr. F. Sourd for providing them with the instances of [18], and the Reviewers for their constructive comments.

Table A1
Constructing a partial solution.

Initialization
1. Set $k=1$
2. Compute the peak function value ϕ_f^1 for each job $f \in N$
3. Select the job f_1^* : $\phi_{f_1^*}^1 = \max_{f \in N}\{\phi_f^1\}$ as the center of the first cluster
4. Define machine i' : $p_{i'f_1^*} = \min_{i=1,\ldots,m}\{p_{if_1^*}\}$ and create a new G-agent $G[i'][f_1^*]$
5. Let $G[i'][f_1^*]$ assign itself to i', create set $G_{i'} = \{G[i'][f_1^*]\}$, and set its group $J_{G[i'][f_1^*]} = \{f_1^*\}$
6. Let f_1^* mark itself "busy" and leave N

Iterative Step
7. WHILE (*true*) DO
8. Recompute the peak function value ϕ_f^{k+1} of each job $f \in N$
9. Select the job f_{k+1}^* : $\phi_{f_{k+1}^*}^{k+1} = \max_{f \in N}\{\phi_f^{k+1}\}$
10. IF ($\phi_{f_{k+1}^*}^{k+1} \geq 1$) THEN
11. Determine the set $\mathfrak{M}_{f_{k+1}^*}$ of machines that can schedule f_{k+1}^* with zero WET.
12. IF ($\mathfrak{M}_{f_{k+1}^*} \neq \varnothing$) THEN
13. Define machine i' : $p_{i'f_{k+1}^*} = \min_{i \in \mathfrak{M}_{f_{k+1}^*}}\{p_{if_{k+1}^*}\}$ and create a new G-agent $G[i'][f_{k+1}^*]$
14. Let $G[i'][f_{k+1}^*]$ assign itself to i', enter the set $G_{i'} = G_{i'} \cup \{G[i'][f_{k+1}^*]\}$, and set its group $J_{G[i'][f_{k+1}^*]} = \{f_{k+1}^*\}$
15. Let f_{k+1}^* mark itself "busy" and leave N
16. END IF
17. Set $k = k+1$
18. ELSE
19. BREAK
20. END IF
21. END WHILE

Table A2
Completing the solution's construction.

1. Set *Redirect*=*false*
2. WHILE ($N \neq \varnothing$) DO
3. Set $i=1$ and set the boolean indicator variable *Flag*=*false*
4. WHILE ($i \leq m$) DO
5. Set $j=1$
6. WHILE ($j \leq |G_i|$) DO
7. Initialize $a=0$ and $b=0$
8. Let $G[i][j]$ run its group-formation action, and set the boolean variable $F_1 = G[i][j].\texttt{GroupFormation}(a, b)$
9. IF (F_1) THEN
10. Merge G-agents on machine i
11. Apply a steepest descent search
12. END IF
13. Update the variable *Flag* setting *Flag*=(*Flag* OR F_1)
14. WHILE ((*Redirect*) AND ($a \geq 1$) AND ($b \geq 1$)) DO
15. IF ($b \leq |G_a|$) THEN
16. set $c=a$
17. Let $G[a][b]$ run its group-formation action, and set the boolean variable $F_2 = G[a][b].\texttt{GroupFormation}(a, b)$
18. IF (F_2) THEN
19. Merge G-agents on machine c
20. Apply a steepest descent search
21. END IF
22. ELSE
23. BREAK
24. END IF
25. END WHILE
26. Increment the counter j setting $j=j+1$
27. END WHILE
28. Increment the counter i setting $i=i+1$
29. END WHILE
30. IF (NOT Flag) THEN
31. set $\ell = 1$
32. WHILE ($\ell \leq |N|$) DO
33. IF (I-agent ℓ is free) THEN
34. Choose machine i' such that $p_{i'\ell} = \min_{i=1,\ldots,m} p_{il}$ and create a new G-agent $G[i'][\ell]$
35. Let $G[i'][\ell]$ assign itself to i', declare $G_{i'} = G_{i'} \cup \{G[i'][\ell]\}$, and $J_{G[i'][\ell]} = \{\ell\}$
36. Let ℓ mark itself busy and leave N.
37. BREAK
38. END IF
39. Increment the counter ℓ setting $\ell = \ell+1$
40. END WHILE
41. END IF
42. END WHILE

Table A3
Group formation action: boolean `GroupFormation`(a, b).

