Noname manuscript No.
(will be inserted by the editor)

Benders decomposition for the mixed no-idle permutation flowshop scheduling problem

Tolga Bektaş · Alper Hamzadayı · Rubén Ruiz

Received: date / Accepted: date

Abstract The mixed no-idle flowshop scheduling problem arises in the modern industry including integrated circuits, ceramic frit and steel production, among others, and where some machines are not allowed to remain idle between jobs. This paper describes an exact algorithm that uses Benders decomposition with a simple yet effective enhancement mechanism that entails the generation of additional cuts by using a referenced local search to help speed up convergence. Using only a single additional optimality cut at each iteration, and combined with combinatorial cuts, the algorithm can optimally solve instances with up to 500 jobs and 15 machines, that are otherwise not within the reach of off-the-shelf optimization software and easily surpass ad-hoc existing metaheuristics. To the best of the authors' knowledge, the algorithm described here is the only exact method for solving the mixed no-idle permutation flowshop scheduling problem.

Keywords Flowshop scheduling · Mixed no-idle · Benders decomposition · Referenced local search

T. Bektaş
University of Liverpool Management School
University of Liverpool
Liverpool, UK, L69 7ZH
E-mail: t.bektas@liverpool.ac.uk

A. Hamzadayı
Van Yuzuncu Yil University
Department of Industrial Engineering
65080 Van, Turkey
E-mail: alperhamzadayi@yyu.edu.tr

R. Ruiz
Grupo de Sistemas de Optimización Aplicada
Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación,
Edifico 8G, Acc. B. Universitat Politècnica de València,
Camino de Vera s/n, 46021, València, Spain
E-mail: rruiz@eio.upv.es

1 Introduction

The permutation flow shop scheduling problem (PFSP) is concerned with sequencing a set N of n jobs on a set M of m machines in a sequential manner. Each job j has a known, fixed and a non-negative amount of processing time on machine i denoted by p_{ij} $(i = 1, \ldots, m;\ j = 1, \ldots, n)$. At any point in time, each job can be processed by at most one machine and each machine can process at most one job. When a machine starts processing a job, it must complete that job without interruption as no preemption is allowed. The sequence of jobs to be processed is the same for each machine, implying that there are $n!$ possible solutions of the problem. One extension of the PFSP is the no-idle permutation flow shop scheduling problem (NPFSP), where machines should run continuously from the time that they start the first job until they complete the last job, i.e., idle times are not allowed at machine in between the processing of consecutive jobs. The problem arises in production environments where setup times or machine operating costs are significant such that it is not cost-effective to let the machines sit idle at any point during the production run (Pan and Ruiz, 2014), such as foundry production (Saadani et al., 2003), fiber glass processing (Kalczynski and Kamburowski, 2005), production of integrated circuits and steel industry (Pan and Ruiz, 2014). In other real life cases, and as pointed out by Ruiz et al. (2009), there might be technological constraints impeding idleness at machines, like high temperature frit kilns, for example.

The first study on the NPFSP is by Adiri and Pohoryles (1982), defined on two-machines with the objective of minimizing the sum of completion times. A more popular objective has been to minimize the total makespan, namely the sum of the completion times of the last job processed on each machine, for which the first study is due to Vachajitpan (1982), where mathematical models and branch-and-bound methods for solving small-scale instances were presented. One exact method using branch-and-bound was described by Baptiste and Hguny (1997). Comprehensive reviews on the problem can be found in Ruiz and Maroto (2005) and Goncharov and Sevastyanov (2009), which indicate an abundance of heuristic algorithms described to solve the problem, including discrete differential evolution and particle swarm optimization (Pan and Wang, 2008a,b), iterated greedy search (Ruiz et al., 2009), variable neighborhood search (Tasgetiren et al., 2013) and memetic algorithms (Shao et al., 2017). To date, however, no effective exact approach has been proposed for NPFSP and those that exist can solve instances with only more than a handful of jobs (Pan and Ruiz, 2014).

The mixed no-idle permutation flow shop scheduling problem (MNPFSP) arises as a more general case of the NPFSP when some machines are allowed to be idle, and others not. Examples can be found in the production of integrated circuits and ceramic frit, as well as in the steel industry (Pan and Ruiz, 2014). In the ceramic frit production, for example, only the central fusing kiln has the no-idle constraint. As an extension of the PFSP, which is known to be NP-Hard for three or more machines (see, e.g., Röck, 1984), the MNPFSP is also NP-

Hard in the strong sense (Pan and Ruiz, 2014). The only study on this problem that minimizes makespan is that of Pan and Ruiz (2014), which describes a mixed integer programming model for the problem, an effective iterated greedy (IG) algorithm and enhancements to accelerate the calculation of insertions used within the local search. Computational results showed that in comparison to the existing methods, the IG algorithm was able to identify solutions for the NPFSP instances that were 61% better on average, with respect to the makespan. However, no exact method, to our knowledge, has been proposed to solve the MNPFSP, which is the aim of this paper. In particular, we contribute to the literature by describing an application of Benders decomposition that is enhanced with a referenced local search (RLS), that is used to generate additional cuts to accelerate the convergence of the algorithm. We propose and test three cut generating strategies, and use combinatorial cuts to discard solutions already evaluated. The algorithms described in this paper are all exact, the performance of which we computationally assess on a test bed of literature instances, and compare with the commercial optimizer CPLEX and its automated Benders decomposition algorithm.

The remainder of this paper is structured as follows. Section 2 formally defines the problem and presents a formulation. The proposed algorithm is described in detail in Section 3. Computational results are presented in Section 4, followed by conclusions and future research in Section 5.

2 The Mixed No-Idle Permutation Flowshop Scheduling Problem

We denote by $\pi_{(j)}$ the job occupying position j in a given permutation π. The PFSP requires the condition $c_{i,\pi_{(j)}} \geq c_{i,\pi_{(j-1)}} + p_{i,\pi_{(j)}}$ to hold for any two consecutive jobs in any permutation, where $c_{i,j}$ is the completion time of task j at machine i. A key difference between the NPFSP and the PFSP is that the former forbids any idle time inbetween any two consecutive tasks, thus transforming the previous inequality into an equality as $c_{i,\pi_{(j)}} = c_{i,\pi_{(j-1)}} + p_{i,\pi_{(j)}}$. The mixed no-idle problem generalizes the two problems in which there exists a subset $M' \subseteq M$ of m' 'no-idle' machines, and it is only for those machines in $M \setminus M'$ that any idle running is allowed. Apart from these key differences, the common PFSP assumptions hold (Baker, 1974): (1) Jobs are independent from each other and are available for their processing from time 0; (2) machines are always available (no breakdowns); (3) machines might only process one task at any given time; (4) jobs can only be processed by one machine at all times; (5) tasks have to be processed without interruptions once started and until their completion (no preemptions allowed); (6) setup times are either sequence-independent and can be directly included in the processing times, or are considered negligible and therefore ignored; and finally (7) there is an infinite in-process buffer capacity between any two machines in the shop.

