Journal of Heuristics (2021) 27:353–438
https://doi.org/10.1007/s10732-020-09461-x

Iterated local search for single machine total weighted tardiness batch scheduling

Eduardo Queiroga[1] · Rian G. S. Pinheiro[2] · Quentin Christ[3] · Anand Subramanian[4] · Artur A. Pessoa[5]

Received: 18 March 2020 / Revised: 17 July 2020 / Accepted: 19 October 2020 /
Published online: 9 November 2020

Abstract
This paper presents an iterated local search (ILS) algorithm for the single machine total weighted tardiness batch scheduling problem. To our knowledge, this is one of the first attempts to apply ILS to solve a batching scheduling problem. The proposed algorithm contains a local search procedure that explores five neighborhood structures, and we show how to efficiently implement them. Moreover, we compare the performance of our algorithm with dynamic programming-based implementations for the problem, including one from the literature and two other ones inspired in biased random-key genetic algorithms and ILS. We also demonstrate that finding the optimal batching for the problem given a fixed sequence of jobs is $\mathcal{NP}$-hard, and provide an exact pseudo-polynomial time dynamic programming algorithm for solving such problem. Extensive computational experiments were conducted on newly proposed benchmark instances, and the results indicate that our algorithm yields highly competitive results when compared to other strategies. Finally, it was also observed that the methods that rely on dynamic programming tend to be time-consuming, even for small size instances.

Keywords Batch scheduling · Total weighted tardiness · Iterated local search · Metaheuristics

1 Introduction

Let $J = \{1, 2, \ldots, n\}$ be a set of jobs to be processed by a single machine.

Each job $j \in J$ has the following attributes: processing time p_j, size s_j, release date r_j (earliest time that job j can begin to be processed), due date d_j and a positive weight w_j. The single machine total weighted tardiness batch scheduling problem (TWTBSP) involves two significant decisions: partitioning the jobs into batches (each job must be in exactly one of the batches) and sequencing the batches. A batch-

Extended author information available on the last page of the article

Table 1 A small TWTBSP instance

Job	p_j	d_j	s_j	w_j	r_j
1	2	3	1	1	4
2	2	2	1	1	0
3	1	3	2	2	2
4	2	4	2	2	1
5	3	2	3	1	0
6	1	1	1	3	1
$Q = 4$					

processing machine (or batching machine) can process several jobs simultaneously as a batch, and the processing time of an each of them is equal to the longest processing time among all jobs in the corresponding batch. For a given sequence of m batches $\mathcal{B} = (B_1, B_2, \ldots, B_m)$, the jobs in B_i are processed together and therefore such batch can only start to be processed at time $r(B_i) = \max_{j \in B_i}\{r_j\}$, and has a processing time $p(B_i) = \max_{j \in B_i}\{p_j\}$. In addition, $C(B_1) = r(B_1) + p(B_1)$ denotes the completion time of the first batch and $C(B_i) = \max\{r(B_i), C(B_{i-1})\} + p(B_i)$ the completion time of B_i, for $i = 2, 3, \ldots, m$.

The machine has a capacity Q which must be satisfied, i.e., $s(B_i) = \sum_{j \in B_i} s_j \leq Q$ for each batch B_i. The tardiness of a job j is given by $T_j = \max\{0, C_j - d_j\}$, where C_j is completion time of job j (which is equal to the completion time of the batch that j belongs). The objective is to find a collection of batches over J and a sequence of these batches, which minimizes $\sum_{j \in J} w_j T_j$. We will hereafter denote the total weighted tardiness (TWT) of a solution $\mathcal{B}$ as $TWT(\mathcal{B})$.

According to the 3-field notation widely used in the scheduling literature (Graham et al. 1979), TWTBSP can be denoted as $1|batch, r_j, s_j, compt|\sum w_j T_j$. The term *compt* indicates that any pair of jobs can be in the same batch, meaning that they are all compatible with each other, as opposed to similar versions of the problem that assume some incompatible job families (Perez et al. 2005; Tangudu and Kurz 2006; Erramilli and Mason 2008; Kurz and Mason 2008). The particular case of TWTBSP in which $r_j = 0$ and $s_j = 1, \forall j \in J$, was proven $\mathcal{NP}$-hard in the strong sense (Brucker et al. 1998), thus motivating the use of heuristic approaches for solving medium and large instances of the problem.

Table 1 presents an example containing a 6-job instance, whose optimal solution is illustrated in Fig. 1.

According to Potts and Kovalyov (2000), there are two main classes of batch scheduling: (i) the family scheduling models (Potts and Wassenhove 1992; Kress et al. 2018); and (ii) the batching machine models (Brucker et al. 1998; Pinedo 2016). In the first (a.k.a. serial-batching), jobs are grouped into families, and there are no setup costs when processing two consecutive jobs that belong to the same family. In this type of model, jobs from the same family are scheduled contiguously to avoid setup

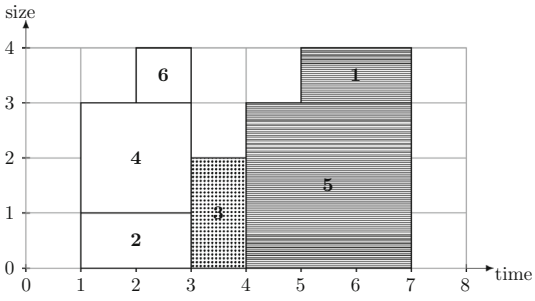


Fig. 1 Graphical representation of a solution for the instance presented in Table 1. The sequence $\mathcal{B} = (\{2, 4, 6\}, \{3\}, \{1, 5\})$ corresponds to the optimal solution with $TWT(\mathcal{B}) = \sum_{j \in J} w_j T_j = (w_2 T_2 + w_4 T_4 + w_6 T_6) + (w_3 T_3) + (w_1 T_1 + w_5 T_5) = 18$

operations. In the second model, jobs are allowed to be scheduled simultaneously, which is the case of the problem considered in this paper.

Motivated by many applications in the manufacturing industry (Mathirajan and Sivakumar 2006; Mönch et al. 2011; Blazewicz et al. 2019), batch scheduling problems have been intensively studied over the past decades. In Lee et al. (1992), a batch scheduling application for semiconductor burn-in operations in device production (e.g., microprocessors and DRAM) was introduced. The purpose of burn-in operations is to subject the chips to thermal stress for an extended period of time in order to bring out latent defects known as infant mortality. A set of chips (jobs) of the same product is placed on one or more boards (the job size is given by the number of boards), and the boards are loaded into the oven. Hence, boards related to different products are grouped into a batch to be processed together in the same oven considering the batch capacity. The processing time of each product is the minimum testing time that it should be exposed to a high temperature. The processing time of a batch is equal to the longest time processing time among all products in the batch because the burn-in phase cannot be interrupted until the longest testing time is performed (other products may remain in the oven longer than necessary). Minimizing the total weighted tardiness

can significantly reduce the costs involved in the production process of circuits, due to the fact that it is a long-term test compared to other testing operations. The main benefit of using the TWT as the objective function is that, in many scenarios, the industry requires metrics focusing on due dates.

TWTBSP was approached by relatively few works in the literature. Chou and Wang (2008) presented a mathematical formulation and two hybrid heuristics using a dynamic programming (DP) procedure, one of them based on a genetic algorithm (GA), which in turn was the one that found the best results. Mathirajan et al. (2010) designed greedy constructive heuristics and a simulated annealing (SA) metaheuristic, which were tested on instances with up to 100 jobs. Wang (2011) proposed a two-phase heuristic called iterated heuristic (IH), where the first phase consists of a population-based reasoning (PBR) procedure which generates a set of job sequences, and the second one consists of forming and sequencing the batches for each sequence by means of the DP algorithm proposed by Chou and Wang (2008). In addition, the same author also employed an iterative approach to improve the solution obtained by the two-phase procedure. In this case, better solutions were obtained than those achieved by the two algorithms proposed in Chou and Wang (2008). Vélez-Gallego et al. (2011) put forward a variable neighborhood search (VNS) metaheuristic, which

found optimal solutions for instances with up to 15 jobs, whereas Zinouri et al. (2012) developed a greedy randomized adaptive search procedure (GRASP) for the problem. More recently, Kohn (2015) devised a VNS-based framework for the general batch scheduling problem which makes use of neighborhood structures involving operations over jobs (insertion and swap) and batches (insertion, swap, split and merge).

When considering TWT batch scheduling problems on a parallel machine environment, very few works addressed the problem taking into account compatible jobs (Hulett et al. 2017). In contrast, there are several references that addressed the problem with families of incompatible jobs, e.g., Balasubramanian et al. (2004), Chiang et al. (2010), Almeder and Mönch (2011), Mönch et al. (2005), Mönch and Roob (2018).

Furthermore, other single machine batch scheduling works considered different objectives, namely: completion time (Chang and Wang 2004), flow time (Zee 2007) and makespan (Li et al. 2005; Chou et al. 2006; Lu et al. 2010; Velez-Gallego et al. 2011; Xu et al. 2012; Zhou et al. 2014).

The main contributions of this paper are as follows:

- We propose a metaheuristic algorithm based on iterated local search (ILS) for solving TWTBSP. To the best of our knowledge, this is one of the first attempts to apply ILS (Lourenço et al. 2019) to solve a batch scheduling problem. In contrast, ILS-based procedures have been successfully implemented to solve other classes of scheduling problems (Grosso et al. 2004; Subramanian et al. 2014; Subramanian and Farias 2017; Lourenço et al. 2019). We thus attempt to fill this gap in the present work, as ILS appears to be a highly promising strategy to efficiently solve different families of scheduling problems.
- We show how to efficiently implement the local search procedures used in our algorithm by avoiding unnecessary operations.
- We demonstrate that finding the optimal batching for TWTBSP given a fixed sequence of jobs is $\mathcal{NP}$-hard, and provide an exact pseudo-polynomial time DP algorithm for solving such problem. Moreover, we also compare the performance of our algorithm with DP-based implementations for the problem, including the heuristic by Wang and Chou (2013) and two other ones inspired in biased random-key genetic algorithms (BRKGA) and ILS. The results show that the former achieves a far better performance when compared to the DP approaches. In addition, we tried to replace the non-exact DP procedure (Chou and Wang 2008) used in the existing heuristics (as a decoder for a job sequence) with the proposed exact DP approach, but the CPU time turned out to be prohibitively expensive, and the solutions obtained using the latter were similar to those found when utilizing the former.
- We introduce a new set of benchmark instances for TWTBSP, as those used in previous works are no longer available.

The remainder of this paper is organized as follows. In Sect. 2, we provide a comprehensive description of our ILS algorithm. Section 3 discusses the DP-based implementations. Section 4 contains the results obtained, whereas Section 5 presents the concluding remarks of this work.

2 Proposed metaheuristic algorithm

ILS is a metaheuristic that explores the space of local optima in order to find a global optimum (Lourenço et al. 2019). It works by applying perturbations to the current local optimum solution, followed by a local search over the perturbed solution. If the solution obtained after the local search satisfies some conditions, it is accepted as the new current solution. This process is repeated until a given stopping criterion is met. In this work, we use a randomized variable neighborhood descent (RVND) procedure in the local search phase of the ILS algorithm. The basic idea of the traditional variable neighborhood descent (VND) approach is to successively explore a set of neighborhoods to improve the current solution (Hansen et al. 2010) using a predefined neighborhood ordering. In the case of RVND, the order in which the neighborhoods are explored is random. The latter is shown to yield promising results when compared to the former strategy (Penna et al. 2013). The combination of ILS and RVND has been successfully employed in solving several combinatorial optimization problems, such as vehicle routing (Penna et al. 2013), scheduling (Subramanian et al. 2014), and manufacturing cell formation (Martins et al. 2015).

Algorithm 1 describes the proposed multi-start metaheuristic, called $\text{ILS}_{\text{batch}}$. The algorithm receives four input parameters: I_R is the total number of restarts, I_{ILS} is the number of consecutive ILS iterations without improvement, I_P is the maximum number of perturbation moves, and α is a parameter used in the constructive heuristic CH_{EDD} (line 4). $\text{ILS}_{\text{batch}}$ follows the standard ILS framework (Lourenço et al. 2019), by iteratively applying local search and perturbation moves (lines 8 and 12).

```
1  Procedure ILS_batch(I_R, I_ILS, I_P, α)
2      TWT* ← ∞
3      for iter = 1 ... I_R do
4          B ← CH_EDD(α)
5          B' ← B
6          iter_ILS ← 0
7          while iter_ILS < I_ILS do
8              B ← LocalSearch(B);
9              if TWT(B) < TWT(B') then
10                 B' ← B
11                 iter_ILS ← 0
12             B ←Perturbation(B', I_P)
13             iter_ILS ← iter_ILS + 1
14         if TWT(B') < TWT* then
15             B* ← B'
16             TWT* ← TWT(B')
17     return B*
```

Algorithm 1: $\text{ILS}_{\text{batch}}$

In what follows, we describe the main components of $\text{ILS}_{\text{batch}}$.

2.1 Constructive procedure

The constructive heuristic CH_{EDD}, described in Algorithm 2, is based on the *earlies due date* (EDD) rule, which is a well-known in the scheduling literature (Vepsalainen 1987). Basically, the algorithm combines greediness and randomness by selecting one of the first α jobs in L (denoted by $F(\alpha, L)$). If the chosen job does not fit in the batch, the algorithm tries to place the job with smaller s_j among the first α jobs (lines 10–16).

```
1  Procedure CH_EDD(J, Q, α)
2      Let L be the list of jobs in J ordered in increasing order of their EDD rule
3      Let B ← ∅ be a batch sequence solution
4      Let B' ← ∅ be a new batch to be created
5      while L ≠ ∅ do
6          Select a random job j from F(α, L)
7          if s_j + s(B') ≤ Q then
8              B' ← B' ∪ {j}; L ← L \ {j}
9          else
10             j' ← arg min_{i∈F(α,L))\j} s_i
11             if s_{j'} + s(B') ≤ Q then
12                 B' ← B' ∪ {j'}; L ← L \ {j'}
13             else
14                 add the batch B' at the end of list B
15                 B' ← ∅ // creating an empty batch
16                 B' ← B' ∪ {j}; L ← L \ {j}
17     return B
```

Algorithm 2: Constructive heuristic using EDD rule.

2.2 Local search

Algorithm 3 shows the pseudocode of the VND-based procedure, which employs the *best improvement* strategy, where each neighborhood of a solution $\mathcal{B}$ is completely examined to determine the move (neighbor) that yields the best TWT improvement.

A neighborhood from set $\mathcal{N}$ is systematically selected (line 11) to continue the search whenever the current neighborhood fails to improve the current solution. In case of improvement, the search continues from the first neighborhood, randomly redefined, over the new solution found (lines 8–9). Here, at least one neighborhood structure is necessary, but it is interesting if more neighborhoods are available. Generally, the larger the number of distinct neighborhoods, the better the results.

```
1  LocalSearch(B)
2      initialize the set of neighborhoods N in a random order
3      k ← 1 // neighborhood structure selector
4      while k ≤ |N| do
           // find the best improvement neighbor B' of B in N_k(B)
5          B' ← BestImprovement(N_k, B)
6          if TWT(B') < TWT(B) then
7              reinitialize N in a random order
8              k ← 1
9              B ← B'
10         else
11             k ← k + 1
12     return B
```

Algorithm 3: VND-based local search.

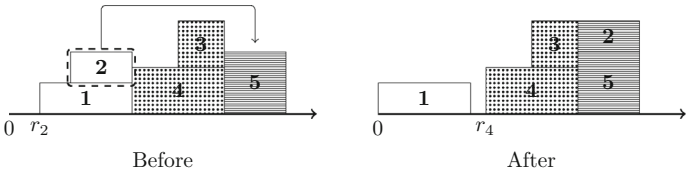


Fig. 2 Let $\mathcal{B} = (B_1, B_2, B_3)$ be a solution. Job 2 was moved from B_1 to B_3

The $\text{ILS}_{\text{batch}}$ operates over five neighborhood structures, which will be thoroughly described in the next subsections.

2.2.1 Job insertion

In this neighborhood, hereafter referred to as $\text{insert}_{\text{job}}$, a job is removed from its current batch and inserted in another batch (see Fig. 2).

A move is valid only if the capacity Q is not exceeded, and the number of batches is maintained (if a job is the only one in the batch, it cannot be moved). There are at most $n \times (m-1)$ neighbors for a solution $\mathcal{B}$, where $n = |J|$ and $m = |\mathcal{B}|$. This neighborhood has complexity $\mathcal{O}(n^2)$, since in the worst case we have $m = n$ (all batches having only one job). Algorithm 4 describes the $\text{insert}_{\text{job}}$ move evaluation in $\mathcal{O}(n)$ time.
Therefore, the best $\text{insert}_{\text{job}}$ move can be determined in $\mathcal{O}(n^3)$ time.

2.2.2 Job swap

In this neighborhood, hereafter referred to as swap_{job}, two jobs from different batches are swapped (see Fig. 3).

Procedure `EvalJobInsertion`$(\mathcal{B}, k, j, l)$

Let $m = |\mathcal{B}|$ and $\mathcal{L} = \min\{k, l\}$

$B_k \leftarrow B_k \setminus \{j\}$ and $B_l \leftarrow B_l \cup \{j\}$ `// performs the move`

Update processing time and release date for B_k and B_l

Let $\delta \leftarrow 0$ be the improvement induced by the move

Let $c'(B_{\mathcal{L}-1}) = c(B_{\mathcal{L}-1})$ if $\mathcal{L} > 1$.

for $i = \mathcal{L}, \mathcal{L}+1, \ldots, m$ **do**

  if $i = 1$ **then**

    $c'(B_i) \leftarrow r(B_i) + p(B_i)$ `// new completion time`

  else

    $c'(B_i) \leftarrow \max\{c'(B_{i-1}), r(B_i)\} + p(B_i)$ `// new completion time`

  `// improvement induced by` j

  if $i = l$ **then**

    $\delta \leftarrow \delta + (w_j \times \max\{0, c(B_k) - d_j\}) - (w_j \times \max\{0, c'(B_l) - d_j\})$

  `// improvement induced by other jobs`

  forall the $q \in B_i$ **do**

    if $q \neq j$ **then**

      $\delta \leftarrow \delta + (w_q \times \max\{0, c(B_i) - d_q\}) - (w_q \times \max\{0, c'(B_i) - d_q\})$

$B_l \leftarrow B_l \setminus \{j\}$ and $B_k \leftarrow B_k \cup \{j\}$ `// undo the move`

Update processing time and release date for B_k and B_l

return $(TWT(\mathcal{B}) - \delta)$

Algorithm 4: Evaluation of a job insertion. The solution evaluated is obtained by moving the job j from B_k to B_l.

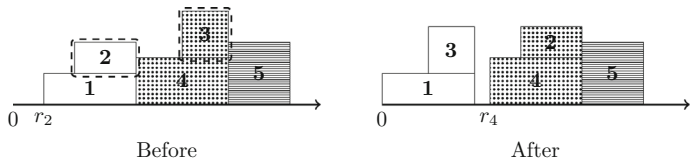


Fig. 3 Let $\mathcal{B} = (B_1, B_2, B_3)$ be a solution. Jobs $2 \in B_1$ and $3 \in B_2$ were swapped

A move is valid only if the capacity Q is not exceeded in both batches. When $n = m$, there are at most $n(n-1)/2$ neighbors, thus this neighborhood has complexity $\mathcal{O}(n^2)$. Algorithm 5 describes the swap_{job} move evaluation in $\mathcal{O}(n)$ time.
Therefore, the best swap_{job} move can be found in $\mathcal{O}(n^3)$ time.

Procedure `EvalJobSwap` $(\mathcal{B}, k, i, l, j)$
  Let $m = |\mathcal{B}|$
  $B_k = (B_k \setminus \{i\}) \cup \{j\}$ and $B_l = (B_l \setminus \{j\}) \cup \{i\}$ `// performs the move`
  Update processing time and release date for B_k and B_l
  Let $\delta = 0$ be the improvement induced by the move
  Let $c'(B_{k-1}) = c(B_{k-1})$ if $k > 1$.
  for $o = k, k+1, \ldots, m$ **do**
    if $o = 1$ **then**
      $c'(B_o) \leftarrow r(B_o) + p(B_o)$ `// new completion time`
    else
      $c'(B_o) \leftarrow \max\{c'(B_{o-1}), r(B_o)\} + p(B_o)$ `// new completion time`
    if $o = k$ **then**
      `// improvement induced by` j
      $\delta \leftarrow \delta + (w_j \times \max\{0, c(B_l) - d_j\}) - (w_j \times \max\{0, c'(B_k) - d_j\})$
    if $o = l$ **then**
      `// improvement induced by` i
      $\delta \leftarrow \delta + (w_i \times \max\{0, c(B_k) - d_i\}) - (w_i \times \max\{0, c'(B_l) - d_i\})$
    `// improvement induced by other jobs`
    forall the $q \in B_o$ **do**
      if $q \neq i$ *and* $q \neq j$ **then**
        $\delta \leftarrow \delta + (w_q \times \max\{0, c(B_o) - d_q\}) - (w_q \times \max\{0, c'(B_o) - d_q\})$
  $B_k = (B_k \setminus \{j\}) \cup \{i\}$ and $B_l = (B_l \setminus \{i\}) \cup \{j\}$ `// undo the move`
  Update processing time and release date for B_k and B_l
  return $(TWT(\mathcal{B}) - \delta)$

Algorithm 5: Evaluation of a job swap. The solution evaluated is obtained by swapping the jobs $i \in B_k$ and $j \in B_l$, where $k < l$.