1. Set the variables $a = 0,\ b = 0$
2. Define $V_{G[i][j]} = \{f \in N :\ [d_f - p_{if}, d_f] \cap [S_{G[i][j]}, C_{G[i][j]}] \neq \emptyset\}$
3. Sort the I-agents of $V_{G[i][j]}$ in non-increasing order of the size of their conflict with $G[i][j]$
4. Set $f = 1$.
5. WHILE ($f \leq |V_{G[i][j]}|$) DO
6. Invite f to join $J_{G[i][j]}$: activate a group-joining action for f, set the boolean variable $Flag = f$.`GroupJoining`(a, b)
7. IF ($Flag$) THEN
8. Insert f in position y of group $J_{G[i][j]}$
9. Ask the M-agent to update the schedule of i according to the optimal solution received from f
10. Let f change its state to "busy", and leave N setting $N = N \backslash \{f\}$
11. Apply `LocalSearch`$(G[i][j])$
12. RETURN *true*
13. END IF
14. Increment f setting $f = f + 1$
15. END WHILE
16. RETURN *false*

Table A4
Group joining action: boolean `GroupJoining`(a,b).

1. IF (f has never received an attachment offer) THEN
2. Determine the set $\mathfrak{M}_f$ of machines other than i that can schedule f with zero WET and without overlapping any previously scheduled job;
3. IF ($\mathfrak{M}_f \neq \emptyset$) THEN
4. Choose the machine $i' \in \mathfrak{M}_f$ that processes f fastest: $p_{i'f} = \min_{i \in \mathfrak{M}_f}\{p_{if}\}$, and ask the M-agent to create a new G-agent $G[i'][f]$
5. Let the G-agent $G[i'][f]$ assign itself to i', enter the set $G_{i'}$, and add f to its group $J_{G[i'][f]}$
6. Mark f "busy", and leave N
7. RETURN *false*
8. END IF
9. END IF
10. IF (($\mathrm{INC}(f, ST_f, G[i^*][j^*]) \geq 0$) AND ($ST_f = ST_{G[i^*][j^*]}$)) THEN
11. IF ($G[i^*][j^*] \neq G[i][j]$) THEN
12. Calculate $\mathrm{WET}(f, G[i][j]) =$ `ComputeMinimalWET`$(G[i][j], \mathrm{WET}(N_i) + \mathrm{INC}(f, ST_f, G[i^*][j^*]))$
13. ELSE
14. Set $\mathrm{WET}(f, G[i][j]) = \mathrm{WET}(N_i) + \mathrm{INC}(f, ST_f, G[i^*][j^*])$
15. END IF
16. ELSE
17. Calculate $\mathrm{WET}(f, G[i][j]) =$ `ComputeMinimalWET`$(G[i][j], \infty)$
18. END IF
19. IF ($\mathrm{WET}(f, G[i][j]) \geq 0$) THEN
20. Set $\mathrm{INC}(f, ST_f, G[i][j]) = \mathrm{WET}(f, G[i][j]) - \mathrm{WET}(N_i)$
21. ELSE
22. RETURN *false*
23. END IF
24. Set $i' = 1$
25. WHILE ($i' \leq m$) DO
26. Set $j' = 1$;
27. WHILE ($j' \leq |G_{j'}|$) DO
28. IF ($G[i'][j'] \neq G[i][j]$) THEN
29. Send to $G[i'][j']$ a request for possible attachment
30. IF ($G[i'][j']$ agrees; i.e., if $[S_{G[i'][j']}, C_{G[i'][j']}] \cap [d_f - p_{i'f}, d_f] \neq \emptyset$) THEN
31. Assess $\mathrm{WET}(f, G[i'][j']) =$ `ComputeMinimalWET`$(G[i'][j'], \mathrm{WET}(N_{i'}) + \mathrm{INC}(f, ST_f, G[i][j]))$
32. IF ($\mathrm{WET}(f, G[i'][j']) \geq 0$) THEN
33. Set the state of f $ST_f = ST_{G[i'][j']}$
34. Set $\mathrm{INC}(f, ST_f, G[i'][j']) = \mathrm{WET}(f, G[i'][j']) - \mathrm{WET}(N_{i'})$
35. Set $a = i'$ and $b = j'$
36. RETURN *false*
37. END IF
38. END IF
39. END IF
40. Set $j' = j' + 1$
41. END WHILE
42. Set $i' = i' + 1$
43. END WHILE
44. RETURN *true*