In the remainder of the paper, we make use of the integer programming formulation below of the problem described in Pan and Ruiz (2014), provided here for the sake of completeness. In this formulation, a binary variable x_{jk}

takes the value 1 if job j is in position k of the sequence, and 0 otherwise. Continuous variables $c_{i,k}$ denote the completion time of job $k = 1, \ldots, n$ on machine $i = 1, \ldots, m$.

$$\text{Minimize} \qquad c_{m,n} \tag{1}$$

subject to

$$\sum_{k=1}^{n} x_{jk} = 1 \qquad j = 1, \ldots, n \tag{2}$$

$$\sum_{j=1}^{n} x_{jk} = 1 \qquad k = 1, \ldots, n \tag{3}$$

$$c_{1,k} \geq \sum_{j=1}^{n} p_{1j} x_{j1} \qquad k = 1, \ldots, n \tag{4}$$

$$c_{i,k} - c_{i-1,k} \geq \sum_{j=1}^{n} p_{ij} x_{jk} \qquad k = 1, \ldots, n; i = 2, \ldots, m \tag{5}$$

$$c_{i,k} - c_{i,k-1} = \sum_{j=1}^{n} p_{ij} x_{jk} \qquad k = 2, \ldots, n; i \in M' \tag{6}$$

$$c_{i,k} - c_{i,k-1} \geq \sum_{j=1}^{n} p_{ij} x_{jk} \qquad k = 2, \ldots, n; i \in M \setminus M' \tag{7}$$

$$c_{i,k} \geq 0 \qquad k = 1, \ldots, n; i = 1, \ldots, m \tag{8}$$

$$x_{jk} \in \{0, 1\} \qquad j, k = 1, \ldots, n. \tag{9}$$

Objective function (1) minimizes the makespan among all jobs. In the PFSP and also for the MNPFSP, the makespan corresponds to the completion time of the last job in the permutation at the last machine of the shop ($c_{m,n}$). With constraints (2) and (3) we force that any job occupies one position in the permutation and also that each position at any permutation is occupied by one job. Constraints (4) control the completion time of the first job in the permutation. Constraints (5) enforce that the completion times of jobs on the second and subsequent machines are larger than the completion times of the preceding tasks of the same job on previous machines plus their processing time, effectively avoiding overlaps of the tasks of the same job. The key characteristic of the MNPFSP is shown in constraints (6) and (7). Constraint (6) forbids idle time at 'no-idle' machines by making the completion time of a job *equal* completion time of the previous job in the sequence plus its processing time. Inequality (7) is the usual PFSP constraint that forbids overlaps of jobs on the same machine and at the same time allows for idle time.

3 Benders Decomposition

In this section, we first describe an application of the traditional Benders decomposition followed by the proposed cut generation strategy using a local search algorithm.

3.1 Application of Benders decomposition

Benders decomposition (Benders, 1962) is a cutting plane algorithm to solve a Benders reformulation of a given model that enables it to be decomposed into two simpler formulations, namely the master problem and the subproblem. The master problem contains only a subset of the variables and of the constraints of the original model. The subproblem is the original model in which the master problem variables are fixed, and the solution of which yields either an optimality or a feasibility cut for the master problem (Costa et al., 2012). Benders reformulation is typically solved using a delayed constraint generation algorithm that iterates between the master and the subproblem, until an optimal solution is identified.

Let $M(c,x)$ denote the formulation (1)–(9) where $x = \{x_{jk}|j,k = 1,\ldots,n\}$ and $c = \{c_{ik}|k = 1,\ldots,n; i = 1,\ldots,m\}$ are the vectors of the decision variables. Let us suppose that variables x have been fixed as $x = \hat{x} \in X = \{x|x \text{ satisfies } (2),(3),(9)\}$. The resulting formulation, shown by $M(c,\hat{x})$, consists only of the variables c, and the constraints of which are assigned the dual variables α and β corresponding to constraints (4) and (5), respectively, and γ corresponding to constraints (6) and (7), respectively. The dual $D(\alpha,\beta,\gamma,\hat{x})$ of $M(c,\hat{x})$ formulation is given by the following:

$$\text{Maximize } \sum_{j=1}^{n}\hat{x}_{j1}p_{1j}\sum_{k=1}^{n}\alpha_k + \sum_{k=1}^{n}\sum_{i=2}^{m}\beta_{ik}\sum_{j=1}^{n}\hat{x}_{jk}p_{ij} + \sum_{k=2}^{n}\sum_{i=1}^{m}\gamma_{ik}\sum_{j=1}^{n}\hat{x}_{jk}p_{ij} \quad (10)$$

subject to

$$\alpha_k - \beta_{i+1,k} - \gamma_{i,k+1} \le 0 \qquad k = i = 1 \quad (11)$$

$$\alpha_k - \beta_{i+1,k} + \gamma_{i,k} - \gamma_{i,k+1} \le 0 \qquad k = 2,\ldots,n-1; i = 1 \quad (12)$$

$$\alpha_k - \beta_{i+1,k} + \gamma_{i,k} \le 0 \qquad k = n, i = 1 \quad (13)$$

$$\beta_{i,k} - \beta_{i+1,k} - \gamma_{i,k+1} \le 0 \qquad k = 1; i = 2,\ldots,m-1 \quad (14)$$

$$\beta_{i,k} - \beta_{i+1,k} + \gamma_{i,k} - \gamma_{i,k+1} \le 0 \qquad k = 2,\ldots,n-1; i = 2,\ldots,m-1 \quad (15)$$

$$\beta_{i,k} - \beta_{i+1,k} + \gamma_{i,k} \le 0 \qquad k = n; i = 2,\ldots,m-1 \quad (16)$$