2.2.3 Batch insertion

In this neighborhood, hereafter referred to as $\text{insert}_{\text{batch}}$, a batch is removed from its current position and inserted in another one (see Fig. 4).

Removing and inserting a batch from the k-th position to l-th position is only valid if $|k - l| > 1$, avoiding intersection with the $\text{swap}_{\text{batch}}$ described in Sect. 2.2.4.

This neighborhood has complexity $\mathcal{O}(n^2)$, since in the worst case we have $m = n$ and for each batch there are $\mathcal{O}(n)$ moves. Algorithm 6 describes the $\text{insert}_{\text{batch}}$ move evaluation in $\mathcal{O}(n)$ time. Therefore, the best $\text{insert}_{\text{batch}}$ move can be found in $\mathcal{O}(n^3)$ time.

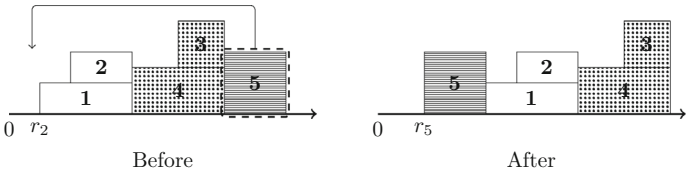


Fig. 4 Let $\mathcal{B} = (B_1, B_2, B_3)$ be a solution. Batch B_3 was moved to the first position

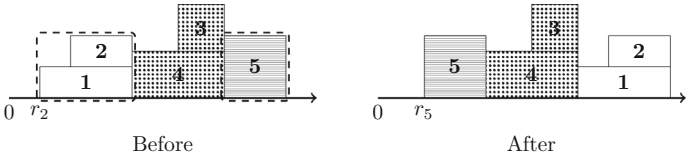


Fig. 5 Swap between batches B_1 and B_3

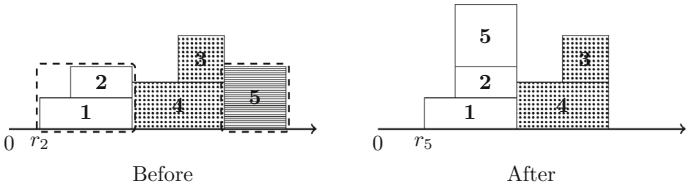


Fig. 6 Batches B_1 and B_3 are merged into one

2.2.4 Batch swap

In this neighborhood, hereafter referred to as $\text{swap}_{\text{batch}}$, two different batches are swapped (see Fig. 5).

This neighborhood has complexity $\mathcal{O}(n^2)$, since there are $n(n-1)/2$ possible swaps when $m = n$. Algorithm 7 describes the $\text{swap}_{\text{batch}}$ move evaluation in $\mathcal{O}(n)$ time. Therefore, the best $\text{swap}_{\text{batch}}$ move can be found in $\mathcal{O}(n^3)$ time.

2.2.5 Batch merge

In this neighborhood, hereafter referred to as $\text{merge}_{\text{batch}}$, two different batches are merged (see Fig. 6). The new batch is positioned in place of the leftmost batch between the two involved in the move.

A merge is only valid if the capacity Q is not violated. This neighborhood has complexity $\mathcal{O}(n^2)$, since there are at most $n \times (n-1)/2$ moves when $n = m$. Algorithm 8 describes the move evaluation in $\mathcal{O}(n)$ time. Therefore, the best move can be found in $\mathcal{O}(n^3)$ time.

Procedure `EvalBatchInsertion`$(\mathcal{B}, k, l)$
- Let $m = |\mathcal{B}|$
- Let $\mathcal{B}(i)$ be the i-th batch of $\mathcal{B}$, where initially $\mathcal{B} = (B_1, B_2, \ldots, B_m)$.
- Let $\delta \leftarrow 0$ be the improvement induced by the move.
- **if** $k < l$ **then**
  - $\mathcal{B} = (\ldots, B_{k-1}, B_{k+1}, \ldots, B_l, B_k, \ldots)$ `// performs the move`
  - Let $c'(B_{k-1}) = c(B_{k-1})$ if $k > 1$.
  - **for** $i = k, k+1, \ldots, m$ **do**
    - **if** $i = 1$ **then**
      - $c'(\mathcal{B}(i)) \leftarrow r(\mathcal{B}(i)) + p(\mathcal{B}(i))$
    - **else**
      - $c'(\mathcal{B}(i)) \leftarrow \max\{c'(\mathcal{B}(i-1)), r(\mathcal{B}(i))\} + p(\mathcal{B}(i))$
    - **forall the** $q \in \mathcal{B}(i)$ **do**
      - $\delta \leftarrow \delta + (w_q \times \max\{0, c(\mathcal{B}(i)) - d_q\}) - (w_q \times \max\{0, c'(\mathcal{B}(i)) - d_q\})$
  - $\mathcal{B} = (\ldots, B_{k-1}, B_k, \ldots, B_l, B_{l+1}, \ldots)$ `// undo the move`
- **else**
  - $\mathcal{B} = (\ldots, B_{l-1}, B_k, B_l \ldots, B_{k-1}, B_{k+1}, \ldots)$ `// performs the move`
  - Let $c'(B_{l-1}) = c(B_{l-1})$ if $l > 1$.
  - **for** $i = l, l+1, \ldots, m$ **do**
    - **if** $i = 1$ **then**
      - $c'(\mathcal{B}(i)) \leftarrow r(\mathcal{B}(i)) + p(\mathcal{B}(i))$
    - **else**
      - $c'(\mathcal{B}(i)) \leftarrow \max\{c'(\mathcal{B}(i-1)), r(\mathcal{B}(i))\} + p(\mathcal{B}(i))$
    - **forall the** $q \in \mathcal{B}(i)$ **do**
      - $\delta \leftarrow \delta + (w_q \times \max\{0, c(\mathcal{B}(i)) - d_q\}) - (w_q \times \max\{0, c'(\mathcal{B}(i)) - d_q\})$
  - $\mathcal{B} = (\ldots, B_{l-1}, B_l \ldots, B_{k-1}, B_k, B_{k+1}, \ldots)$ `// performs the move`
- **return** $(TWT(\mathcal{B}) - \delta)$

Algorithm 6: Evaluation of a batch insertion. The solution evaluated is obtained by moving the batch B_k to position l.

2.3 Perturbation

Perturbation is a key ILS feature which aims at slightly modifying a local optimal solution, thus allowing the local search to converge to a different, and possibly better, local optimum. In this work, we consider the following perturbation mechanisms:

Procedure `EvalBatchSwap`$(\mathcal{B}, k, l)$
- Let $m = |\mathcal{B}|$
- Let $\mathcal{B}(i)$ the i-th batch of $\mathcal{B}$, where initially $\mathcal{B} = (B_1, B_2, \ldots, B_m)$.
- Let $\delta \leftarrow 0$ be the improvement induced by the move.
- $\mathcal{B} = (\ldots, B_{k-1}, B_l, B_{k+1}, \ldots, B_{l-1}, B_k, B_{l+1}, \ldots)$ `// performs the move`
- Let $c'(B_{k-1}) = c(B_{k-1})$ if $k > 1$.
- **for** $i = k, k+1, \ldots, m$ **do**
  - **if** $i = 1$ **then**
    - $c'(\mathcal{B}(i)) \leftarrow r(\mathcal{B}(i)) + p(\mathcal{B}(i))$
  - **else**
    - $c'(\mathcal{B}(i)) \leftarrow \max\{c'(\mathcal{B}(i-1)), r(\mathcal{B}(i))\} + p(\mathcal{B}(i))$
  - **forall the** $q \in \mathcal{B}(i)$ **do**
    - $\delta \leftarrow \delta + (w_q \times \max\{0, c(\mathcal{B}(i)) - d_q\}) - (w_q \times \max\{0, c'(\mathcal{B}(i)) - d_q\})$
- $\mathcal{B} = (\ldots, B_{k-1}, B_k, B_{k+1}, \ldots, B_{l-1}, B_l, B_{l+1}, \ldots)$ `// undo the move`
- **return** $(TWT(\mathcal{B}) - \delta)$

Algorithm 7: Evaluation of a batch swap. The solution evaluated is obtained by exchanging the batches B_k and B_l, such that $k < l$.

1. *Swap of jobs*—A random move from the neighborhood swap_{job} is applied.
2. *Batch split*—At first, a batch $B_i \in \mathcal{B}$ and a cut point $t \in [2, |B_i| - 1]$ are selected. Next, assuming that B_i is in non-ascending order w.r.t. the processing time, B_i is divided into two new batches B_i' and B_i'', where B_i' receives the first t jobs of B_i and B_i'' receives the remaining ones. B_i' and B_i'' are then included, in this order, in place of B_i in the sequence $\mathcal{B}$ (B_i is replaced by the new batches).

At each call of the perturbation procedure, one of the two first mechanisms is randomly chosen, and at most I_P moves associated to the selected mechanism are applied.

If the swap of jobs perturbation is selected and at least one feasible move could not be performed due to the capacity constraints, then the batch split mechanism is applied.

3 Dynamic programming-based approaches

In the heuristic by Wang (2011), a candidate solution is represented by a sequence of jobs, and the DP algorithm proposed by Chou and Wang (2008) is used as a *solution decoder*, i.e., to define a batching from the sequence and to compute the TWT. The problem of finding an optimal batching over a given fixed sequence of jobs (hereafter referred to as $\text{TWTBSP}_{\text{seq}}$) is formally defined and proved to be $\mathcal{NP}$-hard in

Procedure `EvalBatchMerge`$(\mathcal{B}, k, l)$
- Let $m = |\mathcal{B}|$
- Let $\mathcal{B}(i)$ the i-th batch of $\mathcal{B}$, where initially $\mathcal{B} = (B_1, B_2, \ldots, B_m)$.
- Let $\delta \leftarrow 0$ be the improvement induced by the move.
- Let B_{kl} be the batch obtained by merging B_k and B_l.
- $\mathcal{B} = (\ldots, B_{k-1}, B_{kl}, B_{k+1}, \ldots, B_{l-1}, B_{l+1}, \ldots)$ // performs the move
- Let $c'(B_{k-1}) = c(B_{k-1})$ if $k > 1$.
- **for** $i = k, k+1, \ldots, (m-1)$ **do**
  - **if** $i = 1$ **then**
    - $c'(\mathcal{B}(i)) \leftarrow r(\mathcal{B}(i)) + p(\mathcal{B}(i))$
  - **else**
    - $c'(\mathcal{B}(i)) \leftarrow \max\{c'(\mathcal{B}(i-1)), r(\mathcal{B}(i))\} + p(\mathcal{B}(i))$
  - **forall the** $q \in \mathcal{B}(i)$ **do**
    - $\delta \leftarrow \delta + (w_q \times \max\{0, c(\mathcal{B}(i)) - d_q\}) - (w_q \times \max\{0, c'(\mathcal{B}(i)) - d_q\})$
- $\mathcal{B} = (\ldots, B_{k-1}, B_k, B_{k+1}, \ldots, B_{l-1}, B_l, B_{l+1}, \ldots)$ // undo the move
- **return** $(TWT(\mathcal{B}) - \delta)$

Algorithm 8: Evaluation of a batch merge. The solution evaluated is obtained by merging the batches B_k and B_l, such that $k < l$.

Appendix A. We describe in Appendix B the DP algorithm by Chou and Wang (2008), whose complexity is $\mathcal{O}(n^3)$, and a counterexample that proves that the algorithm does not necessarily yield an optimal solution. An exact DP algorithm for $\text{TWTBSP}_{\text{seq}}$ is provided in Appendix C.

In order to compare the performance of our method described in Sect. 2 with those based on DP for TWTBSP, we implemented the heuristic developed in Wang (2011) and also another two DP-based procedures that are inspired on ILS and BRKGA, respectively. Although the DP algorithm by Chou and Wang (2008) is not exact for $\text{TWTBSP}_{\text{seq}}$, preliminary experiments revealed that its performance in terms of solution quality is similar to the exact DP described in Appendix C. We thus decided to use the DP procedure by Chou and Wang (2008) because our exact DP led to prohibitively large CPU times in practice.

3.1 ILS_{DP}

ILS_{DP} basically consists of embedding the DP algorithm in the proposed $\text{ILS}_{\text{batch}}$ framework by using neighborhoods and perturbations over a sequence of jobs. Given a solution $\mathcal{B} = (B_1, B_2, \ldots, B_m)$, we will assume that the jobs of each batch are in descending order w.r.t. the processing time. Given that B_i^j is the j-th job of the batch B_i,

the job sequence is defined as $\pi = (B_1^1, B_1^2, \ldots, B_1^{|B_1|}, B_2^1, B_2^2, \ldots, B_2^{|B_2|}, \ldots, B_m^1, B_m^2, \ldots, B_m^{|B_m|})$. ILS_{DP} has the following neighborhood structures over π:

- $\text{Insert}_{\text{seq}}$: move a job $\pi_{[j]}$ (j-th element of π) to the i-th position of π, such that $j \neq i$.
- Swap_{seq}: exchange two jobs $\pi_{[i]}$ and $\pi_{[j]}$ in π, such that $i \neq j$.

The two perturbation mechanisms of ILS_{DP} apply random moves (according to the Sect. 2.3) of $\text{insert}_{\text{seq}}$ and swap_{seq}. After each local search or perturbation move, the DP algorithm is called to generate the resulting batching.

3.2 BRKGA_{DP}

BRKGA_{DP} employs the DP procedure as a solution decoder within the well-known BRKGA approach (Gonçalves and Resende 2011). We made use of the BRKGA C++ framework developed by Toso and Resende (2015), where we specified the chromosome decoder procedure. To decode the random keys as feasible solutions, we sort them in ascending order, where the sorted random keys correspond to the sequence of the jobs. We then apply the DP procedure to obtain a new batch scheduling in the genetic algorithm.

4 Computational experiments

The experiments were conducted on an Intel Xeon CPU E5-2650 v4 with 2.20 GHz, 128 GB of RAM, running Ubuntu 16.04 LTS. The algorithms were coded in C++ and compiled with g++ 5.4 and '-O3' flag. $\text{ILS}_{\text{batch}}$ and ILS_{DP} were run with the following parameter values: $I_R = 20$, $I_{ILS} = 4n$, $I_p = 2$, $\alpha = 4$. The first value was taken from Subramanian and Farias (2017), the second from Subramanian et al. (2014); Subramanian and Farias (2017), while the third and fourth parameters were calibrated after experimenting with different values ranging from 1 to 5.

IH was run with the same parameters used by Wang (2011), whereas the default parameters suggested by Toso and Resende (2015) were adopted for BRKGA_{DP}. Finally, we performed long runs of $\text{ILS}_{\text{batch}}$ to obtain upper bounds to be used as reference values while comparing the methods. In this case, a time lime of 1 h was imposed as a stopping criterion as opposed to the number of maximum restarts. Detailed results of $\text{ILS}_{\text{batch}}$ are reported in the Appendix.

4.1 Instances

Because the TWTBSP instances used in previous works are no longer available, we decided to generate a new benchmark dataset to test the performance of the methods. In view of this, we used the procedure described in Chou and Wang (2008) and Mathirajan et al. (2010) to create the instances. It is worth mentioning that Wang (2011) used the approach presented in Mathirajan et al. (2010) to generate the instances for the problem.

Table 2 Distribution of the parameters used by the literature for instance generation

Parameters	Levels (Chou and Wang 2008)	Levels (Mathirajan et al. 2010)
n	5, 7, 9, 11, 13, 15, 50, 100, 150, 200	10, 12, 20, 50, 100
r_j	U(0, 48)	U(1, 20), U(1, 30)
p_j	U(8, 48)	U(1, 10), U(1, 15)
d_j	$r_j + p_j + I(R_1, T)$	$r_j + p_j$ + U(1, 30), U(1, 45)
s_j	U(1, 30) U(15, 35)	U(4, 10), U(4, 14)
w_j	U(1, 11)	s_j
Q	40	20

Both Chou and Wang (2008) and Mathirajan et al. (2010) used a uniform distribution over a predefined interval to define the values of the attributes of the problem.

However, no information was provided by Mathirajan et al. (2010) concerning the weight parameter distribution. We then assumed that the authors determined the weights of each job proportionally to their sizes.

Table 2 presents the distribution of the parameters adopted by Chou and Wang (2008) and Mathirajan et al. (2010).

Note that for Chou and Wang (2008), the distribution of the parameters is mostly similar.

However, there is a considerable difference in the due dates distribution, where in this case the interval is not known *a priori*. Instead, it is based on an expected makespan $C^*_{\max}$ obtained by applying the full batch longest processing time (FBLPT) algorithm by Lee (1999). Once the $C^*_{\max}$ is computed, due dates d_j are distributed along the interval $I(R_1, T) = [\mu(1 - R_1/2), \mu(1 + R_1/2)]$, such that $\mu = (1 - T)(\min\{r_i \mid i \in J\} + C^*_{\max})$. The possible values for T and R_1 are $\{0.3, 0.6\}$ and $\{0.5, 2.5\}$, respectively.

We performed preliminary experiments on several instances generated using the two approaches described above. This allowed us to derive some insights on their difficulty. In fact, we were able to identify the following shortcomings:

- Instances turned out to be sometimes very easy, generally when using the configuration by Chou and Wang (2008). For the instances with up to 15 jobs, it is not uncommon to have optimal solutions with associated TWT of 0, and easy to find even with rather basic local search methods. This happens because the distribution of the due dates often leads to instances with very large values of d_j, which are seldom violated in practice.
- The definition of the release dates also tends to decrease the difficulty of the instances. Although the release dates are uniformly distributed within a predefined interval, and they seem somewhat homogeneously distributed over the scheduling horizon for very small instances, one can verify that they become irrelevant for large size problems.

Table 3 Distribution of the parameters for the new benchmark instances

Parameters	Proposed values
n	10, 15, 20, 25, 30, 35, 50, 75, 100
r_j	U(0, $R_2 \cdot C^*_{\max}$)
p_j	U(1, 30)
d_j	$r_j + p_j$ + U(1, 20)
s_j	U(1, 10), U(5, 15) and U(1,20) if $Q = 20$
	U(1, 20), U(10, 30) and U(1, 40) if $Q = 40$
w_j	U(1, 9)
Q	[20, 40]

The possible values for R_2 are {0.1, 0.4, 0.8}

Table 4 Instance groups and their parameter settings

Group	$Q = 20$	$Q = 40$
G1	$s_j \in [1, 10]$; $R_2 = 0.1$	$s_j \in [1, 20]$; $R_2 = 0.1$
G2	$s_j \in [1, 10]$; $R_2 = 0.4$	$s_j \in [1, 20]$; $R_2 = 0.4$
G3	$s_j \in [1, 10]$; $R_2 = 0.8$	$s_j \in [1, 20]$; $R_2 = 0.8$
G4	$s_j \in [5, 15]$; $R_2 = 0.1$	$s_j \in [10, 30]$; $R_2 = 0.1$
G5	$s_j \in [5, 15]$; $R_2 = 0.4$	$s_j \in [10, 30]$; $R_2 = 0.4$
G6	$s_j \in [5, 15]$; $R_2 = 0.8$	$s_j \in [10, 30]$; $R_2 = 0.8$
G7	$s_j \in [1, 20]$; $R_2 = 0.1$	$s_j \in [1, 40]$; $R_2 = 0.1$
G8	$s_j \in [1, 20]$; $R_2 = 0.4$	$s_j \in [1, 40]$; $R_2 = 0.4$
G9	$s_j \in [1, 20]$; $R_2 = 0.8$	$s_j \in [1, 40]$; $R_2 = 0.8$

Therefore, it was thought advisable to adopt different values for some parameters in order to make the instances more challenging, as presented in Table 3. For example, we adopted three different values for parameter $R_2 \in [0, 1]$ to represent scenarios in which the values of the release date correspond to a small (10%), medium (40%) and large (80%) percentage of the expected makespan $C^*_{\max}$. The idea is to generate distinct scenarios with different levels of difficulty so one can evaluate the impact of the release dates on the algorithm's performance.

We provided 5 instances for each configuration of parameters, leading to a total of 810 instances. For each problem size, i.e., $\forall n \in [10, \ldots, 100]$, and for each capacity value $Q = [20, 40]$, there are 9 groups of 5 instances, with varying settings, as detailed in Table 4. The instances are available at http://professor.ic.ufal.br/rian/Instances/TWTBSP.rar.