Table A5
Solution's enhancement: `JobExchanging`($G[i][j], G[i'][j']$).

1.	Set $\omega_{ii'} = \mathrm{WET}(N_i) + \mathrm{WET}(N_{i'})$ and set the real parameter $\xi \in [0.5, 2]$.
2.	Set $k = 0$
3.	WHILE ($k \leq \lvert J_{G[i][j]} \rvert$) DO
4.	Set $\ell = 0$
5.	WHILE ($\ell \leq \lvert J_{G[i'][j']} \rvert$) DO
6.	Set the indicator variable $Flag = false$
7.	IF (($k > 0$) AND ($l > 0$)) THEN
8.	Set $Flag$ = NOT (($d_\ell + \xi p_{i'\ell} \leq d_k - (1+\xi)p_{ik}$) OR ($d_k + \xi p_{ik} \leq d_\ell - (1+\xi)p_{i'\ell}$))
9.	ELSE Set $Flag = true$
10.	IF ((($k > 0$) OR ($\ell > 0$)) AND ($Flag$)) THEN
11.	Set $J^k_{G[i][j]} = J_{G[i][j]}$
12.	IF (($k > 0$) AND ($\ell > 0$)) THEN Change $j_k \in J^k_{G[i][j]}$ by $j_\ell \in J_{G[i'][j']}$
13.	IF (($k = 0$) AND ($\ell > 0$)) THEN Insert $j_\ell \in J_{G[i'][j']}$ into $J^k_{G[i][j]}$ immediately after job j_p where $\lvert d_{j_\ell} - d_{j_p} \rvert = \min_{j_y \in J_{G[i][j]}} \lvert d_{j_\ell} - d_{j_y} \rvert$
14.	IF (($k > 0$) AND ($\ell = 0$)) THEN Exclude job j_k from $J^k_{G[i][j]}$
15.	Time $J^k_{G[i][j]}$ on machine i by call `Timing`($J^k_{G[i][j]}$)
16.	IF ($\ell > 0$) THEN WHILE (`Repositioning`($J^k_{G[i][j]}, j_\ell$)) DO `Timing`($J^k_{G[i][j]}$)
17.	Set $N_i^k = \{N_i \backslash J_{G[i][j]}\} \cup J^k_{G[i][j]}$
18.	Run `Timing`(N_i^k) to compute the optimal completion times
19.	IF ($\omega ii' \leq \mathrm{WET}(N_i^k)$) THEN BREAK
20.	Set $J^\ell_{G[i'][j']} = J_{G[i'][j']}$
21.	IF (($k > 0$) AND ($\ell > 0$)) THEN Change $j_\ell \in J^\ell_{G[i'][j']}$ by $j_k \in J_{G[i][j]}$
22.	IF (($k > 0$) AND ($\ell = 0$)) THEN Insert $j_k \in J_{G[i][j]}$ into $J^\ell_{G[i'][j']}$ immediately after job j_γ where $\lvert d_{j_k} - d_{j_\gamma} \rvert = \min_{j_y \in J_{G[i'][j']}} \lvert d_{j_k} - d_{j_y} \rvert$
23.	IF (($k = 0$) AND ($\ell > 0$)) THEN Exclude job j_ℓ from $J^\ell_{G[i'][j']}$
24.	Time $J^\ell_{G[i'][j']}$ on machine i' by call `Timing`($J^\ell_{G[i'][j']}$)
25.	IF ($k > 0$) THEN WHILE (`Repositioning`($J^\ell_{G[i'][j']}, j_k$)) DO `Timing`($J^\ell_{G[i'][j']}$).
26.	Set $N_{i'}^\ell = \{N_{i'} \backslash J_{G[i'][j']}\} \cup J^\ell_{G[i'][j']}$
27.	Run `Timing`($N_{i'}^\ell$) to compute the optimal completion times
29.	IF ($\omega ii' \leq \mathrm{WET}(N_i^k) + \mathrm{WET}(N_{i'}^\ell)$) THEN BREAK
30.	Update the schedule of machine i according to N_i^k
31.	Update the schedule of machine i' according to $N_{i'}^\ell$
32.	Set $\omega_{ii'} = \mathrm{WET}(N_i^k) + \mathrm{WET}(N_{i'}^\ell)$
33.	END IF
33.	$l = l + 1$
35.	END WHILE
36.	$k = k + 1$
37.	END WHILE