$$\beta_{i,k} - \gamma_{i,k+1} \le 0 \qquad k = 1; i = m \quad (17)$$

$$\beta_{i,k} + \gamma_{i,k} - \gamma_{i,k+1} \le 0 \qquad k = 2,\ldots,n-1; i = m \quad (18)$$

$$\beta_{i,k} + \gamma_{i,k} \le 1 \qquad k = n; i = m \quad (19)$$

$$\alpha_k \ge 0 \qquad k = 1,\ldots,n \quad (20)$$

$$\beta_{i,k} \ge 0 \qquad k = 1,\ldots,n; i = 2,\ldots,m \quad (21)$$

$$\gamma_{i,k} \ge 0 \qquad k = 2,\ldots,n; i \in M \setminus M'. \quad (22)$$

The procedure to calculate the makespan we use here is the one proposed in Pan and Ruiz (2014), which calculates the start and completion times of the jobs in the order that they appear in a given permutation, with the makespan being equal to the completion of the job in the last position. The procedure runs in $\mathcal{O}(nm)$ time, and implies that $M(c, \hat{x})$ always admits a feasible solution for a given $\hat{x} \in X$. This, in turn, means that $D(\alpha, \beta, \gamma, \hat{x})$ is always feasible for a given $\hat{x} \in X$, and for an optimal solution $(\hat{\alpha}, \hat{\beta}, \hat{\gamma})$ of the dual problem, one obtains the following Benders optimality cuts:

$$z \geq \sum_{j=1}^{n} A_j x_{j1} + \sum_{j=1}^{n}\sum_{k=2}^{n} B_{jk} x_{jk},$$

where z is a lower bound on the optimal solution value of $M(c, x)$, $A_j = \sum_{k=1}^{n} \hat{\alpha}_k p_{1j} + \sum_{i=2}^{m} \hat{\beta}_{i1} p_{ij}$ and $B_{jk} = \sum_{i=2}^{m} \hat{\beta}_{ik} p_{ij} + \sum_{i=1}^{m} \hat{\gamma}_{ik} p_{ij}$. Using this result, we are now ready to present the following reformulation of $M(c, x)$, referred to as the master problem constructed using the set P_D of extreme points of the polytope defined by the dual problem $D(\alpha, \beta, \gamma, \hat{x})$, and shown as MP($P_D$) below:

$$\text{Minimize} \qquad z \tag{23}$$

subject to

$$\sum_{k=1}^{n} x_{jk} = 1 \qquad j = 1, \ldots, n \tag{24}$$

$$\sum_{j=1}^{n} x_{jk} = 1 \qquad k = 1, \ldots, n \tag{25}$$

$$z \geq \sum_{j=1}^{n} A_j x_{j1} + \sum_{j=1}^{n}\sum_{k=2}^{n} B_{jk} x_{jk} \qquad (\alpha, \beta, \gamma) \in P_D \tag{26}$$

$$x_{jk} \in \{0, 1\} \qquad j, k = 1, \ldots, n,$$

As the MP includes a large number of optimality cuts, it can be solved using a cutting plane algorithm in practice, normally starting with MP($\varnothing$) with no optimality cuts (26), and generating the cuts on an as-needed basis. The algorithm stops after solving a certain MP(P), where $P \subseteq P_D$.

To help speed up the convergence of the algorithm, we also use the following combinatorial Benders cuts using any solution $\hat{x} \in X$ in the master problem,

$$\sum_{(j,k):\hat{x}_{jk}=1} x_{jk} \leq n - 2, \tag{27}$$

which can be used to cut solution $\hat{x}$ off from the set of feasible solutions to $M(c, x)$. In particular, after the addition of constraint (27), any solution obtained by the formulation will differ from solution $\hat{x}$ with respect to the

position of at least two jobs. This follows from the fact that changing the position of one job in a given sequence implicitly requires the change of position of another job in the sequence.

3.2 Referenced local search algorithm

Another ingredient of our algorithm is a referenced local search (RLS), proposed by Pan et al. (2008), which is initialized with a seed permutation π^{ref} obtained by a good constructive heuristic. In this paper, the reference permutation π^{ref} is taken from the master problem solution. Let $C_{\max}(\pi)$ denote the makespan of permutation π. The RLS procedure first finds the referenced job, which is determined by using the index i in RLS procedure, in the current permutation π. The referenced job is removed from π and inserted into all possible positions of π. If this operation results in a permutation π^* with a lower $C_{\max}(\pi^*)$ as compared to $C_{\max}(\pi)$, then the current permutation is replaced with π^* and the value of the counter controlling the number of runs in RLS is set to 1. Otherwise, the value of the counter is increased by one. This procedure is repeated until the value of the counter reaches the number of jobs in the problem. RLS keeps a list S with the best solutions found. The pseudo code of RLS is given in Algorithm 1. Note that the accelerated makespan calculations and speed-ups proposed in Pan and Ruiz (2014) are employed in the proposed RLS local search.

Algorithm 1 RLS(π^{ref})

1: $i \leftarrow 1$; $counter \leftarrow 0$, $\pi \leftarrow \pi^{ref}$, $S \leftarrow \pi^{ref}$
2: **while** $(counter \leq n)$ **do**
3: Locate and extract job $\pi^{ref}_{(i)}$ from π
4: Insert job $\pi^{ref}_{(i)}$ in all possible positions of π and let π^* be the permutation resulting in the best $C_{\max}$
5: **if** $(C_{\max}(\pi^*) < C_{\max}(\pi))$ **then**
6: $\pi \leftarrow \pi^*$; $counter \leftarrow 1$; update set S of best solutions
7: **else**
8: $counter \leftarrow counter + 1$
9: **end if**
10: $i \leftarrow mod(i + 1, n)$
11: **end while**
12: **return** π, S

3.3 Cut generation using referenced local search

The choice of various ingredients used within Benders decomposition can have a significant effect on the performance of the algorithm. These range from model selection (Magnanti and Wong, 1981), cut improvement (e.g. Papadakos, 2008; Saharidis and Ierapetritou, 2013) and strengthening the master problem through the addition of pre-generated valid inequalities (e.g., Cordeau et al., 2006). However, the choice of the integer solutions obtained from the master problem and the improvement thereof has received much less attention. In particular, the question around the effect of the quality of a feasible solution and the strength of the corresponding cut subsequently added to the master problem on the convergence of the Benders decomposition algorithm has not yet been fully investigated. Costa et al. (2012), for example, suggest the use of intermediate solutions within Benders decomposition, either obtained during the course of the branch-and-cut algorithm, or through local search, and report encouraging results for the fixed charge network design problem. In this paper, we seek to explore this question further and propose a Benders decomposition algorithm that embeds the bespoke RLS into the algorithm for solving the MNPFSP. The aim is to enhance the performance of the algorithm by generating extra cuts within the algorithm induced by the heuristic solutions. This section explains the details of the algorithm, and describes and tests various strategies for an effective implementation.