Table 5 Impact of adding each neighborhood structure at a time

Setting	Neighborhoods	Avg. improvement	#Improved solutions
(0)	None	–	–
(1)	$\text{insert}_{\text{job}}$	54.38%	162
(2)	(1) + swap_{job}	13.13%	139
(3)	(2) + $\text{insert}_{\text{batch}}$	2.67%	105
(4)	(3) + $\text{swap}_{\text{batch}}$	0.20%	35
(5)	(4) + $\text{merge}_{\text{batch}}$	0.81%	58

4.2 Impact of the neighborhood structures

Table 5 shows the effect of adding each neighborhood structure at a time using the order that they were presented in Sect. 2.2. We consider the first of the 5 instances of each group of the newly proposed benchmark dataset, thus resulting in a total of 162 instances. Five executions were performed for every instance when testing each setting. The first row represents the basic configuration, where no local search is performed, i.e., the algorithm only executes I_R iterations of CH_{EDD}.

The other rows represent the addition of a neighborhood to the subset composed of those considered in the previous rows. The average improvement and the number of improvements are computed w.r.t. the previous configuration, thus measuring the benefits of adding the corresponding neighborhood.

The results suggest that all neighborhoods play a relevant role in helping the algorithm to obtain better solutions. For example, after including neighborhood $\text{insert}_{\text{batch}}$, there is an average improvement of 2.67% on the solution quality, and the best result of 105 instances was improved, when compared to the setting that considers only neighborhoods $\text{insert}_{\text{job}} + \text{swap}_{\text{job}}$.

4.3 Impact of the perturbation mechanisms

Table 6 shows the impact of the two perturbation mechanisms presented in Sect. 2.3. Each setting was tested on the same subset of instances considered in Sect. 4.2. For each setting we computed: (i) the average gap w.r.t. the best solution obtained during the experiment (including all settings); (ii) the number of best solutions found; and (iii) the average CPU time in seconds. In this work, the gap between the objective value of two solutions $\mathcal{B}_1$ and $\mathcal{B}_2$, where $TWT(\mathcal{B}_1) \geq TWT(\mathcal{B}_2)$, is computed as $100 \times (TWT(\mathcal{B}_1) - TWT(\mathcal{B}_2))/TWT(\mathcal{B}_1)$. The results clearly suggest that using both perturbation mechanisms yields a superior performance than the other two settings.

Table 6 Impact of the perturbation mechanisms

Setting	Avg. gap (%)	#Best solutions	Avg. time (s)
Swap of jobs	3.78	81	15.11
Batch split	0.16	146	15.40
Swap of jobs + Batch split	**0.12**	**154**	**15.08**

4.4 Performance comparison

All methods were run 10 times on each instance. However, because of the large CPU times, only a single run was performed for ILS_{DP} on the instances with $|J| \geq 75$, and a time limit of 2 h was imposed.

Tables 7 and 8 present aggregate results obtained by the four algorithms on each group, according to the number of jobs (n), for $Q = 20$ and $Q = 40$, respectively. We report the average (Avg) and minimum (Min) gaps w.r.t. the best known solution (BKS) found after performing long runs of $\text{ILS}_{\text{batch}}$, as well as the CPU time in seconds. In addition, we also provide two additional columns for $\text{ILS}_{\text{batch}}$, reporting the average and minimum gaps w.r.t. the lower bound (LB) provided by an exact algorithm (Pessoa et al. 2020), except for 5 groups involving 100 jobs, in which the LB values were not available.

On average, the results clearly indicate that $\text{ILS}_{\text{batch}}$ was capable of outperforming the other heuristics both in solution quality as well as CPU time. ILS_{DP} obtained the second best average gap, but at the expense of high CPU times, especially for larger instances. BRKGA_{DP}, on the other hand, scaled much better than ILS_{DP}, but at the expense of solution quality, even failing to achieve the best solutions for 15-job instances. Finally, IH presented a very poor performance, even for 10-job instances.

Furthermore, in 46 groups (including both tables), the average gap obtained by $\text{ILS}_{\text{batch}}$ equaled the LB, meaning that the proposed method was capable of systematically finding the optimal solutions for the instances of such groups. This happened, in particular, to all 10- and 15-job instances and to those containing 25 jobs associated with group G9 and $Q = 40$.

Tables 9 and 10 present the number of cases where the BKS (found by performing long runs of $\text{ILS}_{\text{batch}}$) was achieved by each method, for $Q = 20$ and $Q = 40$, respectively. We can observe that $\text{ILS}_{\text{batch}}$ (considering regular runs) found most of the BKSs both for $Q = 20$ and $Q = 40$. Overall, the proposed algorithm managed to find 91.60% of the BKSs, against 68.40%, 59.01%, 8.15% of ILS_{DP}, BRKGA_{DP} and IH, respectively.

The results obtained by $\text{ILS}_{\text{batch}}$ can also be analyzed through Fig. 7, which shows, for every group, the average (a) percentage gap; (b) CPU time in seconds; (c) the number of batches; and (d) number of singletons, which consists of batches composed of a single job. Recall from Table 3 that the job size distribution is equal for the first three groups ([1, 10]). The same happens for groups G4, G5, and G6, whose job distribution is [5, 15], as well as for the last three groups ([1, 20]). From Fig. 7a, we can verify that

Table 7 Aggregate results obtained by ILS_{batch}, ILS_{DP}, $BRKGA_{DP}$ and IH for $Q = 20$

n	Group	ILS_{batch}					ILS_{DP}			$BRKGA_{DP}$			IH		
		Avg (%)	Min (%)	Time (s)	Avg (LB) (%)	Min (LB) (%)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)
10	G1	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.5	0.00	0.00	15.1	0.16	0.00	7.2
	G2	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	15.3	1.47	0.00	7.2
	G3	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.5	0.00	0.00	14.9	7.35	0.00	7.2
	G4	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	14.9	7.74	1.80	7.2
	G5	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	14.3	6.45	2.95	7.2
	G6	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	15.0	25.08	12.50	7.2
	G7	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.5	0.00	0.00	15.8	4.05	3.09	7.1
	G8	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	15.8	7.37	3.71	7.1
	G9	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	14.9	16.01	9.01	7.1
15	G1	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	2.0	0.00	0.00	15.9	21.32	16.29	15.2
	G2	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	2.0	0.00	0.00	15.8	35.18	27.58	15.3
	G3	0.00	0.00	0.1	0.00	0.00	0.00	0.00	1.8	0.00	0.00	15.6	42.48	29.64	15.2
	G4	0.00	0.00	0.1	0.00	0.00	0.00	0.00	1.9	0.00	0.00	15.8	20.39	10.83	15.0
	G5	0.00	0.00	0.1	0.00	0.00	0.03	0.00	1.8	0.31	0.00	16.2	35.40	24.08	15.1
	G6	0.00	0.00	0.1	0.00	0.00	0.00	0.00	1.7	0.07	0.00	16.3	55.15	41.29	15.0
	G7	0.00	0.00	0.1	0.00	0.00	0.00	0.00	1.8	0.00	0.00	16.8	25.85	22.06	15.0
	G8	0.00	0.00	0.1	0.00	0.00	0.00	0.00	1.7	0.00	0.00	16.7	37.67	26.16	15.1
	G9	0.00	0.00	0.1	0.00	0.00	0.00	0.00	1.7	0.08	0.00	16.6	57.89	36.54	15.1

Table 7 continued

n	Group	ILS_{batch}					ILS_{DP}			$BRKGA_{DP}$			IH		
		Avg (%)	Min (%)	Time (s)	Avg (LB) (%)	Min (LB) (%)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)
20	G1	0.00	0.00	0.2	0.00	0.00	0.00	0.00	6.6	0.00	0.00	24.1	38.61	33.06	26.0
	G2	0.00	0.00	0.2	0.49	0.00	0.03	0.00	6.8	0.94	0.00	23.8	47.51	38.46	26.0
	G3	0.00	0.00	0.3	0.00	0.00	0.09	0.00	6.6	0.00	0.00	23.2	64.40	48.95	26.1
	G4	0.00	0.00	0.2	0.71	0.00	0.00	0.00	6.4	0.12	0.00	25.1	28.35	23.60	25.9
	G5	0.00	0.00	0.3	0.71	0.00	0.12	0.00	6.4	1.33	0.00	24.8	50.96	40.38	25.9
	G6	0.00	0.00	0.3	0.00	0.00	0.00	0.00	5.6	0.42	0.00	24.8	75.13	64.59	25.8
	G7	0.00	0.00	0.2	0.80	0.00	0.02	0.00	6.0	0.47	0.00	25.4	37.80	28.73	25.8
	G8	0.00	0.00	0.3	0.57	0.00	0.00	0.00	6.2	0.52	0.00	24.1	48.27	42.52	25.7
	G9	0.00	0.00	0.3	0.00	0.00	0.00	0.00	5.4	0.00	0.00	23.7	74.73	61.39	25.8
25	G1	0.03	0.00	0.3	1.52	0.00	0.48	0.00	16.3	0.71	0.00	26.0	44.49	40.97	39.8
	G2	0.00	0.00	0.4	0.60	0.00	0.15	0.00	15.7	0.21	0.00	26.3	60.59	48.83	39.9
	G3	0.00	0.00	0.5	0.09	0.00	0.13	0.00	14.6	0.18	0.00	26.2	77.08	70.07	39.6
	G4	0.00	0.00	0.5	0.70	0.00	0.00	0.00	17.0	0.96	0.00	27.6	41.84	38.29	39.5
	G5	0.00	0.00	0.6	2.17	0.00	0.12	0.00	12.6	0.33	0.00	28.2	66.16	56.84	39.6
	G6	0.00	0.00	0.6	1.10	0.00	0.00	0.00	13.4	0.74	0.00	28.6	77.20	65.09	39.4
	G7	0.02	0.00	0.5	0.26	0.00	0.03	0.00	14.1	0.53	0.12	29.2	41.19	33.33	39.1
	G8	0.00	0.00	0.5	2.55	0.00	0.45	0.00	11.8	0.74	0.00	29.1	64.89	54.68	39.6
	G9	0.00	0.00	0.6	0.70	0.00	0.12	0.00	12.4	1.69	0.00	30.7	68.16	55.88	39.5

Table 7 continued

n	Group	ILS_{batch}					ILS_{DP}			$BRKGA_{DP}$			IH		
		Avg (%)	Min (%)	Time (s)	Avg (LB) (%)	Min (LB) (%)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)
30	G1	0.00	0.00	0.6	1.80	0.90	0.40	0.08	32.4	0.38	0.19	31.0	52.11	46.06	56.7
	G2	0.02	0.00	0.6	2.15	1.24	0.95	0.05	30.7	0.83	0.17	31.7	61.32	53.09	56.7
	G3	0.08	0.00	0.7	1.34	0.00	0.19	0.00	28.7	0.52	0.00	31.4	76.53	71.82	56.7
	G4	0.00	0.00	0.9	1.04	0.09	0.04	0.00	30.9	0.65	0.17	34.0	47.48	38.62	56.3
	G5	0.00	0.00	0.9	1.71	0.00	0.14	0.00	27.1	2.08	0.02	34.1	61.47	47.29	55.9
	G6	0.00	0.00	1.2	2.66	0.00	0.20	0.00	24.5	2.44	0.00	35.2	82.97	78.01	56.4
	G7	0.00	0.00	0.7	1.49	0.00	0.18	0.00	28.0	1.21	0.20	34.1	49.13	41.76	56.3
	G8	0.00	0.00	1.1	1.90	0.00	0.00	0.00	24.1	1.61	0.23	36.0	63.05	57.95	56.6
	G9	0.00	0.00	1.0	0.79	0.00	0.02	0.00	23.9	0.65	0.00	34.6	78.16	65.59	56.5
35	G1	0.08	0.00	0.7	2.20	0.72	1.33	0.51	66.6	2.13	0.61	37.6	54.65	52.61	76.7
	G2	0.00	0.00	1.0	2.43	0.00	1.90	0.37	62.7	2.38	0.33	37.7	72.08	61.94	76.7
	G3	0.06	0.00	1.3	3.15	0.00	1.41	0.00	58.9	2.16	0.20	36.9	80.96	78.52	76.6
	G4	0.00	0.00	1.4	1.06	0.41	0.53	0.04	68.4	1.46	0.43	38.4	48.31	43.40	75.9
	G5	0.00	0.00	1.9	2.21	0.63	0.38	0.00	58.1	2.17	0.78	39.1	65.82	55.98	76.0
	G6	0.00	0.00	2.1	1.63	0.00	0.08	0.00	48.6	3.67	1.18	37.7	86.44	82.08	76.2
	G7	0.06	0.00	1.3	0.72	0.00	0.21	0.00	55.8	0.79	0.15	37.7	50.82	46.89	76.0
	G8	0.00	0.00	1.7	1.59	0.28	0.59	0.20	51.2	2.27	0.67	40.9	65.70	59.69	76.3
	G9	0.00	0.00	2.2	4.45	0.00	0.15	0.00	45.8	1.63	0.00	39.5	83.54	65.79	76.0

Table 7 continued

n	Group	ILS_{batch}					ILS_{DP}			$BRKGA_{DP}$			IH		
		Avg (%)	Min (%)	Time (s)	Avg (LB) (%)	Min (LB) (%)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)
50	G1	0.28	0.03	3.1	2.14	0.87	2.92	2.37	304.6	4.75	3.58	47.2	60.87	57.84	153.9
	G2	0.32	0.00	3.6	3.43	2.30	3.96	2.11	284.0	6.16	3.88	43.1	72.32	71.00	154.5
	G3	0.05	0.00	4.7	5.86	2.50	1.43	0.15	251.3	12.23	9.20	42.6	88.61	84.96	154.9
	G4	0.00	0.00	5.8	0.81	0.00	0.48	0.02	333.4	3.14	2.13	46.2	55.45	49.29	153.7
	G5	0.04	0.00	6.6	1.52	0.65	1.74	0.18	274.3	6.70	4.34	50.4	72.37	66.94	153.8
	G6	0.00	0.00	7.6	4.82	3.73	1.35	0.19	212.9	9.23	3.17	47.1	89.50	85.22	153.6
	G7	0.04	0.00	5.8	0.97	0.13	0.94	0.08	255.9	2.77	2.07	47.2	56.91	52.26	153.8
	G8	0.06	0.00	6.4	2.10	1.09	1.82	0.75	228.4	5.68	4.78	46.1	74.85	72.13	152.6
	G9	0.25	0.00	8.3	2.67	0.84	2.53	0.71	198.3	10.19	6.39	47.5	90.46	87.67	153.3
75	G1	0.55	0.24	14.6	2.16	1.46	4.65	3.60	1863.5	8.30	6.58	63.8	65.07	62.02	347.5
	G2	0.95	0.41	16.3	5.39	4.73	8.00	5.91	1901.6	15.73	13.96	64.7	80.44	77.85	347.8
	G3	0.50	0.00	22.5	6.36	3.78	7.81	2.58	1479.7	21.95	18.27	63.7	90.96	88.14	346.9
	G4	0.07	0.00	29.2	0.55	0.15	1.65	0.96	2451.1	6.39	5.41	65.8	57.56	55.14	344.5
	G5	0.23	0.00	35.7	1.78	0.91	4.82	2.52	1813.0	13.08	9.86	65.2	77.59	73.59	345.5
	G6	0.32	0.00	42.1	4.51	1.16	6.59	5.83	1249.6	28.70	22.81	66.9	92.03	87.73	344.9
	G7	0.31	0.13	26.3	1.45	0.84	2.15	1.19	1684.7	6.51	5.56	75.2	62.50	60.71	337.6
	G8	0.20	0.05	34.3	2.41	1.54	4.07	2.03	1452.2	10.60	6.58	79.8	76.93	74.51	321.1
	G9	0.18	0.00	36.3	4.35	1.70	3.81	0.86	1139.2	21.57	16.81	79.3	93.24	91.83	325.0

Table 7 continued

n	Group	ILS_{batch}					ILS_{DP}			$BRKGA_{DP}$			IH		
		Avg (%)	Min (%)	Time (s)	Avg (LB) (%)	Min (LB) (%)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)
100	G1	0.99	0.67	41.8	3.04	2.35	4.86	3.06	7078.0	11.30	10.41	93.8	67.20	65.87	620.6
	G2	1.56	0.44	51.2	5.54	4.07	8.83	7.36	6649.1	21.13	19.18	93.6	79.84	78.43	617.9
	G3	1.59	0.73	66.8	12.35	11.58	11.75	6.33	5383.2	40.70	33.62	93.0	93.61	91.69	623.5
	G4	0.16	0.01	89.3	0.84	0.48	1.98	1.57	7200.1	8.78	7.55	101.5	61.43	57.68	621.7
	G5	0.57	0.26	105.1	2.70	1.37	6.92	5.25	7200.1	19.61	15.69	95.0	78.64	71.06	620.2
	G6	0.73	0.44	136.1	6.55	5.26	7.92	6.68	4683.3	43.31	35.63	101.2	93.99	92.72	619.2
	G7	0.36	0.21	86.4	1.21	0.81	1.97	1.20	7090.3	8.19	6.75	104.0	62.24	57.91	621.6
	G8	0.37	0.14	113.7	2.58	2.02	6.64	4.02	5121.8	18.29	13.53	104.0	81.13	76.70	621.9
	G9	0.34	0.00	126.9	7.72	5.29	10.12	5.62	4173.8	41.91	32.88	103.0	94.49	91.76	621.3
Mean		**0.14**	**0.05**	**14.27**			1.63	0.92	900.35	5.51	4.03	41.13	57.59	51.15	148.36

Table 8 Aggregate results obtained by ILS_{batch}, ILS_{DP}, $BRKGA_{DP}$ and IH for $Q = 40$

n	Group	ILS_{batch}					ILS_{DP}			$BRKGA_{DP}$			IH		
		Avg (%)	Min (%)	Time (s) (%)	Avg (LB) (%)	Min (LB) (%)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)
10	G1	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	14.4	0.30	0.00	6.6
	G2	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	14.2	2.78	1.23	6.6
	G3	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	14.4	8.36	3.77	6.6
	G4	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	14.2	4.74	1.44	6.6
	G5	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	13.8	6.61	2.55	6.6
	G6	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	14.3	11.01	5.91	6.6
	G7	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.3	0.00	0.00	15.4	2.42	0.22	6.6
	G8	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	15.1	5.27	1.38	6.6
	G9	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	0.4	0.00	0.00	15.0	17.62	8.85	6.6
15	G1	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	1.9	0.08	0.00	15.6	22.91	16.59	13.9
	G2	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	1.8	0.44	0.00	15.6	31.53	23.45	13.8
	G3	0.00	0.00	0.1	0.00	0.00	0.00	0.00	1.7	0.00	0.00	15.6	49.19	30.38	13.9
	G4	0.00	0.00	0.1	0.00	0.00	0.00	0.00	1.9	0.09	0.00	16.3	24.18	19.59	13.9
	G5	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	1.7	0.00	0.00	16.2	36.99	31.26	13.8
	G6	0.00	0.00	0.1	0.00	0.00	0.00	0.00	1.6	0.00	0.00	16.4	53.91	33.25	13.8
	G7	0.00	0.00	< 0.1	0.00	0.00	0.00	0.00	1.8	0.00	0.00	16.3	24.92	16.71	13.9
	G8	0.00	0.00	0.1	0.00	0.00	0.00	0.00	1.6	0.00	0.00	16.5	30.37	20.66	13.8
	G9	0.00	0.00	0.1	0.00	0.00	0.00	0.00	1.5	0.00	0.00	16.6	64.26	25.25	13.8

Table 8 continued

n	Group	ILS_{batch}					ILS_{DP}			$BRKGA_{DP}$			IH		
		Avg (%)	Min (%)	Time (s) (%)	Avg (LB) (%)	Min (LB) (%)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)
20	G1	0.00	0.00	0.2	0.23	0.00	0.00	0.00	6.2	0.29	0.00	23.6	40.62	34.56	23.8
	G2	0.00	0.00	0.2	0.00	0.00	0.26	0.00	5.7	0.11	0.00	23.1	56.87	50.64	23.7
	G3	0.00	0.00	0.3	0.00	0.00	0.00	0.00	5.6	0.00	0.00	23.2	70.14	59.39	23.9
	G4	0.00	0.00	0.2	0.35	0.00	0.01	0.00	5.9	0.06	0.00	25.2	32.93	27.07	23.7
	G5	0.00	0.00	0.2	0.00	0.00	0.04	0.00	6.0	0.14	0.00	26.3	46.24	38.60	23.7
	G6	0.00	0.00	0.3	0.00	0.00	0.05	0.00	5.2	0.00	0.00	26.5	69.62	62.59	23.7
	G7	0.00	0.00	0.2	0.42	0.00	0.00	0.00	5.4	0.04	0.00	25.9	36.58	31.10	23.7
	G8	0.00	0.00	0.3	0.35	0.00	0.00	0.00	5.0	0.13	0.00	24.7	53.55	44.59	23.7
	G9	0.00	0.00	0.3	0.00	0.00	0.00	0.00	5.0	0.30	0.00	24.0	72.83	69.92	23.7