Appendix

The Appendix provides the pseudocode of the procedures used by MAS^{H}.

Table A1 explains the initialization process. Unless otherwise specified, the tasks are undertaken by the M-agent.

Table A2 details the completion of the solution's construction. Unless otherwise specified, the tasks are undertaken by the M-agent. The variables a and b are passed by reference to the `GroupFormation`(a,b) function. Thus, the value of either argument a or b can be modified by the called function.

Table A3 describes the `GroupFormation`(a,b) function. Unless otherwise specified, the tasks are undertaken by a G-agent. In turn, the variables a and b are passed by reference to the `GroupJoining`(a,b) function.

Table A4 gives the pseudocode of `GroupJoining`(a,b). Unless otherwise specified, each action is carried out by the I-agent.

Table A5 details the solution's enhancement procedure `JobExchanging`($G[i][j], G[i'][j']$).

References

[1] Agnetis A, Mirchandani PB, Pacciarelli D, Pacifici A. Scheduling problems with two competing agents. Oper Res 2004;52(2):229–42.

[2] Alidaee B, Panwalkar SS. Single stage minimum absolute lateness problem with a common due date on non identical machines. J Oper Res Soc 1993;44:29–36.

[3] Bank J, Werner F. Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties. Math Comput Modelling 2001;33:363–83.

[4] Barbati M, Bruno G, Genovese A. Applications of agent-based models for optimization problems: a literature review. Expert Syst Appl 2012;39:6020–8.

[5] Biskup D, Cheng TCE. Multiple machine scheduling with earliness, tardiness and completion time penalties. Comput Oper Res 1999;26:45–57.

[6] Bongaerts L, Monostori L, McFarlane D, Kadar B. Hierarchy in distributed shop floor control. Comput Ind 2000;43:123–37.

[7] Chen ZL, Powell WB. A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. Department of Civil Engineering & Operations Research 1997, Princeton University, Princeton, USA, October 1997.

[8] Chun A, Wai H, Wong RYM. Optimizing agent-based meeting scheduling through preference estimation. Eng Appl Artif Intell 2003;16:727–43.

[9] De P, Ghosh JB, Wells CE. On the general solution for a class of early/tardy problems. Comput Oper Res 1993;20:141–9.

[10] Drobouchevitch IG, Jeffrey BS. Minimization of earliness, tardiness and due date penalties on uniform parallel machines with identical jobs. Comput Oper Res 2012;39:1919–26.