The enhanced Benders decomposition is an iterative algorithm that generates optimality cuts (26) at each iteration on the basis of an optimal MP solution x^*, uses x^* as an input to the RLS to generate a user-defined number σ of neighbor solutions, each of which induces an additional optimality cut inserted into the master problem. The reason behind the choice of the RLS, instead of other heuristics and meta-heuristics for the problem, is the simplicity of its implementation and the lack of any special input parameters. The RLS has been shown to work effectively for solving the PFSP (Pan et al., 2008), the NPFSP (Deng and Gu, 2012) and the MNPFSP (Pan and Ruiz, 2014). We propose three different strategies for generating the additional cuts that are described below. Let $S = \{x_1, x_2, \ldots, x_{|S|}\}$ be the set of solutions generated by RLS and let $v(x)$ denote the objective function value of a feasible solution x.

- **Elite:** Choose the first σ solutions in S such that $v(x_1) \geq v(x_2) \geq \ldots \geq v(x_\sigma) \geq v(x^*)$.
- **Highly elite:** Choose the best σ solutions in S in terms of the objective value.
- **Random:** Choose σ solutions in S at random.

The pseudo-code of the proposed algorithm is given in Algorithm 2.

Algorithm 2 Benders decomposition

Input: number σ of cuts to add at each iteration, cut generation strategy, allowable optimality gap $\epsilon \geq 0$

Set $LB \leftarrow -\infty$, $UB \leftarrow +\infty$, $P \leftarrow \varnothing$
while $LB \leq UB - \epsilon$ **do**
  Let x_0 denote the solution of MP(P).
  if $(LB < v(x_0))$ **then**
    Set $LB \leftarrow v(x_0)$
  end if
  Let $V \leftarrow x_0$
  Let S be the set of all neighbor solutions of x_0 obtained by the RLS.
  Let $V \leftarrow x_i$ where x_i is one of the $i = 1, 2, \ldots, \sigma$ solutions obtained according to the cut generation strategy used (elite, highly elite or random)
  for each solution $x_i \in V$ **do**
    Solve $D(\alpha, \beta, \gamma, x_i)$ and let $(\alpha, \beta, \gamma)^i$ denote the resulting solution
    $P \leftarrow (\alpha, \beta, \gamma)^i$
    if $(v(x_i) < UB)$ **then**
      Set $UB \leftarrow v(x_i)$
      Set $x^* \leftarrow x_i$
    end if
    Generate a combinatorial cut using solution x_0 and add to MP(P).
  end for
end while

Output: Best solution x^* and value $v(x^*)$

4 Computational Results

This section presents a computational study to assess the performance of the Benders decomposition algorithm proposed in this paper. The algorithm and its variants are coded in Visual C++, using CPLEX 12.7.1 as the solver. We have used a Intel Core i5-2450M computer with a 2.5 GHz CPU and 4 GB of memory. The tests are conducted on two sets of instances available at `http://soa.iti.es/rruiz` that were originally proposed in Ruiz et al. (2009) and extended in Pan and Ruiz (2014). The experiments are conducted in four sets, which will be explained below along with the results.

4.1 Performance of standard Benders decomposition implementations

In the first stage, we first compare our (deliberately) naive implementation of Benders decomposition (shown by BD) described in Section 3.1, the branch-and-cut algorithm of CPLEX 12.7.1 (shown by CPLEX) to solve the formulation of the problem shown in Section 2, and the automated Benders decomposition available within the software (shown by ABD). The aim is to

show the performance of standard Benders decomposition implementations on the MNPFSP. For this purpose, we generate relatively small size instances from the instance I_3_500_50_1 with 500 jobs and 50 machines available at the above website. A smaller instance with 10 jobs and 3 machines, for example, is generated by using the first 10 jobs and three machines within the larger instance. A total of 27 small-scale instances are formed with $n \in \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$ jobs and $m \in \{3, 6, 9\}$ machines. A computational time limit of 7200 CPU seconds has been imposed on the total run time of each algorithm. The results are given in Table 1, where the first three columns show the instance number as the identifier, number n of jobs and number m of machines. Then, for each of the three methods compared, we provide the value of the best integer solution (BI) identified within the time limit, the final optimality gap (GAP), in percent, the overall solution time (ST), in seconds, and the total number of iterations (NI) for ABD and BD.

As can be seen from Table 1, BD was able to solve 23 out of 27 instances to optimality, while CPLEX and ABD are able to solve all of the instances to optimality within the given time limit. In terms of the time to solve to optimality, CPLEX is the fastest in all instances except for instances 16, 22 and 23, for which ABD shows a better performance. BD is quicker than the other two for instances 4, 7 and 10. These results suggest that a standard implementation of Benders decomposition without any enhancements is highly ineffective and does not seem to provide encouraging results for the MNPFSP.

4.2 Effectiveness of the cut generation strategies

The second stage of the experiments numerically evaluates the cut generation strategies, and the effect of the number of additional cuts σ added at each iteration of the algorithm. We denote by $\sigma' = \sigma + 1$ the total number of cuts added at each iteration, where the '+1' indicates the original optimality cut from the solution of the master problem, and test $\sigma' \in \{2, 5, 10, 25, 50\}$ in the experiments, resulting in a total of 15 combinations applied to the 27 instances described above. A computational time limit of 7200 CPU seconds has also been imposed.

The results of the experiments are reported in Tables 2 and 3 for the elite and the highly elite strategies, respectively, denoted by the notation Eσ' and HEσ' which also indicates the number of additional optimality cuts introduced. We do not report the results associated with the random strategy, as it has consistently produced very poor quality solutions in all cases.