Table 8 continued

n	Group	ILS_{batch}					ILS_{DP}			$BRKGA_{DP}$			IH		
		Avg (%)	Min (%)	Time (s) (%)	Avg (LB) (%)	Min (LB) (%)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)
25	G1	0.06	0.00	0.3	1.37	0.00	0.26	0.00	14.6	1.89	0.00	25.7	50.75	40.41	36.4
	G2	0.00	0.00	0.4	2.48	0.00	0.32	0.00	14.4	0.28	0.00	25.7	59.58	52.31	36.5
	G3	0.00	0.00	0.5	0.34	0.00	0.00	0.00	13.6	0.68	0.00	25.5	76.96	72.30	36.5
	G4	0.00	0.00	0.5	0.78	0.00	0.02	0.00	14.5	0.18	0.00	27.8	38.56	32.79	36.3
	G5	0.00	0.00	0.4	0.99	0.00	0.00	0.00	14.5	0.57	0.00	27.1	57.27	41.87	36.4
	G6	0.00	0.00	0.6	0.07	0.00	0.04	0.00	12.1	0.12	0.00	30.7	80.06	76.46	36.4
	G7	0.00	0.00	0.4	1.19	0.18	0.00	0.00	13.8	0.62	0.00	30.1	44.51	40.89	36.3
	G8	0.00	0.00	0.6	2.03	0.00	0.12	0.00	13.2	0.34	0.00	30.8	57.68	54.31	36.4
	G9	0.00	0.00	0.6	0.00	0.00	0.00	0.00	11.5	0.00	0.00	29.9	85.22	80.76	36.4
30	G1	0.00	0.00	0.5	1.41	0.00	0.54	0.14	32.2	0.91	0.02	29.8	45.24	40.25	51.9
	G2	0.01	0.00	0.8	2.91	2.26	0.71	0.13	30.1	1.29	0.00	28.8	64.60	57.07	51.9
	G3	0.01	0.00	0.9	3.05	0.84	0.46	0.00	28.7	2.44	0.95	28.7	75.29	66.28	51.8
	G4	0.00	0.00	0.7	0.97	0.00	0.10	0.00	32.6	0.54	0.12	32.8	42.30	38.59	51.7
	G5	0.00	0.00	1.1	0.79	0.00	0.25	0.00	28.3	2.52	1.09	32.6	63.46	57.04	51.7
	G6	0.00	0.00	1.2	1.62	0.00	0.24	0.00	24.1	2.98	1.08	34.1	83.19	76.36	51.8
	G7	0.00	0.00	0.7	1.17	0.44	0.12	0.00	28.2	0.31	0.00	34.4	49.53	45.54	51.7
	G8	0.01	0.00	0.9	1.16	0.04	0.59	0.05	26.6	1.29	0.27	34.6	55.37	42.75	51.7
	G9	0.00	0.00	1.0	4.24	0.52	0.61	0.00	25.6	0.70	0.00	35.0	81.05	73.72	51.7

Table 8 continued

n	Group	ILS_{batch}					ILS_{DP}			$BRKGA_{DP}$			IH		
		Avg (%)	Min (%)	Time (s) (%)	Avg (LB) (%)	Min (LB) (%)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)
35	G1	0.04	0.00	0.8	2.05	0.48	1.33	0.62	63.3	1.80	1.03	35.7	58.26	53.52	70.3
	G2	0.14	0.00	0.9	1.37	0.00	1.05	0.00	60.9	2.37	0.06	35.1	68.55	63.08	70.5
	G3	0.00	0.00	1.3	1.99	0.00	0.81	0.00	63.0	3.19	1.94	33.8	82.03	75.01	70.2
	G4	0.05	0.00	1.6	0.91	0.02	0.12	0.01	73.5	1.02	0.20	36.7	48.45	38.94	69.9
	G5	0.07	0.00	1.8	3.55	0.00	0.48	0.00	65.8	2.53	0.99	36.7	64.29	57.26	69.9
	G6	0.00	0.00	2.0	2.33	0.61	0.70	0.00	50.9	5.35	1.14	37.8	85.24	78.26	69.8
	G7	0.00	0.00	1.2	0.85	0.00	0.08	0.00	58.1	1.27	0.28	37.9	47.54	42.87	70.0
	G8	0.00	0.00	1.7	1.68	0.36	0.40	0.05	64.5	1.74	0.36	37.3	71.07	64.21	70.0
	G9	0.00	0.00	2.0	0.89	0.00	0.11	0.00	52.0	2.15	1.19	38.9	82.33	75.54	69.8
50	G1	0.06	0.00	4.1	1.95	0.13	2.63	1.71	295.4	4.06	3.36	45.6	62.83	56.14	141.1
	G2	0.15	0.00	4.5	2.38	0.35	3.61	1.61	297.4	8.26	6.83	46.4	69.05	64.63	141.0
	G3	0.10	0.00	5.2	3.29	0.98	3.67	0.95	263.2	11.11	8.41	47.0	85.37	79.80	141.1
	G4	0.06	0.00	5.6	0.76	0.51	0.55	0.34	374.0	2.59	1.53	52.1	55.38	50.03	140.6
	G5	0.07	0.00	6.7	1.45	0.30	1.25	0.20	331.2	5.26	3.74	51.9	69.87	61.86	140.6
	G6	0.00	0.00	8.0	2.73	0.81	1.01	0.14	232.2	7.82	3.77	48.9	87.71	85.47	140.7
	G7	0.00	0.00	5.6	0.86	0.00	0.70	0.03	283.5	2.46	1.27	49.4	55.40	48.56	140.7
	G8	0.04	0.00	5.9	1.48	0.00	1.96	0.91	277.2	6.60	5.08	51.9	71.34	65.89	140.7
	G9	0.16	0.00	6.7	9.08	2.90	3.14	0.89	218.9	10.33	8.28	51.9	89.45	85.30	140.8

Table 8 continued

n	Group	ILS_{batch}					ILS_{DP}			$BRKGA_{DP}$			IH		
		Avg (%)	Min (%)	Time (s)	(%) Avg (LB)	(%) Min (LB)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)	Avg (%)	Min (%)	Time (s)
75	G1	0.53	0.31	16.7	2.14	1.83	3.88	3.00	2146.7	7.53	6.56	66.2	58.79	56.07	317.2
	G2	0.82	0.05	20.6	4.16	2.54	7.00	5.86	1940.1	14.65	11.68	61.1	76.18	66.82	317.5
	G3	1.86	0.31	26.4	6.55	1.95	8.31	5.89	1515.8	25.28	21.02	60.4	90.52	88.28	317.4
	G4	0.05	0.00	26.6	0.36	0.20	0.99	0.24	2805.7	6.11	5.70	66.2	58.78	56.55	315.8
	G5	0.08	0.00	35.7	1.07	0.93	4.79	2.67	2131.7	12.74	11.32	67.0	75.39	70.87	316.1
	G6	0.23	0.00	41.9	3.74	0.75	5.40	3.88	1540.4	23.06	19.03	67.7	90.76	86.94	316.0
	G7	0.06	0.01	26.2	0.87	0.55	1.76	1.39	1883.0	5.36	3.86	78.0	59.67	55.29	316.4
	G8	0.21	0.01	37.7	1.68	0.67	3.99	2.54	1528.8	12.55	9.18	80.5	78.34	74.35	316.2
	G9	0.07	0.00	40.1	3.74	2.94	2.68	0.77	1263.5	18.88	12.24	75.0	91.11	90.15	316.4
100	G1	0.54	0.25	54.7	–	–	5.91	4.82	6413.2	10.94	10.00	89.8	64.19	61.53	571.0
	G2	1.00	0.59	68.5	–	–	9.20	4.14	5750.7	19.24	15.68	89.2	76.52	72.69	571.1
	G3	1.05	0.50	86.5	–	–	12.07	7.59	4676.2	36.41	31.63	89.5	90.98	89.35	570.7
	G4	0.09	0.03	95.9	0.53	0.33	1.56	0.85	7200.1	8.31	5.91	97.3	60.00	57.58	568.7
	G5	0.41	0.10	128.0	1.86	0.54	4.85	2.98	7026.9	17.52	15.22	99.9	77.73	73.69	568.8
	G6	0.23	0.09	140.2	6.30	3.90	7.20	5.16	4625.8	37.99	33.05	99.6	92.17	88.54	568.8
	G7	0.25	0.03	100.5	1.09	0.73	2.61	1.30	5732.1	7.73	5.79	104.5	62.54	59.73	569.8
	G8	0.70	0.07	106.0	–	–	5.67	3.62	4807.3	18.69	13.45	103.1	81.36	78.94	569.1
	G9	0.45	0.00	133.1	–	–	4.25	0.55	3691.0	40.19	33.17	99.4	93.95	90.77	569.1
Mean		**0.12**	**0.03**	**15.63**			1.49	0.80	867.30	5.22	3.92	40.88	57.04	50.80	136.60

Table 9 Number of best known solutions found by each method for $Q = 20$

$\|J\|$	Method			
	ILS_{batch}	ILS_{DP}	$BRKGA_{DP}$	IH
10	45	45	45	28
15	45	45	45	0
20	45	45	45	0
25	45	44	42	0
30	45	44	37	0
35	45	41	30	0
50	44	17	1	0
75	35	0	0	0
100	19	0	0	0
Total	**368**	281	245	28

Table 10 Number of best known solutions found by each method for $Q = 40$

$\|J\|$	Method			
	ILS_{batch}	ILS_{DP}	$BRKGA_{DP}$	IH
10	45	45	45	38
15	45	45	45	0
20	45	45	45	0
25	45	45	44	0
30	45	39	30	0
35	45	41	24	0
50	44	13	0	0
75	34	0	0	0
100	26	0	0	0
Total	**374**	273	233	38

the average gap tends to be larger for groups G1, G2, G3, where the average job size is smaller. Also, note that the number of singletons increases with the job size, as can be observed in Fig. 7d. This result strengthens the hypothesis that groups with smaller average job size (i.e., G1, G2, and G3) are more difficult since the batching decisions seem to become challenging as the job sizes decrease, leading to a few number of singletons. On the other hand, the key difference between the instances whose groups have the same job size distribution is the release date values, which assume increasing values according to the group. In other words, G1, G4, and G7 are the groups with smaller release date values, whereas G3, G6, and G9 are the groups with larger release

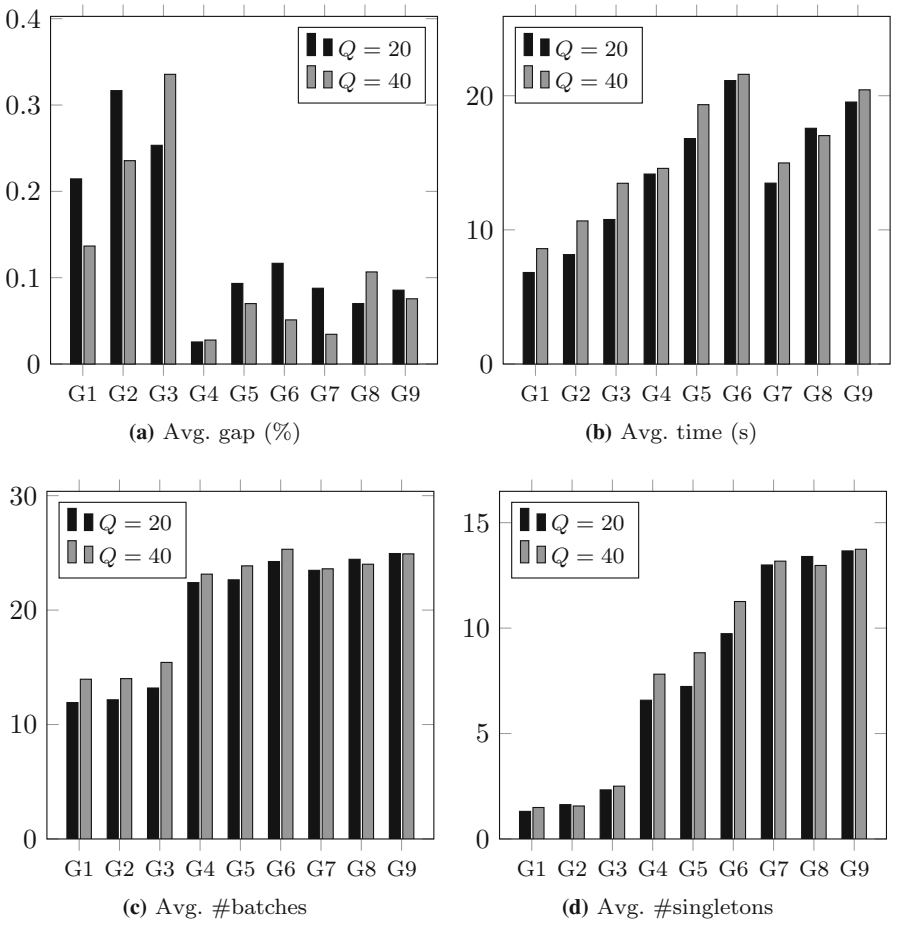


Fig. 7 Results of $\text{ILS}_{\text{batch}}$ by group. Four distinct metrics are considered for analysis

date values. Figure 7b suggests that the CPU time is likely to increase with the job size and release date values. This can be explained by the average number of batches that also follows the same behavior (see Fig. 7c), thus directly affecting the runtime performance because the smaller the number of batches, the faster the local search and vice-versa. Indeed, we can see a clear correlation between Fig. 7b and c.

5 Concluding remarks

In this paper, we addressed the single machine total weighted tardiness batch scheduling problem (TWTBSP). In order to solve the problem, we designed a metaheuristic algorithm based on iterated local search (ILS) that is composed of five neighborhood structures and two perturbation mechanisms. Also, three alternative dynamic programming (DP)-based procedures were implemented to serve as a reference for

measuring the efficiency of the proposed algorithm. One of them is a heuristic developed by Wang (2011), whereas the other two are based on biased random key genetic algorithm (BRKGA) and ILS, respectively. Moreover, we showed that the problem of finding the optimal TWT batching given a fixed sequence of jobs is $\mathcal{NP}$-hard and provided an exact pseudo-polynomial DP for the problem. However, the existing non-exact DP procedure scales much better than the exact one without degrading the solution quality, and it was therefore used in the DP-based heuristics.

Extensive computational experiments on newly proposed benchmark instances showed that the results obtained by our heuristic, called $\text{ILS}_{\text{batch}}$, outperformed the other ones both in solution quality (by a far margin) and CPU time. The results suggest that, despite its convenience and popular use in the batch scheduling literature, DP-based procedures may not efficiently scale to larger instances. Furthermore, $\text{ILS}_{\text{batch}}$ was compared to lower bounds made available by Pessoa et al. (2020), where it was observed that our algorithm was capable of finding all the known optima and producing solutions with relatively small gaps, even for instances with up to 100 jobs. Finally, it was also shown that the instances tend to become difficult as the average job size decreases and that the CPU time increases as the average job size and release date values increase. The first happens because the smaller the average job size, the smaller the number of batches (thus fewer singletons), and therefore the batching decisions become more intricate. The second occurs because the average number of batches increases with the average job size and thus the local search phase becomes more time consuming, as the complexity of some of the neighborhoods increases with the number of batches.

Future research includes the development of a hybrid approach combining the advantages of $\text{ILS}_{\text{batch}}$ with an exact method. For example, one can use the information found during the heuristic search to feed an integer programming-based algorithm to find improved solutions. Finally, one can also extend the proposed algorithm to solve related problems, such as the one with incompatible job families (Balasubramanian et al. 2004; Perez et al. 2005).

Acknowledgements This research was partially supported by the Brazilian research agencies CNPq, Grants 428549/2016-0, 307843/2018-1, 306033/2019-4 and by CAPES—Finance Code 001.

Appendix A : TWTBSP with fixed sequence of jobs

SINGLE MACHINE TOTAL WEIGHTED TARDINESS BATCH SCHEDULING WITH FIXED SEQUENCE OF JOBS ($\text{TWTBSP}_{\text{SEQ}}$)
Instance: A sequence of jobs $\pi = (\pi_{[1]}, \pi_{[2]}, \ldots, \pi_{[n]})$
Objective: Find a batching $\mathcal{B} = (B_1, B_2, \ldots, B_m)$ which minimizes the TWT and is feasible for the TWTBSP, such that $\forall i, j \in \{1, 2, \ldots, n\}, i < j$, the batch associated with $\pi_{[i]}$ must be processed before the one associated with $\pi_{[j]}$, unless they have been assigned to the same batch.

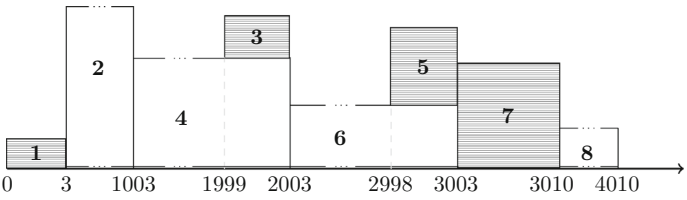


Fig. 8 A certificate for a TWTBSDP$_{\text{seq}}$ instance built from a SSP instance with $S = \{3, 4, 5, 7\}$, and $V = 10$

The decision version of TWTBSP$_{\text{seq}}$ (or TWTBSDP$_{\text{seq}}$ for short) includes the maximum weighted tardiness TWT$_{\max}$ as an additional data, and asks whether a feasible TWTBSP$_{\text{seq}}$ batching with TWT $\leq$ TWT$_{\max}$ exists or not. The next proposition proves the hardness of TWTBSP$_{\text{seq}}$ by showing that its decision version is $\mathcal{NP}$-Complete.

Proposition 1 *The decision version of TWTBSP$_{seq}$ is $\mathcal{NP}$-Complete, even if $r_j = 0$, for $j = 1, \ldots, n$.*

Proof Clearly the TWTBSDP$_{\text{seq}}$ belongs to $\mathcal{NP}$ (since a certificate for the positive answer is easily verified in polynomial time). It remains to prove its completeness, which we show through a reduction from the Subset Sum Problem (SSP).

Given a set $S = \{v_1, \ldots, v_{\bar{n}}\}$ of positive integers and a target positive integer V, the SSP asks whether a subset S' of S whose elements sum exactly V exists or not. For the sake of simplicity, we assume without loss of generality in this proof that $\pi_{[j]} = j$, for $j = 1, \ldots, n$. Let also M and M' be sufficiently large numbers whose values we define later, with $M > M'$, and $\bar{V} = \sum_{j=1}^{\bar{n}} v_j - V$.

For a given instance of the SSP, we show next how to build a corresponding instance for the TWTBSDP$_{\text{seq}}$ such that $r_j = 0$, for $j = 1, \ldots, n$:

- $n = 2\bar{n}$, $Q = \bar{n} + 1$, and TWT$_{\max} = M\bar{V} + M'$,
- $p_{2j-1} = v_j$, $d_{2j-1} = (j-1)M$, $w_{2j-1} = v_j$, and $s_{2j-1} = j$, for $j = 1, \ldots, \bar{n}$,
- $p_{2j} = M$, $d_{2j} = w_{2j} = 0$, and $s_{2j} = \bar{n} - j + 1$, for $j = 1, \ldots, \bar{n} - 1$,
- $p_n = M$, $d_n = Mn + V$, $w_n =$ TWT$_{\max} + 1$, and $s_n = 1$.

Jobs with odd indices are referred to as the *short* ones and those with even indices as the *long* ones.

For example, consider a SSP instance with $S = \{3, 4, 5, 7\}$, and $V = 10$, where $v_1 = 3$ and $v_4 = 7$ compose a certificate for the positive answer. Let us also assume that $M = 1000$. In this case, the corresponding TWTBSDP$_{\text{seq}}$ instance would have $p_1 = 3$, $p_3 = 4$, $p_5 = 5$, $p_7 = 7$, and the remaining processing times equal to 1000. Moreover, $s_1, s_2, s_3, s_4, s_5, s_6, s_7$, and s_8 would be equal to 1, 4, 2, 3, 3, 2, 4, and 1, respectively, and $Q = 5$. Figure 8 represents a certificate for the positive answer to this instance, where short jobs are shadowed. Note that the numbers v_1 and v_4 selected in the SSP certificate correspond to the short jobs that belong to unitary batches.

In general, given a certificate S' for the SSP instance, the batches of a certificate for the corresponding TWTBSDP$_{\text{seq}}$ instance are constructed as follows. Start with

an empty $\mathcal{B}$. Then, for $j = 1, \ldots, \bar{n}$, if $v_j \in S'$, add two unitary batches $\{2j-1\}$ and $\{2j\}$ to $\mathcal{B}$. Otherwise, if $v_j \notin S'$, add a single batch $\{2j-1, 2j\}$ to $\mathcal{B}$.