[11] Federgruen A, Mosheiov G. Heuristics for multimachine scheduling problems with earliness and tardiness costs. Manage Sci 1996;42:1544–55.

[12] FIPA Interaction Protocol Specification, ⟨http://www.fipa.org/⟩last accessed July 2012.

[13] Gerstl E, Mosheiov G. Scheduling job classes on uniform machines. Comput Oper Res 2012;39:1927–32.

[14] Gordon V, Proth JM, Chu C. A survey of the state-of-the-art of common due date assignment and scheduling research. Eur J Oper Res 2002;139:1–25.

[15] Hendel Y, Sourd F. An improved earliness–tardiness timing algorithm. Comput Oper Res 2007;34:2931–8.
[16] Jennings NR, Wooldridge MJ. Applications of intelligent agents. In: Jennings NR, Wooldridge MJ, editors. Agent technology: foundations, applications and markets. Agent technology: foundations, applications and markets. Springer; 1998. p. 3–28.
[17] Jennings NR, Wooldridge MJ. Applying agent technology. Int J Appl Artif Intell 1995;9:351–69.
[18] Kedad-Sidhoum S, Solis RY, Sourd F. Lower bounds for the earliness–tardiness scheduling problem on parallel machines with distinct due dates. Eur J Oper Res 2008;189:1305–16.
[19] Kubiak W, Lou S, Sethi R. Equivalence of mean flow time problems and mean absolute deviation problems. Oper Res Lett 1990;9:371–4.
[20] Lauff V, Werner F. Scheduling with common due date, earliness and tardiness penalties for multi-machine problems: a survey. Math Comput Modelling 2004;40(5–6):637–55.
[21] Li DC, Hsu PH. Solving a two-agent single-machine scheduling problem considering learning effect. Comput Oper Res 2012;39:1644–51.
[22] M'Hallah R, Al-Khamis T. Minimising total weighted earliness and tardiness on parallel machines using a hybrid heuristic. Int J Prod Res 2012;50:2639–64.
[23] Mason SJ, Jin S, Jampani J. A moving block heuristic to minimise earliness and tardiness costs on parallel machines. Int J Prod Res 2009;47:5377–90.
[24] Mosheiov G, Sarig A. Due-date assignment on uniform machines. Eur J Oper Res 2009;193:49–58.
[25] Parunak HVD. Agents in overalls: experiences and issues in the development and deployment of industrial agent-based systems. Int J Coop Inf Syst 2000;9:209–28.
[26] Polyakovsky S, M'Hallah R. An agent-based approach to the two-dimensional guillotine bin packing problem. Eur J Oper Res 2009;192(3):767–81.
[27] Polykovskiy S, M'Hallah R. An intelligent framework to online bin packing in a just-in-time environment. In: Mehrotra KG, Mohan CK, Oh JC, Varshney PK, editors, Modern approaches in applied intelligence, part II, Berlin, Heidelberg; 2011. p. 226–36.
[28] Shen W, Wang L, Hao Q. Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. IEEE Trans Syst Man Cybern Part C: Appl Rev 2006;36(4):563–77.
[29] Solis YR, Sourd F. Exponential neighborhood search for a parallel machine scheduling problem. Comput Oper Res 2008;35:1697–712.
[30] Sun H, Wang G. Parallel machine earliness and tardiness scheduling with proportional weights. Comput Oper Res 2003;30:801–8.
[31] Toksari MD, Güner E. Parallel machine earliness/tardiness scheduling problem under the effects of position based learning and linear/nonlinear deterioration. Comput Oper Res 2009;36:2394–417.
[32] Wooldridge MJ, Jennings NR. Agent theories, architectures and languages: a survey. In: ECAI94 workshop on agent theories architectures and languages, Amsterdam, Netherlands; 1994. p. 1–32.
[33] Yager R, Filev D. Essentials of fuzzy modeling and control. New York: John Wiley; 1984.