The findings indicate that the elite strategy has found the optimum solutions for all instances with all settings of σ', with the exception of instances 24 and 27 for E2, instances 18, 24, 26 and 27 for E5, and instances 18, 23 and 26 for E10. The highly elite strategy, on the other hand, has found all the optimum solutions for all values of σ', as can be seen from Table 3. Furthermore, the variant HE2 solves 19 (out of 27) instances to optimality quicker than the other two cut generation strategies. Comparing HE2 with the results

of CPLEX and ABD reported in Table 1, we observe that it yields the best performance on 11 out of the 27 instances, and is able to solve instances to optimality for which BD has failed to do so within the time limit.

4.3 Results on larger-scale instances

The third and last phase of the computational study compares the algorithms on medium and large-scale instances. The former set includes a further 30 instances, generated in the same manner as the small-size instances by choosing n to range from 50 to 500 in increments of 50, and m as either 5, 10 and 15 from within the main instance I_3_500_50_1.

Having established the superiority of HE2 over other variants of the Benders decomposition algorithm in the previous section, we now employ two further enhancements to this algorithm. The first is the addition of combinatorial cuts (27) within the algorithm, shown by HE2, and the second is a further strengthening of the optimality cuts using the Pareto-optimal cut generation scheme described by Magnanti and Wong (1981), indicated by HE2+PO. These two variants have also been tested by deactivating the combinatorial cuts, indicated using the same names but with the suffix '-CC'. It should be noted with the addition of cardinality cuts, the optimality gaps are no longer valid, reason for which we do not report them here. The tests comparing the four variants have been conducted on a limited set of instances with n chosen as either 50, 100 or 150, and m equal to 5, 10 or 15. The results are reported in Table 4.

The results in Table 4 show that the variants of the algorithm, namely HE2 and HE2+PO that use combinatorial cuts, always result in a superior performance over the cases where they are not used. However, a performance comparison between HE2 and HE2+PO is not conclusive and seems to be instance-dependent. In particular, whilst HE2 is faster for instances 28, 29, 31, 34 and 35, instances 30 and 32 are solved significantly faster with HE2+PO.

To further compare HE2 and HE2+PO along with the CPLEX solver and the ABD, we provide further results on instances with up to 500 jobs in Table 5. These results are indicative of the superior performance of the two variants of the algorithm over CPLEX and ABD, in terms of both the computational solution time and solution quality. In particular, with the exception of one instance, one of the two variants always yield the best performance. More remarkably, whilst neither CPLEX nor ABD were able to identify a feasible solution for instances ($n = 150$, $m = 10$), ($n = 300$, $m = 5$), ($n = 350$, $m = 10$), ($n = 500$, $m = 5$) and ($n = 500$, $m = 5$) within the allowed time limit, at least one of HE2 or HE2+PO solved these instances to optimality.

4.4 Comparison with a state-of-the-art heuristic

The last set of experiments reported in this section are conducted to compare the algorithms we propose in this paper with a state-of-the-art heuristic de-

scribed for the problem, namely the iterated greedy algorithm (IGA) of Pan and Ruiz (2014), in terms of the value of the solutions identified. These tests use the same set of instances shown in Table 5. In their study, Pan and Ruiz (2014) run the IGA under a time limit of $n \times (m/2) \times \rho$ milliseconds, where $\rho \in \{10, 20, 30, 60, 90\}$. In our experiments, we run the IGA five times, using the largest value for $\rho = 90$ in setting the time limit for each of the five runs. The results are shown in Table 6, where the column titled BI corresponds to the best value found for the corresponding instance, and the column titled Method indicates which of the four algorithms of Table 5 were able to identify the best value. Columns five to seven report the minimum, maximum and the average solution values for each instance, respectively, across the five runs of the IGA, and the standard deviation. For reasons of fairness, we also run the IGA once for each instance, under the same time limit of 7200 seconds used for the exact algorithms, and report the value of the best solution identified in the last column of the table.

As the results in Table 6 indicate, the exact algorithms proposed in this paper, in particular HE2 and HE2+PO, are highly competitive with the IGA for instances with $n \leq 250$, and are able to identify the same solution values in most cases. However, the exact algorithms often yield better quality solutions for larger instances in comparison, irrespective of whether the IGA is run under a 7200 second time limit or shorter. In particular, for instances with ($n = 350$, $m = 15$), ($n = 400$, $m = 10$), ($n = 400$, $m = 15$), ($n = 450$, $m = 10$), ($n = 450$, $m = 15$), ($n = 500$, $m = 10$) and ($n = 500$, $m = 15$), the exact algorithms yield better quality solutions under the same time limit.

5 Conclusions and future research

This paper presented an enhancement to the traditional Benders decomposition by generating cuts using a local search algorithm for the mixed no-idle permutation flowshop scheduling problem. The latter algorithm was used as an 'oracle' to generate high-quality solutions, which in turn were used to construct additional cuts used within the iterative algorithm. Our findings indicate that such a strategy can substantially improve the performance of the algorithm, even with only a single additional cut added at each iteration, and that the quality of the solution used to generate the additional cut is of paramount importance. In particular, it is only high-quality solutions that help to improve the convergence and that randomly generated solutions worsen the efficiency of the algorithm. Our algorithm also makes use of combinatorial cuts that eliminate feasible solutions from the search space, which leads to the solution of instances of up to 500 jobs and five machines that otherwise are not solved with a commercial solver. The results encourage the use of such a strategy on other types of problems, assuming that an 'oracle' is available to generate high-quality solutions in short computational times.