We prove now that (by setting appropriate values for M and M') this is a valid certificate for the TWTBSDP$_{\text{seq}}$ instance. First, note that, since there are exactly $\bar{n}$ long jobs and the processing times of all short jobs that are alone in their batches sum up to V, the completion time of job n is exactly $d_n = M\bar{n} + V$. Moreover, for all j such that $v_j \in S'$, its completion time is exactly $(j-1)M + \sum_{\substack{k=1 \\ v_k \in S'}}^{j} v_k$, contributing with $v_j \sum_{\substack{k=1 \\ v_k \in S'}}^{j} v_k$ to the TWT. Furthermore, for all j such that $v_j \in S \setminus S'$, its completion time is exactly $jM + \sum_{\substack{k=1 \\ v_k \in S'}}^{j-1} v_k$, contributing with $v_j \left(M + \sum_{\substack{k=1 \\ v_k \in S'}}^{j-1} v_k \right)$ to the TWT. Then, by setting $M' = \left(\sum_{k=1}^{\bar{n}} v_k \right)^2$, we make sure that the sum of all TWT terms that do not depend on M does not exceed M'. Clearly, the TWT terms that depend on M sum exactly $M\bar{V}$, which results in a TWT not greater then TWT$_{\max}$.

It remains to prove that a valid certificate for the TWTBSDP$_{\text{seq}}$ instance also implies a positive answer to the SSP instance. To see this, note that the jobs $2j$ and $2j+1$, for $j = 1, \ldots, \bar{n}-1$, cannot belong to the same batch due to the capacity constraints. Thus, it only remains the choices of whether or not to join the jobs $2j-1$ and $2j$ into a single batch, for $j = 1, \ldots, \bar{n}$.

Now, consider that set S' built as follows: for all $j \in \{1, \ldots, \bar{n}\}$ such that jobs $2j-1$ and $2j$ do not form a batch, add v_j to S'. We argue that $\sum_{j \in S'} v_j = V$. First, if we had $\sum_{j \in S'} v_j > V$, this would lead to a completion time greater than d_n for job n, forcing the TWT to be at least TWT$_{\max} + 1$. Moreover, if we had $\sum_{v \in S'} v < V$, and thus $\sum_{v \in S \setminus S'} v > \bar{V}$, the contribution of the short jobs to the TWT would not be smaller than $M(\bar{V}+1)$ which exceeds TWT$_{\max}$, assuming that $M > M'$.

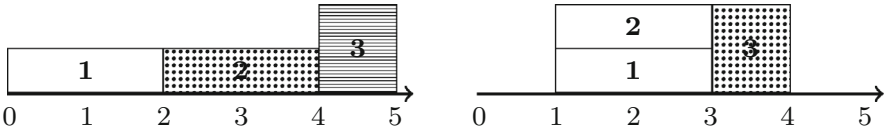
To complete this proof, we observe that setting $M = M' + 1$ satisfies all the assumptions made so far on the value of M. □

Appendix B : DP proposed by Chou and Wang (2008)

Given a sequence of jobs $\pi = (\pi_{[1]}, \pi_{[2]}, \ldots, \pi_{[k]}, \ldots, \pi_{[n]})$, we define $\pi^k = (\pi_{[1]}, \pi_{[2]}, \ldots, \pi_{[k]})$ as a subset of π containing the first k jobs, and $\pi^k_j = (\pi_{[k-j+1]}, \ldots, \pi_{[k]})$ as a subset of π^k containing the j jobs from position $k-j+1$ to position k. The minimum TWT for π^k is given by $TWT(\pi^k) = \min_{1 \le j \le k}\{f(\pi^k, j)\}$

Job	p_j	d_j	s_j	w_j	r_j
1	2	2	1	1	0
2	2	4	1	1	1
3	1	4	2	2	0
$Q = 2$					

(a) Instance



(b) DP solution with TWT=2 **(c)** Optimal solution with TWT=1

Fig. 9 Couterexample showing that the DP algorithm by Chou and Wang (2008) is not exact for the $\text{TWTBSP}_{\text{seq}}$. The input is the job sequence $\pi = (1, 2, 3)$

such that

$$f(\pi^k, j) = \begin{cases} TWT(\pi^{k-j}) + \sum_{i \in \pi_j^k} w_i \mathcal{T}_{ij}, & \text{if } \sum_{i \in \pi_j^k} q_i \leq Q \\ \infty, & \text{otherwise} \end{cases}$$

where $\mathcal{T}_{ij} = \max(0, g(\pi^k, j) - d_i)$. Let $\gamma(\pi^k) = \arg\min_{1 \leq j \leq k}\{f(\pi^k, j)\}$ be the index which defines the last batch of π^k as $(\pi_{[k-\gamma(\pi^k)+1]}, \ldots, \pi_{[k]})$. The recursive function $g(\pi^k, j) = \max_{i \in \pi_j^k}\{g(\pi^{k-j}, \gamma(\pi^{k-j})), r_i\} + \max_{i \in \pi_j^k} p_i$ returns the completion time of the last batch of π^k assuming that it is defined as $(\pi_{[j]}, \ldots, \pi_{[k]})$. In addition, we have $\pi^0 = \emptyset$, $\gamma(\emptyset) = \emptyset$, $g(\emptyset, \emptyset) = 0$, and $TWT(\emptyset) = 0$. To recover the solution, it is necessary to build the batches from right to left using γ. For example, if $l_1 = \gamma(\pi^n)$, the last batch is $(\pi_{[n-l_1+1]}, \ldots, \pi_{[n]})$. Given that $l_2 = l_1 + \gamma(\pi^{n-l_1})$, the last but one batch is $(\pi_{[n-l_2+1]}, \ldots, \pi_{[n-l_1]})$. The process of finding $l_k = l_{k-1} + \gamma(\pi^{n-l_{k-1}})$ is repeated until the first batch is built.

Figure 9 shows a 3-job instance for which the DP algorithm fails to find the optimal batching over the sequence $\pi = (1, 2, 3)$. This is due to the fact that solving a subproblem π^k optimally ($k < n$) can be seen as a greedy decision because it does not guarantee that the completion time of the last batch of π^k is minimal. In the example, the DP procedure places jobs 1 and 2 in different batches, where $TWT(\pi^2) = 0$ and $g(\pi^2) = 4$. Thus, $TWT(\pi^3) = 2$ because the job 3 is forced to have weighted tardiness of 2 (the solution where 1 and 2 are in the same batch with completion time 3 is not evaluated). Table 11 shows the complete execution of the algorithm. Let $l_1 = \gamma(\pi^3) = 1$, we have that the third batch is $(\pi_{[3-l_1+1]}, \ldots, \pi_{[3]}) = (\pi_{[3-1+1]}, \ldots, \pi_{[3]}) = (\pi_{[3]})$; let $l_2 = l_1 + \gamma(\pi^{3-l_1}) = 1 + \gamma(\pi^2) = 2$, the second batch is $(\pi_{[3-l_2+1]}, \ldots, \pi_{[3-l_1]}) = (\pi_{[3-2+1]}, \ldots, \pi_{[2]}) = (\pi_{[2]})$; and let $l_3 = l_2 + \gamma(\pi^{3-l_2}) = 2 + \gamma(\pi^1) = 3$, the first batch is $(\pi_{[3-l_3+1]}, \ldots, \pi_{[3-l_2]}) = (\pi_{[3-3+1]}, \ldots, \pi_{[1]}) = (\pi_{[1]})$.

Table 11 Execution of the DP algorithm by Chou and Wang (2008) over the instance in Fig. 9

k	π^k	j	$f(\pi^k, j)$	$g(\pi^k, j)$	$\gamma(\pi^k)$	$TWT(\pi^k)$
1	(1)	1	$TWT(\pi^0) + w_1\mathcal{T}_{11} =$ $= 0 + \max(0, g(\pi^1, 1) - d_1) =$ $= \max(0, 0) = 0$	$\max(g(\pi^0, \gamma(\pi^0)), r_1)$ $+p_1 = r_1 + p_1 = 2$	1	0
2	(1, 2)	1	$TWT(\pi^1) + w_2\mathcal{T}_{21} =$ $= 0 + \max(0, g(\pi^2, 1) - d_2) =$ $= \max(0, 4 - 4) = 0$	$\max(g(\pi^1, \gamma(\pi^1)), r_2)$ $+p_2$ $= \max(2, 0) + 2 = 4$	1	0
		2	$TWT(\pi^0) + w_1\mathcal{T}_{12} + w_2\mathcal{T}_{22} =$ $= 0 + \max(0, g(\pi^2, 2) - d_1) +$ $\max(0, g(\pi^2, 2) - d_2) = 1$	$\max(g(\emptyset, \emptyset), r_2) +$ $\max(p_1, p_2) =$ $= 1 + 2 = 3$		
3	(1, 2, 3)	1	$TWT(\pi^2) + w_3\mathcal{T}_{31}$ $= 0 + 2\max(0, g(\pi^3, 1) - d_3) =$ $= 2\max(0, 5 - 4) = 2$	$\max(g(\pi^2, \gamma(\pi^2)), r_3)$ $+p_3 =$ $= \max(4, 0) + 1 = 5$	1	2
		2	∞ $(\sum_{i \in \pi_j^k} q_i > Q)$	–		
		3	∞ $(\sum_{i \in \pi_j^k} q_i > Q)$	–		

Appendix C: Exact DP algorithm for the TWTBSP$_{\text{seq}}$

In this appendix, we describe an exact dynamic programming approach for TWTBSP$_{\text{seq}}$. Given a sequence of jobs $\pi = (\pi_{[1]}, \pi_{[2]}, \ldots, \pi_{[k]}, \ldots, \pi_{[n]})$, the algorithm determines the values $TWT^r_{i,j}$ for $1 \leq i < j \leq n$ and $r \geq 0$. Each value $TWT^r_{i,j}$ is defined as the total weighted tardiness found by the DP approach for TWTBSP$_{\text{seq}}$ on an instance $\{\pi_{[i]}, \ldots, \pi_{[j]}\} \subseteq \pi$ starting with a release date r. If $TWT^{r'}_{i',j}$ is known for all $i < i'$ and $r < r'$, then we can consider each possible initial batch B' containing jobs $\{\pi_{[i]}, \ldots, \pi_{[k]}\}$ and compute the corresponding solution $TWT^r_{i,j}$ using the following recursive formula:

$$TWT^r_{i,j} = \min_{i \leq k \leq j} \left(w(i,k) + TWT^{C^r_{i,k}}_{k+1,j} \right) \tag{1}$$

where $C^r_{i,k}$ is the completion time of the batch B' with release date r and

$$w(i,k) = \begin{cases} \sum_{\rho=i}^{k} w_\rho T_\rho & \text{if } \sum_{\rho=i}^{k} s_\rho \leq Q \\ w(i,k) = \infty & \text{otherwise.} \end{cases}$$

Note that if B' is a feasible batch, then $w(i, j)$ represents the weighted tardiness of B', and when $k = j$, the procedure sets $TWT^{C^r_{i,k}}_{k+1,j} = 0$.

Proposition 2 *Given sequence of jobs $\pi = (\pi_{[i]}, \pi_{[i+1]}, \ldots, \pi_{[j]})$. The DP algorithm finds an optimal batch partitioning for the TWTBSP in $\mathcal{O}(n^3 \max(r_{\max}, p_{sum}))$ pseudo-polynomial time, where $r_{\max}$ is the maximum release date for all jobs in π and $p_{sum} = \sum_{k=i}^{j} p_k$.*

Proof The correctness of the algorithm follows directly by induction on $j - i$. When $j - i = 1$, we have only one job in a single batch, and thus $TWT^r_{i,j} = w(i, j) = OPT(i, j, r)$. Now consider arbitrary values of i, j and r, and suppose the statement is true for all pair (i', j') with $j' - i' < j - i$.

Let $\mathcal{B} = \{B_1, \ldots, B_\eta\}$ be an optimal solution with total weighted tardiness $OPT(i, j, r)$ and initial batch $B_1 = \{\pi_{[i]}, \ldots, \pi_{[\ell]}\}$ with completion time $C^r_{i,\ell}$.

The algorithm uses recurrence (1) to compute $TWT^r_{i,j}$ by minimizing

$$TWT^r_{i,j} = \min_{i \leq k \leq j} \left(w(i,k) + TWT^{C^r_{i,k}}_{k+1,j} \right) \leq \left(w(i,\ell) + TWT^{C^r_{i,\ell}}_{\ell+1,j} \right)$$

Since $j - (\ell + 1) < j - i$, and using the induction hypothesis, we know that $TWT^{C^r_{i,\ell}}_{\ell+1,j} = OPT(\ell + 1, j, C^r_{i,\ell})$. Therefore, we have that $TWT^r_{i,j} \leq w(i, \ell) + OPT(\ell + 1, j, C^r_{i,\ell}) = OPT(i, j, r)$, and thus $TWT^r_{i,j}$ is an optimal value.

Let $r_{\max} = \max_{k=1}^{j} r_k$ and $p_{sum} = \sum_{k=i}^{j} p_k$. The proposed dynamic programming can be solved in $\mathcal{O}(n^3 \max(r_{\max}, p_{sum}))$. □

It is obvious that by removing the first batch B_1 from the optimal partitioning, the remaining solution set must be an optimal solution to the subproblem defined by release date $r = C(B_1)$ and jobs set $\{\pi_{[|B_1|+1]}, \ldots, \pi_{[n]}\}$. Any other choice may increase the optimal solution value. Hence, $\text{TWTBSP}_{\text{seq}}$ has the property of an optimal substructure needed for applying a dynamic programming approach.

Appendix D : Detailed results obtained by $\text{ILS}_{\text{batch}}$

This appendix reports the detailed results found by $\text{ILS}_{\text{batch}}$ on each individual instance. More precisely, we report the minimum (Min) and average (Avg) TWT values among the 10 runs, as well as the CPU time in seconds. Moreover, we also report the best known solution (BKS) found after performing a long run of the algorithm, in which a time limit of 1 h was imposed as a stopping criterion, instead of the maximum number of restarts. Finally, we provide the lower bounds (LBs) made available in Pessoa et al. (2020). Bold values indicate that the optimal solution was found, whereas underlined values indicate that the BKS is better than the minimum value found during the regular runs (Tables 12 and 13).

Table 12 Detailed results of ILS_{batch} with $Q = 20$

Instance	LB	Min	Avg	Time (s)	BKS
Q20_10J_G1_1	**404.00**	**404**	404.00	0.04	**404**
Q20_10J_G1_2	**186.00**	**186**	186.00	0.02	**186**
Q20_10J_G1_3	**209.00**	**209**	209.00	0.03	**209**
Q20_10J_G1_4	**186.00**	**186**	186.00	0.02	**186**
Q20_10J_G1_5	**448.00**	**448**	448.00	0.03	**448**
Q20_10J_G2_1	**382.00**	**382**	382.00	0.04	**382**
Q20_10J_G2_2	**384.00**	**384**	384.00	0.06	**384**
Q20_10J_G2_3	**486.00**	**486**	486.00	0.04	**486**
Q20_10J_G2_4	**545.00**	**545**	545.00	0.05	**545**
Q20_10J_G2_5	**207.00**	**207**	207.00	0.02	**207**
Q20_10J_G3_1	**249.00**	**249**	249.00	0.04	**249**
Q20_10J_G3_2	**66.00**	**66**	66.00	0.04	**66**
Q20_10J_G3_3	**37.00**	**37**	37.00	0.04	**37**
Q20_10J_G3_4	**177.00**	**177**	177.00	0.03	**177**
Q20_10J_G3_5	**87.00**	**87**	87.00	0.04	**87**
Q20_10J_G4_1	**481.00**	**481**	481.00	0.04	**481**
Q20_10J_G4_2	**1432.00**	**1432**	1432.00	0.03	**1432**
Q20_10J_G4_3	**257.00**	**257**	257.00	0.02	**257**
Q20_10J_G4_4	**368.00**	**368**	368.00	0.03	**368**
Q20_10J_G4_5	**330.00**	**330**	330.00	0.04	**330**

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_10J_G5_1	**308.00**	**308**	308.00	0.04	**308**
Q20_10J_G5_2	**463.00**	**463**	463.00	0.05	**463**
Q20_10J_G5_3	**389.00**	**389**	389.00	0.04	**389**
Q20_10J_G5_4	**476.00**	**476**	476.00	0.03	**476**
Q20_10J_G5_5	**386.00**	**386**	386.00	0.03	**386**
Q20_10J_G6_1	**850.00**	**850**	850.00	0.03	**850**
Q20_10J_G6_2	**289.00**	**289**	289.00	0.05	**289**
Q20_10J_G6_3	**271.00**	**271**	271.00	0.05	**271**
Q20_10J_G6_4	**126.00**	**126**	126.00	0.04	**126**
Q20_10J_G6_5	**302.00**	**302**	302.00	0.03	**302**
Q20_10J_G7_1	**1234.00**	**1234**	1234.00	0.03	**1234**
Q20_10J_G7_2	**422.00**	**422**	422.00	0.04	**422**
Q20_10J_G7_3	**401.00**	**401**	401.00	0.04	**401**
Q20_10J_G7_4	**882.00**	**882**	882.00	0.02	**882**
Q20_10J_G7_5	**621.00**	**621**	621.00	0.04	**621**
Q20_10J_G8_1	**366.00**	**366**	366.00	0.03	**366**
Q20_10J_G8_2	**125.00**	**125**	125.00	0.03	**125**
Q20_10J_G8_3	**531.00**	**531**	531.00	0.03	**531**
Q20_10J_G8_4	**281.00**	**281**	281.00	0.05	**281**

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_10J_G8_5	**376.00**	**376**	376.00	0.03	**376**
Q20_10J_G9_1	**504.00**	**504**	504.00	0.04	**504**
Q20_10J_G9_2	**526.00**	**526**	526.00	0.05	**526**
Q20_10J_G9_3	**653.00**	**653**	653.00	0.03	**653**
Q20_10J_G9_4	**373.00**	**373**	373.00	0.03	**373**
Q20_10J_G9_5	**237.00**	**237**	237.00	0.04	**237**
Q20_15J_G1_1	**774.00**	**774**	774.00	0.08	**774**
Q20_15J_G1_2	**785.00**	**785**	785.00	0.08	**785**
Q20_15J_G1_3	**417.00**	**417**	417.00	0.09	**417**
Q20_15J_G1_4	**1236.00**	**1236**	1236.00	0.08	**1236**
Q20_15J_G1_5	**764.00**	**764**	764.00	0.08	**764**
Q20_15J_G2_1	**408.00**	**408**	408.00	0.11	**408**
Q20_15J_G2_2	**554.00**	**554**	554.00	0.11	**554**
Q20_15J_G2_3	**575.00**	**575**	575.00	0.08	**575**
Q20_15J_G2_4	**262.00**	**262**	262.00	0.09	**262**
Q20_15J_G2_5	**549.00**	**549**	549.00	0.08	**549**
Q20_15J_G3_1	**406.00**	**406**	406.00	0.13	**406**
Q20_15J_G3_2	**390.00**	**390**	390.00	0.14	**390**
Q20_15J_G3_3	**443.00**	**443**	443.00	0.09	**443**

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_15J_G3_4	**781.00**	**781**	781.00	0.11	**781**
Q20_15J_G3_5	**592.00**	**592**	592.00	0.08	**592**
Q20_15J_G4_1	**1567.00**	**1567**	1567.00	0.09	**1567**
Q20_15J_G4_2	**1714.00**	**1714**	1714.00	0.11	**1714**
Q20_15J_G4_3	**1164.00**	**1164**	1164.00	0.11	**1164**
Q20_15J_G4_4	**1685.00**	**1685**	1685.00	0.13	**1685**
Q20_15J_G4_5	**2448.00**	**2448**	2448.00	0.12	**2448**
Q20_15J_G5_1	**872.00**	**872**	872.00	0.13	**872**
Q20_15J_G5_2	**1564.00**	**1564**	1564.00	0.11	**1564**
Q20_15J_G5_3	**1268.00**	**1268**	1268.00	0.13	**1268**
Q20_15J_G5_4	**440.00**	**440**	440.00	0.12	**440**
Q20_15J_G5_5	**1805.00**	**1805**	1805.00	0.10	**1805**
Q20_15J_G6_1	**586.00**	**586**	586.00	0.13	**586**
Q20_15J_G6_2	**763.00**	**763**	763.00	0.14	**763**
Q20_15J_G6_3	**707.00**	**707**	707.00	0.11	**707**
Q20_15J_G6_4	**692.00**	**692**	692.00	0.13	**692**
Q20_15J_G6_5	**598.00**	**598**	598.00	0.10	**598**

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_15J_G7_1	**1847.00**	**1847**	1847.00	0.10	**1847**
Q20_15J_G7_2	**1841.00**	**1841**	1841.00	0.12	**1841**
Q20_15J_G7_3	**1742.00**	**1742**	1742.00	0.09	**1742**
Q20_15J_G7_4	**1450.00**	**1450**	1450.00	0.11	**1450**
Q20_15J_G7_5	**1949.00**	**1949**	1949.00	0.10	**1949**
Q20_15J_G8_1	**887.00**	**887**	887.00	0.09	**887**
Q20_15J_G8_2	**2219.00**	**2219**	2219.00	0.14	**2219**
Q20_15J_G8_3	**898.00**	**898**	898.00	0.12	**898**
Q20_15J_G8_4	**1317.00**	**1317**	1317.00	0.11	**1317**
Q20_15J_G8_5	**1402.00**	**1402**	1402.00	0.11	**1402**
Q20_15J_G9_1	**275.00**	**275**	275.00	0.10	**275**
Q20_15J_G9_2	**346.00**	**346**	346.00	0.10	**346**
Q20_15J_G9_3	**907.00**	**907**	907.00	0.10	**907**
Q20_15J_G9_4	**636.00**	**636**	636.00	0.11	**636**
Q20_15J_G9_5	**1110.00**	**1110**	1110.00	0.11	**1110**
Q20_20J_G1_1	**1547.00**	**1547**	1547.00	0.17	**1547**
Q20_20J_G1_2	**2268.00**	**2268**	2268.00	0.19	**2268**
Q20_20J_G1_3	**1499.00**	**1499**	1499.00	0.15	**1499**

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_20J_G1_4	**1532.00**	**1532**	1532.00	0.17	**1532**
Q20_20J_G1_5	**1164.00**	**1164**	1164.00	0.14	**1164**
Q20_20J_G2_1	**1078.00**	**1078**	1078.00	0.20	**1078**
Q20_20J_G2_2	**652.00**	**652**	652.00	0.24	**652**
Q20_20J_G2_3	**977.00**	**977**	977.00	0.18	**977**
Q20_20J_G2_4	**949.00**	**949**	949.00	0.20	**949**
Q20_20J_G2_5	1868.35	1915	1915.00	0.17	1915
Q20_20J_G3_1	**416.00**	**416**	416.00	0.35	**416**
Q20_20J_G3_2	**273.00**	**273**	273.00	0.29	**273**
Q20_20J_G3_3	**907.00**	**907**	907.00	0.25	**907**
Q20_20J_G3_4	**585.00**	**585**	585.00	0.23	**585**
Q20_20J_G3_5	**1120.00**	**1120**	1120.00	0.21	**1120**
Q20_20J_G4_1	3965.41	4017	4017.00	0.22	4017
Q20_20J_G4_2	**5664.00**	**5664**	5664.00	0.29	**5664**

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_20J_G4_3	3601.47	3645	3645.00	0.24	3645
Q20_20J_G4_4	**4095.00**	**4095**	4095.00	0.17	**4095**
Q20_20J_G4_5	4322.55	4369	4369.00	0.25	4369
Q20_20J_G5_1	1079.44	1089	1089.00	0.25	1089
Q20_20J_G5_2	2006.78	2051	2051.00	0.30	2051
Q20_20J_G5_3	**2920.00**	**2920**	2920.00	0.29	**2920**
Q20_20J_G5_4	2202.65	2214	2214.00	0.23	2214
Q20_20J_G5_5	**999.00**	**999**	999.00	0.24	**999**
Q20_20J_G6_1	**1515.00**	**1515**	1515.00	0.34	**1515**
Q20_20J_G6_2	**904.00**	**904**	904.00	0.26	**904**
Q20_20J_G6_3	**1615.00**	**1615**	1615.00	0.24	**1615**
Q20_20J_G6_4	**450.00**	**450**	450.00	0.33	**450**
Q20_20J_G6_5	**508.00**	**508**	508.00	0.31	**508**
Q20_20J_G7_1	4715.83	4733	4733.00	0.23	4733
Q20_20J_G7_2	**2576.00**	**2576**	2576.00	0.21	**2576**
Q20_20J_G7_3	2705.82	2808	2808.00	0.19	2808
Q20_20J_G7_4	**1357.00**	**1357**	1357.00	0.26	**1357**
Q20_20J_G7_5	**3985.00**	**3985**	3985.00	0.21	**3985**
Q20_20J_G8_1	2259.83	2324	2324.00	0.26	2324
Q20_20J_G8_2	**1879.00**	**1879**	1879.00	0.27	**1879**

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_20J_G8_3	**3293.00**	**3293**	3293.00	0.34	**3293**
Q20_20J_G8_4	2059.85	2062	2062.00	0.21	2062
Q20_20J_G8_5	**1388.00**	**1388**	1388.00	0.31	**1388**
Q20_20J_G9_1	**1752.00**	**1752**	1752.00	0.23	**1752**
Q20_20J_G9_2	**703.00**	**703**	703.00	0.29	**703**
Q20_20J_G9_3	**708.00**	**708**	708.00	0.28	**708**
Q20_20J_G9_4	**427.00**	**427**	427.00	0.28	**427**
Q20_20J_G9_5	**1244.00**	**1244**	1244.00	0.27	**1244**
Q20_25J_G1_1	3120.45	3134	3134.00	0.37	3134
Q20_25J_G1_2	**1852.00**	**1852**	1852.00	0.35	**1852**
Q20_25J_G1_3	3642.54	3734	3738.70	0.34	3734
Q20_25J_G1_4	1905.48	1939	1939.10	0.30	1939
Q20_25J_G1_5	2628.47	2706	2706.00	0.30	2706
Q20_25J_G2_1	**2442.00**	**2442**	2442.00	0.43	**2442**
Q20_25J_G2_2	**1408.00**	**1408**	1408.00	0.34	**1408**
Q20_25J_G2_3	**502.00**	**502**	502.00	0.41	**502**
Q20_25J_G2_4	1625.50	1647	1647.00	0.29	1647
Q20_25J_G2_5	2033.39	2068	2068.00	0.33	2068
Q20_25J_G3_1	**1216.00**	**1216**	1216.00	0.36	**1216**
Q20_25J_G3_2	**604.00**	**604**	604.00	0.36	**604**

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_25J_G3_3	**442.00**	**442**	442.00	0.60	**442**
Q20_25J_G3_4	**931.00**	**931**	931.00	0.55	**931**
Q20_25J_G3_5	1562.12	1569	1569.00	0.68	1569
Q20_25J_G4_1	4612.01	4706	4706.00	0.45	4706
Q20_25J_G4_2	4394.92	4456	4456.00	0.41	4456
Q20_25J_G4_3	4751.00	4758	4758.00	0.54	4758
Q20_25J_G4_4	**4687.00**	**4687**	4687.00	0.45	**4687**
Q20_25J_G4_5	**5132.00**	**5132**	5132.00	0.58	**5132**
Q20_25J_G5_1	3462.85	3529	3529.00	0.55	3529
Q20_25J_G5_2	1927.60	1999	1999.00	0.71	1999
Q20_25J_G5_3	1185.06	1253	1253.00	0.86	1253
Q20_25J_G5_4	**4499.00**	**4499**	4499.00	0.49	**4499**
Q20_25J_G5_5	**1399.00**	**1399**	1399.00	0.53	**1399**
Q20_25J_G6_1	**1249.00**	**1249**	1249.00	0.74	**1249**
Q20_25J_G6_2	913.50	960	960.00	0.60	960
Q20_25J_G6_3	1214.96	1223	1223.00	0.57	1223
Q20_25J_G6_4	**1585.00**	**1585**	1585.00	0.71	**1585**
Q20_25J_G6_5	**3327.00**	**3327**	3327.00	0.37	**3327**
Q20_25J_G7_1	**5641.00**	**5641**	5641.00	0.55	**5641**
Q20_25J_G7_2	**5592.00**	**5592**	5592.00	0.38	**5592**

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_25J_G7_3	**5835.00**	**5835**	5835.00	0.49	**5835**
Q20_25J_G7_4	**5043.00**	**5043**	5043.00	0.54	**5043**
Q20_25J_G7_5	5896.27	5968	5973.10	0.44	5968
Q20_25J_G8_1	2662.16	2675	2675.00	0.65	2675
Q20_25J_G8_2	**2523.00**	**2523**	2523.00	0.40	**2523**
Q20_25J_G8_3	**4459.00**	**4459**	4459.00	0.48	**4459**
Q20_25J_G8_4	1582.88	1730	1730.00	0.59	1730
Q20_25J_G8_5	2111.88	2195	2195.00	0.54	2195
Q20_25J_G9_1	1933.87	1937	1937.00	0.54	1937
Q20_25J_G9_2	**2128.00**	**2128**	2128.00	0.53	**2128**
Q20_25J_G9_3	**1670.00**	**1670**	1670.00	0.50	**1670**
Q20_25J_G9_4	4842.88	4933	4933.00	0.66	4933
Q20_25J_G9_5	2355.43	2391	2391.00	0.67	2391
Q20_30J_G1_1	2271.77	2327	2327.00	0.55	2327
Q20_30J_G1_2	2311.12	2332	2332.00	0.60	2332
Q20_30J_G1_3	3744.52	3819	3819.60	0.57	3819
Q20_30J_G1_4	3597.13	3691	3691.00	0.57	3691

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_30J_G1_5	3920.55	3969	3969.00	0.71	3969
Q20_30J_G2_1	2855.00	2894	2894.00	0.51	2894
Q20_30J_G2_2	3704.98	3818	3818.00	0.71	3818
Q20_30J_G2_3	1608.83	1629	1629.00	0.52	1629
Q20_30J_G2_4	3645.70	3765	3769.00	0.58	3765
Q20_30J_G2_5	2418.60	2466	2466.00	0.69	2466
Q20_30J_G3_1	1948.87	1995	1995.00	0.77	1995
Q20_30J_G3_2	**1643.00**	**1643**	1643.00	0.90	**1643**
Q20_30J_G3_3	**808.00**	**808**	808.00	0.68	**808**
Q20_30J_G3_4	2544.21	2651	2661.20	0.51	2651
Q20_30J_G3_5	**2022.00**	**2022**	2022.00	0.71	**2022**
Q20_30J_G4_1	7739.45	7775	7775.00	0.79	7775
Q20_30J_G4_2	5782.81	5836	5836.00	0.89	5836
Q20_30J_G4_3	7228.12	7312	7312.00	0.83	7312
Q20_30J_G4_4	4365.64	4481	4481.00	0.88	4481
Q20_30J_G4_5	9453.54	9462	9462.00	0.96	9462
Q20_30J_G5_1	**5090.00**	**5090**	5090.00	0.82	**5090**

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_30J_G5_2	7796.99	7817	7817.00	0.90	7817
Q20_30J_G5_3	3218.93	3395	3395.00	0.93	3395
Q20_30J_G5_4	3942.97	4070	4070.00	0.91	4070
Q20_30J_G5_5	**4230.00**	**4230**	4230.00	1.03	**4230**
Q20_30J_G6_1	2621.35	2790	2790.00	1.00	2790
Q20_30J_G6_2	**1444.00**	**1444**	1444.00	1.09	**1444**
Q20_30J_G6_3	**2290.00**	**2290**	2290.00	1.10	**2290**
Q20_30J_G6_4	1439.11	1476	1476.00	1.57	1476
Q20_30J_G6_5	2395.80	2516	2516.00	1.39	2516
Q20_30J_G7_1	6036.27	6107	6107.00	0.66	6107
Q20_30J_G7_2	7355.29	7497	7497.00	0.74	7497
Q20_30J_G7_3	6980.23	7052	7052.00	0.69	7052
Q20_30J_G7_4	**5457.00**	**5457**	5457.00	0.72	**5457**
Q20_30J_G7_5	7509.09	7774	7774.00	0.86	7774
Q20_30J_G8_1	3064.14	3266	3266.00	1.11	3266
Q20_30J_G8_2	4459.86	4528	4528.00	1.10	4528
Q20_30J_G8_3	4844.22	4911	4911.00	1.21	4911
Q20_30J_G8_4	4995.50	5019	5019.00	1.02	5019
Q20_30J_G8_5	**5398.00**	**5398**	5398.00	0.97	**5398**
Q20_30J_G9_1	**4082.00**	**4082**	4082.00	1.29	**4082**

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_30J_G9_2	**576.00**	**576**	576.00	0.85	**576**
Q20_30J_G9_3	**1572.00**	**1572**	1572.00	1.06	**1572**
Q20_30J_G9_4	5016.65	5171	5172.20	0.97	5171
Q20_30J_G9_5	2952.41	2980	2980.00	0.83	2980
Q20_35J_G1_1	3938.30	4070	4070.00	0.85	4070
Q20_35J_G1_2	5216.53	5360	5360.00	0.82	5360
Q20_35J_G1_3	3517.23	3609	3609.00	0.71	3609
Q20_35J_G1_4	4218.33	4232	4249.20	0.60	4232
Q20_35J_G1_5	4674.58	4761	4761.00	0.76	4761
Q20_35J_G2_1	**4315.00**	**4315**	4315.00	0.94	**4315**
Q20_35J_G2_2	1850.10	1898	1898.00	0.91	1898
Q20_35J_G2_3	1879.01	1925	1925.00	1.36	1925
Q20_35J_G2_4	2094.92	2131	2131.00	0.78	2131
Q20_35J_G2_5	1382.50	1464	1464.00	1.20	1464
Q20_35J_G3_1	**1912.00**	**1912**	1912.00	1.13	**1912**
Q20_35J_G3_2	1540.95	1609	1614.10	1.34	1609
Q20_35J_G3_3	2351.35	2466	2466.00	1.53	2466
Q20_35J_G3_4	1854.31	1926	1926.00	1.25	1926
Q20_35J_G3_5	1463.77	1507	1507.00	1.30	1507

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_35J_G4_1	6724.78	6755	6755.00	1.55	6755
Q20_35J_G4_2	11516.64	11653	11653.00	1.28	11653
Q20_35J_G4_3	11692.16	12017	12017.00	1.10	12017
Q20_35J_G4_4	8033.12	8066	8066.00	1.89	8066
Q20_35J_G4_5	7553.38	7595	7595.00	1.18	7595
Q20_35J_G5_1	7316.81	7449	7449.00	2.57	7449
Q20_35J_G5_2	6853.34	6897	6897.00	1.85	6897
Q20_35J_G5_3	5621.89	5837	5837.00	1.73	5837
Q20_35J_G5_4	8311.50	8510	8510.00	1.65	8510
Q20_35J_G5_5	3541.09	3637	3637.00	1.94	3637
Q20_35J_G6_1	2979.66	3048	3048.00	2.35	3048
Q20_35J_G6_2	**3540.00**	**3540**	3540.00	1.99	**3540**
Q20_35J_G6_3	2824.37	2833	2833.00	1.69	2833
Q20_35J_G6_4	**1688.00**	**1688**	1688.00	2.07	**1688**
Q20_35J_G6_5	2052.18	2174	2174.00	2.39	2174

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_35J_G7_1	14032.21	14074	14074.00	1.22	14074
Q20_35J_G7_2	5803.85	5914	5914.00	1.05	5914
Q20_35J_G7_3	9162.08	9238	9263.60	1.83	9238
Q20_35J_G7_4	9816.21	9850	9850.00	1.20	9850
Q20_35J_G7_5	**8487.00**	**8487**	8487.00	1.05	**8487**
Q20_35J_G8_1	4589.93	4726	4726.00	1.70	4726
Q20_35J_G8_2	7124.79	7285	7285.00	1.81	7285
Q20_35J_G8_3	9885.65	9919	9919.00	1.76	9919
Q20_35J_G8_4	5928.15	5945	5945.00	1.58	5945
Q20_35J_G8_5	4813.24	4923	4923.00	1.51	4923
Q20_35J_G9_1	1966.93	2270	2270.00	2.01	2270
Q20_35J_G9_2	**755.00**	**755**	755.00	1.97	**755**
Q20_35J_G9_3	10,048.54	10,175	10,175.00	2.77	10,175
Q20_35J_G9_4	1773.73	1868	1868.00	1.92	1868

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_35J_G9_5	3461.78	3555	3555.00	2.09	3555
Q20_50J_G1_1	9265.91	9346	9349.60	3.13	9346
Q20_50J_G1_2	8322.07	8742	8744.70	3.00	8742
Q20_50J_G1_3	10,639.50	10,811	10,873.70	3.18	10,811
Q20_50J_G1_4	9059.63	9218	9241.60	3.12	9218
Q20_50J_G1_5	7537.95	7568	7604.60	3.03	7568
Q20_50J_G2_1	5411.79	5548	5551.30	3.13	5548
Q20_50J_G2_2	5810.84	6127	6151.40	4.35	6097
Q20_50J_G2_3	7873.95	8059	8059.00	3.84	8059
Q20_50J_G2_4	5716.57	5978	5979.50	3.73	5978
Q20_50J_G2_5	7703.91	7843	7895.80	3.10	7843
Q20_50J_G3_1	2012.23	2179	2179.00	4.60	2179
Q20_50J_G3_2	2948.04	3283	3283.00	5.11	3283
Q20_50J_G3_3	2124.59	2179	2179.00	4.20	2179
Q20_50J_G3_4	3038.06	3204	3207.90	5.34	3204

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_50J_G3_5	4174.54	4329	4333.60	4.43	4329
Q20_50J_G4_1	14,424.91	14,498	14,498.00	4.68	14,498
Q20_50J_G4_2	**25,677.00**	**25,677**	25, 677.00	5.93	**25,677**
Q20_50J_G4_3	19,041.69	19,051	19,051.00	6.16	19,051
Q20_50J_G4_4	16,485.77	16,733	16,735.50	5.50	16,733
Q20_50J_G4_5	19,402.73	19,799	19,799.00	6.95	19,799
Q20_50J_G5_1	14,392.37	14,589	14,589.00	6.97	14,589
Q20_50J_G5_2	12,165.83	12,308	12,319.20	6.26	12,308
Q20_50J_G5_3	8137.89	8312	8312.00	5.55	8312
Q20_50J_G5_4	13,764.82	14,067	14,079.60	8.03	14,067
Q20_50J_G5_5	11,400.01	11,475	11,475.00	6.34	11,475
Q20_50J_G6_1	5322.65	5557	5558.00	7.47	5557
Q20_50J_G6_2	4424.02	4688	4688.00	7.50	4688
Q20_50J_G6_3	4666.45	4954	4954.00	7.67	4954
Q20_50J_G6_4	7728.42	8108	8108.00	8.29	8108
Q20_50J_G6_5	3097.16	3217	3217.00	7.26	3217
Q20_50J_G7_1	16,801.30	16,873	16,874.40	5.91	16,873
Q20_50J_G7_2	15,149.01	15,376	15,408.20	6.48	15,376
Q20_50J_G7_3	25,315.83	25,350	25,350.00	5.40	25,350
Q20_50J_G7_4	11,275.05	11,530	11,530.00	5.52	11,530

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_50J_G7_5	23,685.90	23,777	23,777.00	5.65	23,777
Q20_50J_G8_1	10,704.01	11,043	11,043.00	7.66	11,043
Q20_50J_G8_2	11,661.24	11,790	11,790.00	5.01	11,790
Q20_50J_G8_3	11,119.40	11,372	11,372.00	6.62	11,372
Q20_50J_G8_4	10,672.38	10,988	10,990.10	6.49	10,988
Q20_50J_G8_5	12,568.27	12,684	12,722.30	6.14	12,684
Q20_50J_G9_1	3081.25	3188	3188.00	7.15	3188
Q20_50J_G9_2	6266.25	6474	6530.20	8.13	6474
Q20_50J_G9_3	6977.10	7087	7113.40	8.00	7087
Q20_50J_G9_4	6065.58	6117	6117.00	10.51	6117
Q20_50J_G9_5	1890.05	1952	1952.90	7.94	1952
Q20_75J_G1_1	21,006.61	21,284	21,317.50	17.88	21,267
Q20_75J_G1_2	24,809.55	25,285	25,332.80	12.30	25,201
Q20_75J_G1_3	20,346.40	20,702	20,777.30	12.93	20,668
Q20_75J_G1_4	25,821.72	26,327	26,464.60	15.95	26,210
Q20_75J_G1_5	19,665.88	20,129	20,232.40	13.71	20,129
Q20_75J_G2_1	12,436.07	13,012	13,093.00	16.47	12,911
Q20_75J_G2_2	12,785.29	13,339	13,420.20	15.55	13,339

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_75J_G2_3	10,012.05	10,373	10,514.60	15.91	10,373
Q20_75J_G2_4	10,811.24	11,369	11,416.30	14.16	11,369
Q20_75J_G2_5	8604.05	9174	9268.50	19.33	9174
Q20_75J_G3_1	4835.50	5214	5214.00	22.05	5214
Q20_75J_G3_2	3546.02	3684	3685.50	23.69	3684
Q20_75J_G3_3	6324.97	6827	6924.30	21.06	6827
Q20_75J_G3_4	8576.14	9152	9223.20	25.17	9152
Q20_75J_G3_5	8218.97	8633	8660.00	20.69	8633
Q20_75J_G4_1	41,151.80	41,213	41,213.00	29.69	41,213
Q20_75J_G4_2	51,809.91	52,242	52,310.10	29.86	52,242
Q20_75J_G4_3	54,095.38	54,298	54,298.00	30.13	54,298