Table 1 Computational results for the small instances

no.	n	m	CPLEX			ABD				BD			
			BI	ST	GAP	BI	ST	GAP	NI	BI	ST	GAP	NI
1	10	3	526	0.03	0.00	526	0.17	0.00	1	526	0.25	0.00	8
2	10	6	886	0.06	0.00	886	0.34	0.00	19	886	3.08	0.00	39
3	10	9	1210	0.30	0.00	1210	1.53	0.00	85	1210	265.24	0.00	164
4	15	3	820	0.03	0.00	820	0.19	0.00	1	820	0.10	0.00	4
5	15	6	1181	0.30	0.00	1181	3.11	0.00	65	1181	137.69	0.00	239
6	15	9	1514	0.32	0.00	1514	16.70	0.00	304	1525	7200.00	2.36	268
7	20	3	1166	0.05	0.00	1166	0.27	0.00	1	1166	0.12	0.00	4
8	20	6	1717	0.19	0.00	1717	0.28	0.00	4	1717	5.01	0.00	48
9	20	9	2122	0.27	0.00	2122	0.58	0.00	34	2122	2.87	0.00	35
10	25	3	1447	0.08	0.00	1447	0.22	0.00	3	1447	0.10	0.00	3
11	25	6	2134	0.28	0.00	2134	0.58	0.00	21	2134	3.24	0.00	24
12	25	9	2638	0.31	0.00	2638	1.97	0.00	26	2638	33.49	0.00	114
13	30	3	1748	0.12	0.00	1748	0.55	0.00	1	1748	1.00	0.00	11
14	30	6	2511	0.23	0.00	2511	0.70	0.00	4	2511	7.47	0.00	10
15	30	9	2904	0.36	0.00	2904	2.94	0.00	45	2904	1145.57	0.00	477
16	35	3	1933	0.25	0.00	1933	0.17	0.00	1	1933	1.07	0.00	11
17	35	6	2685	1.17	0.00	2685	4.28	0.00	45	2685	162.37	0.00	186
18	35	9	3047	1.63	0.00	3047	1.75	0.00	12	3047	3198.76	0.00	569
19	40	3	2161	0.27	0.00	2161	0.28	0.00	4	2161	1.16	0.00	7
20	40	6	2980	0.64	0.00	2980	1.20	0.00	12	2980	698.17	0.00	41
21	40	9	3256	11.49	0.00	3256	15.98	0.00	106	3426	7200.00	24.93	13
22	45	3	2500	0.28	0.00	2500	0.25	0.00	2	2500	1.11	0.00	7
23	45	6	3309	2.80	0.00	3309	1.64	0.00	45	3309	33.82	0.00	66
24	45	9	3660	8.92	0.00	3660	9.38	0.00	61	3717	7200.00	15.93	15
25	50	3	2730	0.44	0.00	2730	0.97	0.00	3	2730	3.27	0.00	7
26	50	6	3642	2.25	0.00	3642	2.61	0.00	37	3642	744.45	0.00	15
27	50	9	4011	4.91	0.00	4011	11.84	0.00	54	4133	7200.00	24.86	10

Table 2 Computational results of the elite strategy

no.	$\sigma' = 2$		$\sigma' = 5$		$\sigma' = 10$		$\sigma' = 25$		$\sigma' = 50$	
	ST	**NI**	**ST**	**NI**	**ST**	**NI**	**ST**	**NI**	**ST**	**NI**
1	0.28	6	0.46	11	0.67	3	0.56	3	0.26	3
2	1.75	20	1.76	12	2.96	11	1.14	6	0.91	6
3	132.94	97	120.41	68	152.09	65	98.99	61	96.17	61
4	0.12	3	0.38	3	0.8	3	0.28	2	0.37	2
5	182.4	123	61.2	57	34.42	33	36.66	35	8.77	18
6	815.26	115	624.12	85	517.62	63	484.12	54	452.19	45
7	0.16	3	0.08	2	1.19	2	0.1	2	0.18	2
8	0.54	6	0.62	5	1.7	4	0.54	2	0.49	2
9	1.21	12	3.01	15	2.33	6	0.67	2	0.65	2
10	0.65	6	0.35	3	0.87	3	0.37	2	0.29	2
11	1.08	7	0.81	4	1.09	2	0.9	2	0.84	2
12	74.43	65	24.24	38	10.59	17	2.87	6	2.75	6
13	0.74	7	0.37	3	1.14	3	0.75	2	0.67	2
14	7.17	11	0.51	3	1.41	3	1.42	3	1.22	2
15	476.86	116	442.93	132	46.19	28	8.64	9	8.26	8
16	1.49	10	1.59	7	1.33	3	0.84	2	0.72	2
17	430.9	86	870.89	31	310.29	9	7.95	9	2.28	3
18	3342.12	227	7200	8	7200	5	3.64	4	2.83	3
19	2.12	13	1.59	6	0.91	3	1.29	3	1.47	2
20	2549.29	12	235.16	59	782.15	7	2.7	3	1.99	2
21	450.32	232	362.45	172	278.31	100	251.76	64	212.76	38
22	1.55	8	0.45	3	1.68	3	1.11	2	1.04	2
23	158.22	92	3240.12	63	7200	5	178.76	4	1.65	2
24	7200	7	7200	202	3225.52	91	94.47	16	11.41	5
25	3.61	15	1.52	5	2.37	4	1.01	2	1.02	2
26	2513.16	22	7200	4	7200	3	5.96	4	2.32	2
27	7200	7	7200	5	1488.01	74	126.91	18	3.45	2

Table 3 Computational results of the highly elite strategy

no.	$\sigma' = 2$		$\sigma' = 5$		$\sigma' = 10$		$\sigma' = 25$		$\sigma' = 50$	
	ST	**NI**	**ST**	**NI**	**ST**	**NI**	**ST**	**NI**	**ST**	**NI**
1	0.29	7	0.93	6	0.32	2	0.96	2	1.34	2
2	3.05	35	4.24	22	2.11	10	2.63	6	1.51	2
3	197.89	135	171.32	85	100.09	44	56.61	23	33.2	15
4	0.06	2	0.13	2	0.22	2	0.74	2	1.24	2
5	92.6	121	181.41	61	20.7	29	18.76	14	23.66	11
6	216.13	120	245.23	90	317.12	75	314.96	66	410	44
7	0.15	2	0.15	2	0.32	2	0.87	2	1.97	2
8	0.09	2	0.33	2	0.49	2	1.08	2	2.45	2
9	0.14	2	0.25	2	0.49	2	1.19	2	2.55	2
10	0.11	2	0.22	2	0.42	2	0.99	2	1.89	2
11	0.55	5	0.31	2	0.52	2	1.36	2	2.32	2
12	1.14	11	11.22	31	2.18	6	5.03	6	8.15	5
13	0.15	2	0.46	2	0.43	2	1.17	2	2.11	2
14	0.15	2	0.33	2	0.65	2	1.31	2	2.31	2
15	1.65	11	1.8	7	12.5	18	2.48	3	2.85	2
16	0.11	2	0.3	2	0.58	2	0.87	2	1.81	2
17	0.71	4	1.51	6	7.7	12	3.67	4	2.61	2
18	2.48	13	1.38	5	5.71	10	1.05	2	5	3
19	0.32	2	0.37	2	0.64	2	1.36	2	2.49	2
20	0.14	2	0.64	2	0.86	2	1.55	2	2.63	2
21	130.25	109	162.56	86	178.12	63	250.56	51	312.76	38
22	0.23	2	0.4	2	0.66	2	1.28	2	2.33	2
23	0.21	2	0.83	2	0.79	2	2.08	2	2.86	2
24	15.59	32	27.32	24	9.29	10	5.95	4	160.62	13
25	0.21	2	0.44	2	0.86	2	1.46	2	2.47	2
26	0.2	2	0.66	2	1.08	2	1.8	2	3.44	2
27	0.3	2	0.64	2	1.07	2	2.55	2	3.52	2

Table 4 Testing variants of HE2

no.	n	m	**HE2**			**HE2-CC**				**HE2+PO-CC**				**HE2+PO**		
			BI	**ST**	**NI**	**BI**	**ST**	**GAP**	**NI**	**BI**	**ST**	**GAP**	**NI**	**BI**	**ST**	**NI**
28	50	5	3083	**0.32**	2	3083	1.84	0	2	3083	2.86	0	2	3083	1.07	2
29	50	10	4050	**1.01**	2	4050	2.11	0	6	4050	9.04	0	4	4050	3.25	4
30	50	15	5053	1917.41	209	5053	7200	0.28	226	5053	7200	0.14	272	5053	**307.09**	210
31	100	5	5630	**1.09**	2	5630	1.47	0	2	5630	3.24	0	2	5630	6.97	2
32	100	10	6855	1186.13	128	6855	1194.39	0	128	6855	1187.89	0	128	6855	**611.90**	43
33	100	15	7933	7200	66	7956	7200	1.02	67	7963	7200	1.05	66	7933	7200	68
34	150	5	8335	**2.34**	2	8335	3.01	0	2	8335	7.13	0	2	8335	6.09	2
35	150	10	9502	**56.62**	8	9502	284.43	0	18	9502	814.86	0	23	9502	1918.77	33
36	150	15	10626	7200	14	10649	7200	3.04	19	10635	7200	1.65	26	10621	7200	26

Table 5 Comparisons on larger-scale instances

n	m	**CPLEX**			**ABD**				**HE2**			**HE2+PO**		
		BI	**ST**	**GAP**	**BI**	**ST**	**GAP**	**NI**	**BI**	**ST**	**NI**	**BI**	**ST**	**NI**
50	5	3083	1.47	0	3083	1.83	0	17	3083	**0.32**	2	3083	1.07	2
50	10	4050	13.09	0	4050	21.14	0	94	4050	**1.01**	2	4050	3.25	4
50	15	5053	**259.91**	0	5053	762.8	0	550	5053	1917.41	209	5053	307.09	210
100	5	5630	3.34	0	5630	5.92	0	18	5630	**1.09**	2	5630	6.97	2
100	10	6987	7200	1.98	6855	4384.63	0	359	6855	1186.13	128	6855	**611.90**	43
100	15	8175	7200	3.56	8357	7200	6.01	225	7933	7200	66	7933	7200	68
150	5	8335	24.48	0	8335	20.16	0	21	8335	**2.34**	2	8335	6.09	2
150	10	9641	7200	1.46	9837	7200	3.51	66	9502	**56.62**	8	9502	1918.77	33
150	15	11077	7200	5.27	11556	7200	10.93	42	10626	7200	14	10621	7200	26
200	5	11121	86.23	0	11121	24.33	0	15	11121	**4.5**	2	11121	8.87	2
200	10	13096	7200	1.07	13453	7200	4.4	35	12980	7200	25	12978	7200	40
200	15	14322	7200	4.04	17055	7200	-	-	13796	7200	27	13837	7200	20
250	5	13357	239.06	0	13357	56.67	0	15	13357	**9.57**	2	13357	16.86	2
250	10	15567	7200	1.34	17619	7200	-	-	15376	7200	32	15373	7200	36
250	15	20366	7200	100	20366	7200	-	-	16285	7200	14	16310	7200	19
300	5	15160	7200	0.38	15162	7200	0.4	38	15102	**14.23**	2	15102	22.55	2
300	10	17451	7200	1.58	20128	7200	-	-	17185	7200	21	17185	7200	25
300	15	19053	7200	3.26	22779	7200	-	-	18540	7200	13	18556	7200	13
350	5	18329	1536.28	0	18329	1216.45	0	55	18329	**23.13**	2	18329	49.81	2
350	10	21299	7200	1.24	23394	7200	-	-	21044	4618.28	20	21044	**3578.19**	23
350	15	26428	7200	100	26428	7200	-	-	22301	7200	15	22294	7200	13
400	5	21094	1697.33	0	21094	174.42	0	17	21094	**30.5**	2	21094	49.96	2
400	10	23967	7200	1.82	26042	7200	-	-	23542	7200	18	23541	**632.66**	27
400	15	26372	7200	6.61	29034	7200	-	-	24676	7200	17	24695	7200	10
450	5	23682	2881.28	0	23682	361.59	0	13	23682	**40.06**	2	23682	62.56	2
450	10	29314	7200	100	29314	7200	-	-	26445	7200	16	26445	**2963.05**	14
450	15	32151	7200	100	32151	7200	-	-	27499	7200	14	27430	7200	17
500	5	25401	7200	0.94	25218	7200	0.23	80	25161	**325.69**	8	25161	1138.62	12
500	10	32018	7200	100	32018	7200	-	-	27426	7200	21	27426	7200	24
500	15	34792	7200	100	34792	7200	-	-	28504	7200	9	28336	**3437.65**	15

Acknowledgements This research project was partially supported by the Scientific and Technological Research Council of Turkey (TÜBITAK) under grant 1059B191600107. While writing this paper, Dr Hamzadayı was a visiting researcher at the Southampton Business School at the University of Southampton. Rubén Ruiz is supported is supported by the Spanish Ministry of Science, Innovation, and Universities, under the project "OPTEP-Port Terminal Operations Optimization"(No. RTI2018-094940-B-I00) financed with FEDER funds.

Thanks are due to two anonymous reviewers for their careful reading of the paper and helpful suggestions.