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_75J_G4_4	52,083.60	52,318	52,335.40	29.50	52,318
Q20_75J_G4_5	52,130.15	52,478	52,550.80	26.73	52,464
Q20_75J_G5_1	34,092.92	34,361	34,404.80	36.53	34,361
Q20_75J_G5_2	22,518.71	23,019	23,105.10	29.36	23,019
Q20_75J_G5_3	26,024.72	26,275	26,275.00	40.31	26,275
Q20_75J_G5_4	19,665.02	20,172	20,278.80	35.70	20,172
Q20_75J_G5_5	21,033.18	21,317	21,345.40	36.74	21,317
Q20_75J_G6_1	16,790.88	17,267	17,311.00	47.18	17,260
Q20_75J_G6_2	8500.12	8816	8816.00	43.98	8816
Q20_75J_G6_3	9749.39	9864	9864.00	40.81	9864
Q20_75J_G6_4	6685.74	7201	7204.80	33.46	7201
Q20_75J_G6_5	5365.48	5736	5808.80	45.32	5736
Q20_75J_G7_1	40,197.51	40,482	40,637.90	26.87	40,482
Q20_75J_G7_2	43,122.41	43,432	43,489.50	36.45	43,432

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_75J_G7_3	30,333.38	30,948	31,027.70	24.36	30,858
Q20_75J_G7_4	42,021.37	42,611	42,720.30	23.55	42,611
Q20_75J_G7_5	36,871.77	37,387	37,418.80	20.26	37,337
Q20_75J_G8_1	28,561.78	29,340	29,429.20	34.85	29,340
Q20_75J_G8_2	17,224.53	17,614	17,623.70	32.64	17,614
Q20_75J_G8_3	35,597.16	36,518	36,581.00	35.82	36,414
Q20_75J_G8_4	26,034.53	26,407	26,440.90	36.60	26,407
Q20_75J_G8_5	24,773.59	25,416	25,434.40	31.72	25,416
Q20_75J_G9_1	6908.53	7028	7028.00	35.88	7028
Q20_75J_G9_2	5212.60	5616	5616.00	39.60	5616
Q20_75J_G9_3	10,221.89	10,710	10,771.70	34.96	10,710
Q20_75J_G9_4	8544.75	8934	8962.20	36.72	8934
Q20_75J_G9_5	9894.28	10,211	10,211.00	34.54	10,211
Q20_100J_G1_1	35,205.10	36,109	36,278.60	40.37	35,930
Q20_100J_G1_2	39,092.30	40,318	40,612.50	40.01	39,985
Q20_100J_G1_3	40,960.58	41,823	41,947.40	47.90	41,519
Q20_100J_G1_4	37,365.52	38,459	38,556.20	38.00	38,254
Q20_100J_G1_5	34,809.20	35,731	35,903.00	42.85	35,663
Q20_100J_G2_1	29,342.03	30,461	30,587.70	53.20	30,453

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_100J_G2_2	23,717.58	24,749	25,312.90	52.75	24,639
Q20_100J_G2_3	21,804.13	22,584	22,902.20	52.24	22,477
Q20_100J_G2_4	23,276.94	24,671	25,049.40	50.76	24,671
Q20_100J_G2_5	24,742.71	25,921	26,182.20	47.12	25,818
Q20_100J_G3_1	7449.83	8510	8573.20	68.84	8510
Q20_100J_G3_2	4524.81	5107	5127.50	59.08	5061
Q20_100J_G3_3	11,635.56	12,902	13,263.10	59.21	12,871
Q20_100J_G3_4	6644.86	7481	7514.80	69.93	7438
Q20_100J_G3_5	10,561.57	11,999	12,147.30	76.91	11,910
Q20_100J_G4_1	94,699.32	95,150	95,157.00	87.46	95,150
Q20_100J_G4_2	65,523.59	66,298	66,439.30	83.84	66,235
Q20_100J_G4_3	69,671.64	70,065	70,149.60	91.51	70,037
Q20_100J_G4_4	81,881.55	82,578	82,714.40	93.42	82,578
Q20_100J_G4_5	82,965.40	83,416	83,529.60	90.41	83,416
Q20_100J_G5_1	46,313.53	47,466	47,555.60	93.12	47,434
Q20_100J_G5_2	31,651.30	32,600	32,961.30	121.42	32,600
Q20_100J_G5_3	51,635.75	52,786	52,933.40	109.51	52,786
Q20_100J_G5_4	69,372.41	70,025	70,335.70	98.78	70,025
Q20_100J_G5_5	37,169.67	38,135	38,348.80	102.74	38,043
Q20_100J_G6_1	13,005.06	13,788	13,848.60	122.34	13,788

Table 12 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q20_100J_G6_2	17,459.60	18,514	18,646.00	146.67	18,446
Q20_100J_G6_3	15,609.43	17,071	17,168.20	118.92	17,044
Q20_100J_G6_4	8873.67	9374	9436.40	135.80	9374
Q20_100J_G6_5	15,666.43	16,485	16,537.10	156.52	16,408
Q20_100J_G7_1	72,195.10	72,602	72,806.60	103.34	72,602
Q20_100J_G7_2	53,706.02	54,496	54,745.80	74.62	54,387
Q20_100J_G7_3	98,158.55	98,760	98,965.30	86.66	98,760
Q20_100J_G7_4	82,930.39	83,776	83,944.70	77.15	83,706
Q20_100J_G7_5	78,394.74	79,203	79,406.00	90.13	79,099
Q20_100J_G8_1	54,473.10	55,451	55,593.90	128.55	55,451
Q20_100J_G8_2	57,422.98	58,513	58,700.70	135.24	58,513
Q20_100J_G8_3	27,337.01	28,079	28,168.60	102.65	28,079
Q20_100J_G8_4	44,579.69	45,507	45,846.90	108.55	45,478
Q20_100J_G8_5	34,191.92	35,205	35,253.00	93.30	35,204
Q20_100J_G9_1	9109.82	10,054	10,175.10	122.22	10,054
Q20_100J_G9_2	22,536.43	23,794	23,796.40	146.15	23,786
Q20_100J_G9_3	11,880.43	13,196	13,255.50	122.17	13,196
Q20_100J_G9_4	9168.29	9696	9700.80	117.56	9696
Q20_100J_G9_5	10,893.26	11,712	11,712.00	126.59	11,712

Table 13 Detailed results of ILS_{batch} with $Q = 40$

Instance	LB	Min	Avg	Time (s)	BKS
Q40_10J_G1_1	**754.00**	**754**	754.00	0.03	**754**
Q40_10J_G1_2	**351.00**	**351**	351.00	0.03	**351**
Q40_10J_G1_3	**287.00**	**287**	287.00	0.02	**287**
Q40_10J_G1_4	**428.00**	**428**	428.00	0.02	**428**
Q40_10J_G1_5	**235.00**	**235**	235.00	0.03	**235**
Q40_10J_G2_1	**228.00**	**228**	228.00	0.04	**228**
Q40_10J_G2_2	**109.00**	**109**	109.00	0.03	**109**
Q40_10J_G2_3	**493.00**	**493**	493.00	0.04	**493**
Q40_10J_G2_4	**230.00**	**230**	230.00	0.03	**230**
Q40_10J_G2_5	**374.00**	**374**	374.00	0.04	**374**
Q40_10J_G3_1	**159.00**	**159**	159.00	0.04	**159**
Q40_10J_G3_2	**126.00**	**126**	126.00	0.06	**126**
Q40_10J_G3_3	**50.00**	**50**	50.00	0.02	**50**
Q40_10J_G3_4	**323.00**	**323**	323.00	0.06	**323**
Q40_10J_G3_5	**185.00**	**185**	185.00	0.06	**185**
Q40_10J_G4_1	**174.00**	**174**	174.00	0.03	**174**
Q40_10J_G4_2	**248.00**	**248**	248.00	0.03	**248**
Q40_10J_G4_3	**241.00**	**241**	241.00	0.03	**241**
Q40_10J_G4_4	**297.00**	**297**	297.00	0.03	**297**

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_10J_G4_5	**848.00**	**848**	848.00	0.03	**848**
Q40_10J_G5_1	**317.00**	**317**	317.00	0.03	**317**
Q40_10J_G5_2	**576.00**	**576**	576.00	0.03	**576**
Q40_10J_G5_3	**885.00**	**885**	885.00	0.04	**885**
Q40_10J_G5_4	**956.00**	**956**	956.00	0.04	**956**
Q40_10J_G5_5	**282.00**	**282**	282.00	0.04	**282**
Q40_10J_G6_1	**355.00**	**355**	355.00	0.04	**355**
Q40_10J_G6_2	**453.00**	**453**	453.00	0.05	**453**
Q40_10J_G6_3	**370.00**	**370**	370.00	0.05	**370**
Q40_10J_G6_4	**213.00**	**213**	213.00	0.04	**213**
Q40_10J_G6_5	**337.00**	**337**	337.00	0.03	**337**
Q40_10J_G7_1	**990.00**	**990**	990.00	0.04	**990**
Q40_10J_G7_2	**629.00**	**629**	629.00	0.02	**629**
Q40_10J_G7_3	**1402.00**	**1402**	1402.00	0.02	**1402**
Q40_10J_G7_4	**280.00**	**280**	280.00	0.02	**280**
Q40_10J_G8_5	**433.00**	**433**	433.00	0.03	**433**
Q40_10J_G9_1	**171.00**	**171**	171.00	0.02	**171**
Q40_10J_G9_2	**153.00**	**153**	153.00	0.02	**153**
Q40_10J_G9_3	**474.00**	**474**	474.00	0.03	**474**
Q40_10J_G9_4	**261.00**	**261**	261.00	0.03	**261**

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_10J_G7_5	**795.00**	**795**	795.00	0.02	**795**
Q40_10J_G8_1	**504.00**	**504**	504.00	0.02	**504**
Q40_10J_G8_2	**546.00**	**546**	546.00	0.02	**546**
Q40_10J_G8_3	**389.00**	**389**	389.00	0.04	**389**
Q40_10J_G8_4	**656.00**	**656**	656.00	0.02	**656**
Q40_10J_G9_5	**371.00**	**371**	371.00	0.05	**371**
Q40_15J_G1_1	**1388.00**	**1388**	1388.00	0.07	**1388**
Q40_15J_G1_2	**898.00**	**898**	898.00	0.07	**898**
Q40_15J_G1_3	**1199.00**	**1199**	1199.00	0.11	**1199**
Q40_15J_G1_4	**933.00**	**933**	933.00	0.09	**933**
Q40_15J_G1_5	**693.00**	**693**	693.00	0.08	**693**
Q40_15J_G2_1	**769.00**	**769**	769.00	0.10	**769**
Q40_15J_G2_2	**351.00**	**351**	351.00	0.06	**351**
Q40_15J_G2_3	**858.00**	**858**	858.00	0.11	**858**
Q40_15J_G2_4	**517.00**	**517**	517.00	0.08	**517**
Q40_15J_G2_5	**1205.00**	**1205**	1205.00	0.10	**1205**
Q40_15J_G3_1	**912.00**	**912**	912.00	0.13	**912**
Q40_15J_G3_2	**1247.00**	**1247**	1247.00	0.12	**1247**

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_15J_G3_3	**366.00**	**366**	366.00	0.12	**366**
Q40_15J_G3_4	**91.00**	**91**	91.00	0.16	**91**
Q40_15J_G3_5	**364.00**	**364**	364.00	0.11	**364**
Q40_15J_G4_1	**2555.00**	**2555**	2555.00	0.13	**2555**
Q40_15J_G4_2	**3719.00**	**3719**	3719.00	0.11	**3719**
Q40_15J_G4_3	**2009.00**	**2009**	2009.00	0.10	**2009**
Q40_15J_G4_4	**1025.00**	**1025**	1025.00	0.10	**1025**
Q40_15J_G4_5	**1730.00**	**1730**	1730.00	0.08	**1730**
Q40_15J_G5_1	**1196.00**	**1196**	1196.00	0.09	**1196**
Q40_15J_G5_2	**1337.00**	**1337**	1337.00	0.08	**1337**
Q40_15J_G5_3	**923.00**	**923**	923.00	0.09	**923**
Q40_15J_G5_4	**937.00**	**937**	937.00	0.11	**937**
Q40_15J_G5_5	**899.00**	**899**	899.00	0.08	**899**
Q40_15J_G6_1	**329.00**	**329**	329.00	0.14	**329**
Q40_15J_G6_2	**366.00**	**366**	366.00	0.11	**366**
Q40_15J_G6_3	**1895.00**	**1895**	1895.00	0.13	**1895**
Q40_15J_G6_4	**693.00**	**693**	693.00	0.11	**693**
Q40_15J_G6_5	**415.00**	**415**	415.00	0.12	**415**
Q40_15J_G7_1	**2007.00**	**2007**	2007.00	0.10	**2007**
Q40_15J_G7_2	**1303.00**	**1303**	1303.00	0.09	**1303**

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_15J_G7_3	**1009.00**	**1009**	1009.00	0.10	**1009**
Q40_15J_G7_4	**1048.00**	**1048**	1048.00	0.12	**1048**
Q40_15J_G7_5	**1428.00**	**1428**	1428.00	0.07	**1428**
Q40_15J_G8_1	**1669.00**	**1669**	1669.00	0.09	**1669**
Q40_15J_G8_2	**1613.00**	**1613**	1613.00	0.12	**1613**
Q40_15J_G8_3	**1800.00**	**1800**	1800.00	0.13	**1800**
Q40_15J_G8_4	**1282.00**	**1282**	1282.00	0.09	**1282**
Q40_15J_G8_5	**1487.00**	**1487**	1487.00	0.08	**1487**
Q40_15J_G9_1	**106.00**	**106**	106.00	0.11	**106**
Q40_15J_G9_2	**832.00**	**832**	832.00	0.12	**832**
Q40_15J_G9_3	**419.00**	**419**	419.00	0.09	**419**
Q40_15J_G9_4	**2203.00**	**2203**	2203.00	0.10	**2203**
Q40_15J_G9_5	**327.00**	**327**	327.00	0.14	**327**
Q40_20J_G1_1	1308.67	1324	1324.00	0.21	1324
Q40_20J_G1_2	**1885.00**	**1885**	1885.00	0.16	**1885**
Q40_20J_G1_3	**1170.00**	**1170**	1170.00	0.16	**1170**
Q40_20J_G1_4	**1935.00**	**1935**	1935.00	0.15	**1935**
Q40_20J_G1_5	**1898.00**	**1898**	1898.00	0.25	**1898**
Q40_20J_G2_1	**597.00**	**597**	597.00	0.21	**597**

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_20J_G2_2	**1446.00**	**1446**	1446.00	0.20	**1446**
Q40_20J_G2_3	**1036.00**	**1036**	1036.00	0.24	**1036**
Q40_20J_G2_4	**935.00**	**935**	935.00	0.24	**935**
Q40_20J_G2_5	**1236.00**	**1236**	1236.00	0.17	**1236**
Q40_20J_G3_1	**391.00**	**391**	391.00	0.31	**391**
Q40_20J_G3_2	**608.00**	**608**	608.00	0.34	**608**
Q40_20J_G3_3	**684.00**	**684**	684.00	0.29	**684**
Q40_20J_G3_4	**1326.00**	**1326**	1326.00	0.37	**1326**
Q40_20J_G3_5	**645.00**	**645**	645.00	0.29	**645**
Q40_20J_G4_1	**3446.00**	**3446**	3446.00	0.26	**3446**
Q40_20J_G4_2	2710.91	2734	2734.00	0.25	2734
Q40_20J_G4_3	**2834.00**	**2834**	2834.00	0.31	**2834**
Q40_20J_G4_4	**3802.00**	**3802**	3802.00	0.23	**3802**
Q40_20J_G4_5	4499.90	4541	4541.00	0.20	4541
Q40_20J_G5_1	**1931.00**	**1931**	1931.00	0.25	**1931**
Q40_20J_G5_2	**2395.00**	**2395**	2395.00	0.27	**2395**

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_20J_G5_3	**4309.00**	**4309**	4309.00	0.20	**4309**
Q40_20J_G5_4	**2045.00**	**2045**	2045.00	0.24	**2045**
Q40_20J_G5_5	**2093.00**	**2093**	2093.00	0.21	**2093**
Q40_20J_G6_1	**1088.00**	**1088**	1088.00	0.22	**1088**
Q40_20J_G6_2	**1368.00**	**1368**	1368.00	0.26	**1368**
Q40_20J_G6_3	**1793.00**	**1793**	1793.00	0.28	**1793**
Q40_20J_G6_4	**828.00**	**828**	828.00	0.29	**828**
Q40_20J_G6_5	**2044.00**	**2044**	2044.00	0.22	**2044**
Q40_20J_G7_1	**1798.00**	**1798**	1798.00	0.16	**1798**
Q40_20J_G7_2	**2540.00**	**2540**	2540.00	0.18	**2540**
Q40_20J_G7_3	**3026.00**	**3026**	3026.00	0.20	**3026**
Q40_20J_G7_4	4320.07	4413	4413.00	0.25	4413
Q40_20J_G7_5	**4302.00**	**4302**	4302.00	0.21	**4302**
Q40_20J_G8_1	**1250.00**	**1250**	1250.00	0.25	**1250**
Q40_20J_G8_2	2226.86	2267	2267.00	0.23	2267
Q40_20J_G8_3	**1792.00**	**1792**	1792.00	0.26	**1792**
Q40_20J_G8_4	**1572.00**	**1572**	1572.00	0.23	**1572**
Q40_20J_G8_5	**1569.00**	**1569**	1569.00	0.29	**1569**
Q40_20J_G9_1	**1562.00**	**1562**	1562.00	0.23	**1562**

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_20J_G9_2	**1266.00**	**1266**	1266.00	0.28	**1266**
Q40_20J_G9_3	**535.00**	**535**	535.00	0.36	**535**
Q40_20J_G9_4	**909.00**	**909**	909.00	0.26	**909**
Q40_20J_G9_5	**1097.00**	**1097**	1097.00	0.28	**1097**
Q40_25J_G1_1	**3422.00**	**3422**	3422.00	0.32	**3422**
Q40_25J_G1_2	1972.48	2015	2021.40	0.34	2015
Q40_25J_G1_3	2058.97	2083	2083.00	0.30	2083
Q40_25J_G1_4	2873.65	2922	2922.00	0.33	2922
Q40_25J_G1_5	2096.16	2131	2131.00	0.36	2131
Q40_25J_G2_1	1681.19	1704	1704.00	0.44	1704
Q40_25J_G2_2	1287.00	1378	1378.00	0.32	1378
Q40_25J_G2_3	1535.80	1584	1584.00	0.37	1584
Q40_25J_G2_4	**2369.00**	**2369**	2369.00	0.36	**2369**
Q40_25J_G2_5	1917.87	1945	1945.00	0.34	1945
Q40_25J_G3_1	947.53	948	948.00	0.56	948
Q40_25J_G3_2	1230.12	1238	1238.00	0.39	1238
Q40_25J_G3_3	971.24	979	979.00	0.54	979
Q40_25J_G3_4	**476.00**	**476**	476.00	0.59	**476**
Q40_25J_G3_5	1092.61	1095	1095.00	0.42	1095
Q40_25J_G4_1	**5640.00**	**5640**	5640.00	0.41	**5640**

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_25J_G4_2	**3916.00**	**3916**	3916.00	0.52	**3916**
Q40_25J_G4_3	5613.67	5752	5752.00	0.36	5752
Q40_25J_G4_4	5732.50	5802	5802.00	0.50	5802
Q40_25J_G4_5	6456.67	6477	6477.00	0.50	6477
Q40_25J_G5_1	**1821.00**	**1821**	1821.00	0.44	**1821**
Q40_25J_G5_2	**3676.00**	**3676**	3676.00	0.41	**3676**
Q40_25J_G5_3	4428.00	4440	4440.00	0.38	4440
Q40_25J_G5_4	**6856.00**	**6856**	6856.00	0.44	**6856**
Q40_25J_G5_5	2057.06	2158	2158.00	0.39	2158
Q40_25J_G6_1	**1154.00**	**1154**	1154.00	0.52	**1154**
Q40_25J_G6_2	**2008.00**	**2008**	2008.00	0.68	**2008**
Q40_25J_G6_3	2069.90	2077	2077.00	0.62	2077
Q40_25J_G6_4	**1303.00**	**1303**	1303.00	0.56	**1303**
Q40_25J_G6_5	**1264.00**	**1264**	1264.00	0.70	**1264**
Q40_25J_G7_1	4318.00	4326	4326.00	0.56	4326
Q40_25J_G7_2	5287.00	5302	5302.00	0.35	5302
Q40_25J_G7_3	6318.02	6393	6393.00	0.45	6393
Q40_25J_G7_4	3289.58	3408	3408.00	0.37	3408