References

Adiri I, Pohoryles D (1982) Flowshop no-idle or no-wait scheduling to minimize the sum of completion times. Naval Research Logistics 29(3):495–504

Baker KR (1974) Introduction to Sequencing and Scheduling. John Wiley & Sons, New York

Baptiste P, Hguny LK (1997) A branch and bound algorithm for the F/no-idle/C_{max}. In: Proceedings of the International Conference on Industrial Engineering and Production Management, IEPM'97, Lyon, France, vol 1, pp 429–438

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4(1):238–252

Cordeau JF, Pasin F, Solomon M (2006) An integrated model for logistics network design. Annals of Operations Research 144(1):59–82

Costa AM, Cordeau JF, Gendron B, Laporte G (2012) Accelerating Benders decomposition with heuristic master problem solutions. Pesquisa Operacional 32(1):3–20

Deng G, Gu X (2012) A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion. Computers & Operations Research 39(9):2152–2160

Goncharov Y, Sevastyanov S (2009) The flow shop problem with no-idle constraints: A review and approximation. European Journal of Operational Research 196(2):450–456

Kalczynski PJ, Kamburowski J (2005) A heuristic for minimizing the makespan in no-idle permutation flow shops. Computers & Industrial Engineering 49(1):146–154

Magnanti TL, Wong RT (1981) Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. Operations Research 29(3):464–484

Pan QK, Ruiz R (2014) An effective iterated greedy algorithm for the mixed no-idle flowshop scheduling problem. Omega 44(1):41–50

Pan QK, Wang L (2008a) No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. International Journal of Advanced Manufacturing Technology 39(7-8):796–807

Pan QK, Wang L (2008b) A novel differential evolution algorithm for no-idle permutation flow-shop scheduling problems. European Journal of Industrial Engineering 2(3):279–297

Table 6 Comparisons with a state-of-the-art heuristic

n	m	**BI**	**Method**	**IGA**				**IGA**
				Min.	**Max.**	**Avg.**	**Std.**	(7200s)
50	5	**3083**	All	**3083**	**3083**	**3083**	0	**3083**
50	10	**4050**	All	**4050**	**4050**	**4050**	0	**4050**
50	15	**5053**	All	**5053**	5100	5062.4	21.01	**5053**
100	5	**5630**	All	**5630**	**5630**	**5630**	0	**5630**
100	10	**6855**	ABD/HE2/HE2+PO	**6855**	6883	6867.8	10.03	**6855**
100	15	**7933**	HE2/HE2+PO	7935	7975	7956.8	14.46	**7933**
150	5	**8335**	All	**8335**	**8335**	**8335**	0	**8335**
150	10	**9502**	HE2/HE2+PO	**9502**	9549	9511.4	21.01	**9502**
150	15	10621	HE2+PO	10636	10675	10648.4	17.89	**10617**
200	5	**11121**	All	**11121**	**11121**	**11121**	0	**11121**
200	10	**12978**	HE2+PO	**12978**	13002	12995.6	10.11	**12978**
200	15	13796	HE2	13796	13857	13819	26.76	**13795**
250	5	**13357**	All	**13357**	**13357**	**13357**	0	**13357**
250	10	15373	HE2+PO	15376	15407	15392	12.7	**15360**
250	15	**16285**	HE2	16286	16362	16325.8	29.98	**16285**
300	5	**15102**	HE2/HE2+PO	15120	15141	15130.6	7.43	**15102**
300	10	**17185**	HE2/HE2+PO	**17185**	17225	17199	16.37	17188
300	15	18540	HE2	18562	18589	18578.2	14.78	**18481**
350	5	**18329**	All	**18329**	**18329**	**18329**	0	**18329**
350	10	**21044**	HE2/HE2+PO	21050	21066	21057.6	6.34	**21044**
350	15	**22294**	HE2+PO	22299	22313	22305.4	6.8	22303
400	5	**21094**	All	**21094**	**21094**	**21094**	0	**21094**
400	10	**23541**	HE2+PO	23561	23580	23567	7.48	23542
400	15	**24676**	HE2	24687	24776	24718.2	38.04	24714
450	5	**23682**	All	**23682**	23682	23682	0	**23682**
450	10	**26445**	HE2/HE2+PO	26446	26451	26448	2.73	26446
450	15	**27430**	HE2+PO	27594	27644	27610.4	22.38	27596
500	5	**25161**	HE2/HE2+PO	**25161**	25193	25171	12.81	**25161**
500	10	**27426**	HE2/HE2+PO	27428	27443	27436.8	5.44	27443
500	15	**28336**	HE2+PO	28446	28496	28466	27.38	28446

Pan QK, Fatih Tasgetiren M, Liang YC (2008) A discrete differential evolution algorithm for the permutation flowshop scheduling problem. Computers & Industrial Engineering 55(4):795–816

Papadakos N (2008) Practical enhancements to the Magnanti–Wong method. Operations Research Letters 36(4):444–449

Röck H (1984) The three-machine no-wait flow shop is NP-complete. Journal of the Association for Computing Machinery 31(2):336–345

Ruiz R, Maroto C (2005) A comprehensive review and evaluation of permutation flowshop heuristics. European Journal of Operational Research 165(2):479–494

Ruiz R, Vallada E, Fernández-Martínez C (2009) Scheduling in flowshops with no-idle machines. In: Chakraborty U (ed) Computational Intelligence in Flow Shop and Job Shop Scheduling, Springer, New York, chap 2, pp 21–51

Saadani NEH, Guinet A, Moalla M (2003) Three stage no-idle flow-shops. Computers & Industrial Engineering 44(3):425–434

Saharidis G, Ierapetritou M (2013) Speed-up Benders decomposition using maximum density cut (MDC) generation. Annals of Operations Research

210:101–123
Shao W, Pi D, Shao Z (2017) Memetic algorithm with node and edge histogram for no-idle flow shop scheduling problem to minimize the makespan criterion. Applied Soft Computing 54:164–182
Tasgetiren MF, Buyukdagli O, Pan QK, Suganthan PN (2013) A general variable neighborhood search algorithm for the no-idle permutation flowshop scheduling problem. In: Panigrahi BK, Suganthan PN, Das S, Dash SS (eds) Swarm, Evolutionary, and Memetic Computing, Springer International Publishing, Cham, pp 24–34
Vachajitpan P (1982) Job sequencing with continuous machine operation. Computers & Industrial Engineering 6(3):255–259