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_25J_G7_5	3753.14	3784	3784.00	0.51	3784
Q40_25J_G8_1	3867.31	3906	3906.00	0.56	3906
Q40_25J_G8_2	2474.67	2498	2498.00	0.40	2498
Q40_25J_G8_3	**3283.00**	**3283**	3283.00	0.73	**3283**
Q40_25J_G8_4	4753.00	4870	4870.00	0.51	4870
Q40_25J_G8_5	2821.70	2996	2996.00	0.65	2996
Q40_25J_G9_1	**1387.00**	**1387**	1387.00	0.57	**1387**
Q40_25J_G9_2	**1579.00**	**1579**	1579.00	0.63	**1579**
Q40_25J_G9_3	**825.00**	**825**	825.00	0.55	**825**
Q40_25J_G9_4	**472.00**	**472**	472.00	0.67	**472**
Q40_25J_G9_5	**740.00**	**740**	740.00	0.58	**740**
Q40_30J_G1_1	5986.89	6133	6133.00	0.67	6133
Q40_30J_G1_2	5294.63	5498	5498.00	0.53	5498

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_30J_G1_3	**4068.00**	**4068**	4068.00	0.48	**4068**
Q40_30J_G1_4	4738.74	4785	4785.00	0.65	4785
Q40_30J_G1_5	**3949.00**	**3949**	3949.00	0.41	**3949**
Q40_30J_G2_1	2475.79	2574	2574.00	0.78	2574
Q40_30J_G2_2	3429.65	3515	3516.90	0.92	3515
Q40_30J_G2_3	2609.53	2670	2670.00	0.58	2670
Q40_30J_G2_4	2110.63	2167	2167.00	0.85	2167
Q40_30J_G2_5	2383.49	2467	2467.00	0.78	2467
Q40_30J_G3_1	982.96	1045	1045.00	0.85	1045
Q40_30J_G3_2	2232.48	2325	2326.00	1.08	2325
Q40_30J_G3_3	3073.62	3138	3138.00	0.98	3138
Q40_30J_G3_4	1471.83	1508	1508.00	0.85	1508
Q40_30J_G3_5	2103.22	2121	2121.00	0.91	2121
Q40_30J_G4_1	9104.64	9118	9118.00	0.73	9118
Q40_30J_G4_2	7485.63	7812	7812.00	0.77	7812
Q40_30J_G4_3	7169.76	7196	7196.00	0.78	7196
Q40_30J_G4_4	**7680.00**	**7680**	7680.00	0.63	**7680**
Q40_30J_G4_5	10,097.80	10,112	10,112.00	0.79	10,112

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_30J_G5_1	**4881.00**	**4881**	4881.00	1.17	**4881**
Q40_30J_G5_2	4666.62	4686	4686.00	1.36	4686
Q40_30J_G5_3	3890.24	3979	3979.00	0.89	3979
Q40_30J_G5_4	6078.50	6159	6159.00	1.25	6159
Q40_30J_G5_5	**5620.00**	**5620**	5620.00	1.06	**5620**
Q40_30J_G6_1	2595.80	2762	2762.00	1.34	2762
Q40_30J_G6_2	**2652.00**	**2652**	2652.00	1.22	**2652**
Q40_30J_G6_3	1064.92	1074	1074.00	1.08	1074
Q40_30J_G6_4	**1429.00**	**1429**	1429.00	1.06	**1429**
Q40_30J_G6_5	2060.62	2086	2086.00	1.34	2086
Q40_30J_G7_1	7038.97	7119	7119.00	0.69	7119
Q40_30J_G7_2	4901.03	5045	5045.00	0.72	5045
Q40_30J_G7_3	6632.34	6689	6689.00	0.87	6689
Q40_30J_G7_4	5503.76	5528	5528.00	0.73	5528
Q40_30J_G7_5	4715.00	4742	4742.00	0.62	4742

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_30J_G8_1	**4489.00**	**4489**	4491.00	0.97	**4489**
Q40_30J_G8_2	3925.77	4018	4018.00	0.89	4018
Q40_30J_G8_3	10,042.50	10,158	10,158.00	0.83	10,158
Q40_30J_G8_4	5065.29	5091	5091.00	0.99	5091
Q40_30J_G8_5	6221.31	6337	6337.00	0.80	6337
Q40_30J_G9_1	2652.10	2666	2666.00	0.77	2666
Q40_30J_G9_2	1665.88	1795	1795.00	1.07	1795
Q40_30J_G9_3	1271.00	1335	1335.00	1.15	1335
Q40_30J_G9_4	2521.41	2722	2722.00	0.91	2722
Q40_30J_G9_5	2555.88	2590	2590.00	1.26	2590
Q40_35J_G1_1	3236.48	3302	3302.00	0.81	3302
Q40_35J_G1_2	3938.16	4068	4069.00	0.84	4068
Q40_35J_G1_3	5706.61	5833	5833.00	0.85	5833
Q40_35J_G1_4	3593.97	3677	3683.20	0.81	3677
Q40_35J_G1_5	4351.00	4372	4372.00	0.78	4372
Q40_35J_G2_1	**2118.00**	**2118**	2118.00	0.87	**2118**
Q40_35J_G2_2	3833.21	3962	3989.00	1.05	3962
Q40_35J_G2_3	2441.79	2446	2446.00	1.12	2446
Q40_35J_G2_4	1851.45	1895	1895.00	0.88	1895
Q40_35J_G2_5	3012.14	3026	3026.00	0.77	3026

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_35J_G3_1	2413.10	2558	2558.00	1.14	2558
Q40_35J_G3_2	1814.04	1832	1832.00	1.61	1832
Q40_35J_G3_3	2035.76	2094	2094.00	1.38	2094
Q40_35J_G3_4	**1383.00**	**1383**	1383.00	1.25	**1383**
Q40_35J_G3_5	2430.18	2443	2443.00	1.04	2443
Q40_35J_G4_1	7432.25	7434	7434.00	1.17	7434
Q40_35J_G4_2	6864.34	6973	6990.90	1.55	6973
Q40_35J_G4_3	7443.55	7499	7499.00	2.00	7499
Q40_35J_G4_4	14,458.45	14,655	14,655.00	1.48	14,655
Q40_35J_G4_5	11,870.34	11,946	11,946.00	1.68	11,946
Q40_35J_G5_1	9108.26	9243	9243.00	2.28	9243
Q40_35J_G5_2	5405.36	5725	5745.00	1.94	5725
Q40_35J_G5_3	**5616.00**	**5616**	5616.00	1.24	**5616**
Q40_35J_G5_4	8968.19	9224	9224.00	1.74	9224
Q40_35J_G5_5	5901.34	6388	6388.00	1.59	6388
Q40_35J_G6_1	2322.93	2351	2351.00	2.49	2351
Q40_35J_G6_2	2409.15	2465	2465.00	1.91	2465
Q40_35J_G6_3	2990.33	3197	3197.00	1.81	3197
Q40_35J_G6_4	1475.88	1485	1485.00	2.34	1485
Q40_35J_G6_5	4903.30	4958	4958.00	1.54	4958

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_35J_G7_1	**10,440.00**	**10,440**	10,440.00	1.33	**10,440**
Q40_35J_G7_2	12,627.42	12,643	12,643.00	1.13	12,643
Q40_35J_G7_3	12,897.52	13,146	13,146.00	1.18	13,146
Q40_35J_G7_4	9444.74	9661	9661.00	1.54	9661
Q40_35J_G7_5	**7792.00**	**7792**	7792.00	0.89	**7792**
Q40_35J_G8_1	5468.24	5488	5488.00	1.34	5488
Q40_35J_G8_2	5633.46	5681	5681.00	1.96	5681
Q40_35J_G8_3	7376.22	7566	7566.00	1.55	7566
Q40_35J_G8_4	2639.94	2675	2675.00	1.74	2675
Q40_35J_G8_5	5117.39	5295	5295.60	1.90	5295
Q40_35J_G9_1	2247.71	2265	2265.00	2.06	2265
Q40_35J_G9_2	3597.15	3663	3663.00	1.75	3663
Q40_35J_G9_3	**2205.00**	**2205**	2205.00	2.01	**2205**
Q40_35J_G9_4	6301.65	6399	6399.00	2.05	6399
Q40_35J_G9_5	5328.67	5349	5349.00	1.97	5349
Q40_50J_G1_1	14,410.94	14,525	14,540.50	3.78	14,525
Q40_50J_G1_2	8280.14	8499	8511.40	3.67	8499
Q40_50J_G1_3	8698.47	8710	8710.00	4.52	8710

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_50J_G1_4	10,395.25	10,784	10,784.00	3.61	10,784
Q40_50J_G1_5	11,059.75	11,329	11,334.20	4.87	11,329
Q40_50J_G2_1	10,446.85	10,484	10,484.00	4.16	10,484
Q40_50J_G2_2	10,578.43	10,853	10,865.40	4.23	10,853
Q40_50J_G2_3	5071.90	5434	5453.00	5.55	5419
Q40_50J_G2_4	11,135.62	11,180	11,180.00	3.93	11,180
Q40_50J_G2_5	10,290.48	10,448	10,448.00	4.86	10,448
Q40_50J_G3_1	5833.38	6025	6057.00	5.31	6025
Q40_50J_G3_2	2400.82	2451	2451.00	5.41	2451
Q40_50J_G3_3	4281.56	4324	4324.00	5.11	4324
Q40_50J_G3_4	3250.95	3452	3452.00	4.93	3452
Q40_50J_G3_5	5609.98	5837	5837.00	5.23	5837
Q40_50J_G4_1	14,605.43	14,692	14,717.60	5.49	14,692
Q40_50J_G4_2	18,192.93	18,287	18,287.00	4.81	18,287
Q40_50J_G4_3	20,605.27	20,903	20,910.00	6.40	20,903
Q40_50J_G4_4	22,971.60	23,093	23,093.00	6.54	23,093

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_50J_G4_5	24,063.57	24,165	24,190.00	4.77	24,165
Q40_50J_G5_1	20,666.70	20,931	20,944.00	6.42	20,931
Q40_50J_G5_2	12,297.78	12,524	12,543.00	6.70	12,524
Q40_50J_G5_3	9019.91	9284	9284.00	6.51	9284
Q40_50J_G5_4	11,306.79	11,326	11,341.20	7.60	11,326
Q40_50J_G5_5	18,352.25	18,501	18,501.00	6.33	18,501
Q40_50J_G6_1	5021.03	5097	5097.00	6.88	5097
Q40_50J_G6_2	6135.38	6262	6262.00	7.32	6262
Q40_50J_G6_3	4994.12	5296	5296.00	7.99	5296
Q40_50J_G6_4	6021.53	6246	6246.60	8.92	6246
Q40_50J_G6_5	7116.81	7175	7175.00	8.90	7175
Q40_50J_G7_1	17,911.03	18,019	18,019.00	6.56	18,019
Q40_50J_G7_2	22,211.55	22,535	22,535.00	5.25	22,535
Q40_50J_G7_3	**24,433.00**	**24,433**	24, 433.00	5.30	**24,433**
Q40_50J_G7_4	20,237.79	20,449	20,449.00	5.89	20,449
Q40_50J_G7_5	13,226.42	13,390	13,390.00	4.87	13,390

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_50J_G8_1	14,109.55	14,382	14,403.60	5.56	14,382
Q40_50J_G8_2	10,387.36	10,542	10,542.00	4.75	10,542
Q40_50J_G8_3	**11,870.00**	**11,870**	11, 870.00	5.51	**11,870**
Q40_50J_G8_4	16,124.11	16,458	16,467.10	6.64	16,458
Q40_50J_G8_5	11,259.84	11,467	11,467.00	7.27	11,467
Q40_50J_G9_1	5423.22	5916	5947.20	6.80	5916
Q40_50J_G9_2	5961.71	6301	6301.00	5.73	6301
Q40_50J_G9_3	2179.73	2645	2647.10	7.20	2645
Q40_50J_G9_4	5862.69	6027	6037.70	7.07	6027
Q40_50J_G9_5	3660.05	4096	4097.60	6.61	4096
Q40_75J_G1_1	30,049.34	30,527	30,637.80	17.23	30,452
Q40_75J_G1_2	32,982.90	33,499	33,598.60	15.32	33,418
Q40_75J_G1_3	30,418.73	30,893	31,020.30	15.57	30,826
Q40_75J_G1_4	28,324.26	29,019	29,109.60	19.20	29,019
Q40_75J_G1_5	27,558.80	28,042	28,209.10	16.10	28,042
Q40_75J_G2_1	25,842.55	26,470	26,581.90	19.25	26,251
Q40_75J_G2_2	13,302.13	13,642	13,648.40	19.73	13,642
Q40_75J_G2_3	15,969.72	16,643	16,749.50	22.07	16,587
Q40_75J_G2_4	10,556.54	11,155	11,242.70	20.71	11,155
Q40_75J_G2_5	14,900.19	15,523	15,640.10	21.17	15,472

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_75J_G3_1	7369.94	7814	8181.50	27.04	7691
Q40_75J_G3_2	7263.16	7597	7622.70	25.95	7597
Q40_75J_G3_3	6421.98	6976	7118.40	26.61	6976
Q40_75J_G3_4	5801.39	6183	6202.30	24.73	6183
Q40_75J_G3_5	9733.85	9855	9928.50	27.47	9855
Q40_75J_G4_1	40,135.43	40,225	40,241.00	27.45	40,225
Q40_75J_G4_2	47,976.01	48,177	48,177.00	27.55	48,177
Q40_75J_G4_3	41,313.83	41,390	41,396.00	27.19	41,390
Q40_75J_G4_4	52,103.01	52,217	52,310.50	25.83	52,217
Q40_75J_G4_5	46,359.91	46,605	46,607.30	24.94	46,605
Q40_75J_G5_1	25,362.66	25,676	25,683.10	41.08	25,676
Q40_75J_G5_2	34,122.89	34,465	34,488.10	30.67	34,457
Q40_75J_G5_3	36,176.96	36,416	36,516.30	37.52	36,416
Q40_75J_G5_4	23,283.05	23,501	23,501.00	33.15	23,501

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_75J_G5_5	23,799.69	24,088	24,088.00	35.84	24,088
Q40_75J_G6_1	5770.82	6095	6095.00	36.22	6095
Q40_75J_G6_2	9883.89	10,353	10,353.00	42.42	10,353
Q40_75J_G6_3	13,389.65	14,300	14,344.10	39.14	14,182
Q40_75J_G6_4	18,595.65	18,867	18,867.00	39.63	18,867
Q40_75J_G6_5	15,438.88	15,556	15,556.00	51.84	15,556
Q40_75J_G7_1	41,112.83	41,511	41,516.10	29.81	41,511
Q40_75J_G7_2	57,219.47	57,877	57,925.80	28.31	57,877
Q40_75J_G7_3	44,119.15	44,538	44,570.70	24.59	44,538
Q40_75J_G7_4	46,232.29	46,482	46,489.20	25.37	46,482
Q40_75J_G7_5	41,084.99	41,298	41,327.20	23.02	41,273
Q40_75J_G8_1	26,349.07	26,625	26,639.50	36.62	26,625
Q40_75J_G8_2	23,281.04	23,717	23,738.50	36.22	23,717
Q40_75J_G8_3	33,268.86	33,489	33,492.60	51.90	33,489
Q40_75J_G8_4	20,524.87	21,139	21,210.30	32.60	21,125

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_75J_G8_5	31,221.03	31,536	31,697.60	31.10	31,536
Q40_75J_G9_1	11,043.59	11,465	11,480.20	46.79	11,465
Q40_75J_G9_2	12,616.98	12,999	12,999.00	37.21	12,999
Q40_75J_G9_3	12,640.60	13,076	13,076.00	39.96	13,076
Q40_75J_G9_4	11,739.87	12,404	12,430.80	39.36	12,404
Q40_75J_G9_5	7154.89	7379	7380.90	37.28	7379
Q40_100J_G1_1	77.99	45,831	45,953.10	49.95	45,829
Q40_100J_G1_2	170.28	57,383	57,565.60	54.38	57,067
Q40_100J_G1_3	610.74	45,965	46,092.30	52.53	45,820
Q40_100J_G1_4	38,205.44	38,786	38,834.10	56.12	38,738
Q40_100J_G1_5	626.15	59,648	59,833.50	60.44	59,385
Q40_100J_G2_1	126.44	42,378	42,544.90	71.95	42,175
Q40_100J_G2_2	235.41	37,100	37,315.00	60.40	37,093
Q40_100J_G2_3	217.98	34,484	35,077.80	78.89	34,370

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_100J_G2_4	56.42	28,677	28,870.90	69.05	28,676
Q40_100J_G2_5	216.06	33,849	34,113.60	62.01	33,820
Q40_100J_G3_1	440.28	19,591	19,714.90	80.99	19,512
Q40_100J_G3_2	300.40	15,601	15,691.30	99.47	15,601
Q40_100J_G3_3	276.54	17,322	17,542.80	88.29	17,242
Q40_100J_G3_4	420.95	10,827	10,882.10	82.40	10,827
Q40_100J_G3_5	254.20	14,112	14,321.30	81.44	14,112
Q40_100J_G4_1	95,694.62	96,013	96,044.20	101.29	96,013
Q40_100J_G4_2	80,696.38	80,883	80,962.00	87.52	80,883
Q40_100J_G4_3	85,853.45	86,275	86,424.80	111.06	86,275
Q40_100J_G4_4	84,622.12	85,089	85,150.30	93.46	85,089
Q40_100J_G4_5	95,385.83	95,961	96,038.10	85.99	95,961
Q40_100J_G5_1	61,353.79	61,698	61,760.10	104.66	61,698
Q40_100J_G5_2	53,238.81	54,039	54,141.50	123.20	53,968
Q40_100J_G5_3	47,573.43	48,431	48,573.00	121.53	48,431
Q40_100J_G5_4	56,602.82	56,829	56,912.30	149.38	56,829
Q40_100J_G5_5	45,414.25	46,914	47,482.80	141.03	46,914
Q40_100J_G6_1	28,481.90	29,549	29,638.30	156.44	29,549

Table 13 continued

Instance	LB	Min	Avg	Time (s)	BKS
Q40_100J_G6_2	14,364.25	15,511	15,526.20	117.82	15,511
Q40_100J_G6_3	30,386.06	31,579	31,722.90	131.74	31,579
Q40_100J_G6_4	9202.30	10,339	10,358.00	129.65	10,339
Q40_100J_G6_5	18,256.67	19,147	19,165.00	165.29	19,147
Q40_100J_G7_1	89,822.18	90,452	90,484.20	103.96	90,452
Q40_100J_G7_2	59,880.09	60,536	60,714.40	96.80	60,377
Q40_100J_G7_3	86,484.81	87,341	87,370.70	109.89	87,341
Q40_100J_G7_4	74,908.15	75,441	75,690.90	92.40	75,426
Q40_100J_G7_5	68,383.69	69,176	69,272.50	99.24	69,084
Q40_100J_G8_1	26,097.73	27,559	27,578.00	113.81	27,559
Q40_100J_G8_2	49,530.86	50,868	51,002.80	115.92	50,868
Q40_100J_G8_3	1114.98	38,922	39,336.30	98.82	38,922
Q40_100J_G8_4	552.60	41,660	41,836.60	92.45	41436
Q40_100J_G8_5	599.01	43,238	43,741.90	109.06	43,238
Q40_100J_G9_1	600.07	17,952	18,022.40	126.85	17,952
Q40_100J_G9_2	15,518.21	16,724	16,829.00	159.60	16,724
Q40_100J_G9_3	1137.03	24,371	24,472.40	127.36	24,364
Q40_100J_G9_4	363.94	4169	4169.00	119.67	4169
Q40_100J_G9_5	575.10	16,213	16,330.00	131.78	16,197

References

Almeder, C., Mönch, L.: Metaheuristics for scheduling jobs with incompatible families on parallel batching machines. J. Oper. Res. Soc. **62**(12), 2083–2096 (2011)

Balasubramanian, H., Mönch, L., Fowler, J., Pfund, M.: Genetic algorithm based scheduling of parallel batch machines with incompatible job families to minimize total weighted tardiness. Int. J. Prod. Res. **42**(8), 1621–1638 (2004)

Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Sterna, M., Weglarz, J.: Handbook on Scheduling. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-99849-7

Brucker, P., Gladky, A., Hoogeveen, H., Kovalyov, M.Y., Potts, C.N., Tautenhahn, T., van de Velde, S.L.: Scheduling a batching machine. J. Sched. **1**(1), 31–54 (1998)

Chang, P.C., Wang, H.M.: A heuristic for a batch processing machine scheduled to minimise total completion time with non-identical job sizes. Int. J. Adv. Manuf. Technol. **24**(7), 615–620 (2004)

Chiang, T.C., Cheng, H.C., Fu, L.C.: A memetic algorithm for minimizing total weighted tardiness on parallel batch machines with incompatible job families and dynamic job arrival. Comput. Oper. Res. **37**(12), 2257–2269 (2010)

Chou, F.D., Wang, H.M.: Scheduling for a single semiconductor batch-processing machine to minimize total weighted tardiness. J. Chin. Inst. Ind. Eng. **25**(2), 136–147 (2008)

Chou, F.D., Chang, P.C., Wang, H.M.: A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem. Int. J. Adv. Manuf. Technol. **31**(3), 350–359 (2006)

Erramilli, V., Mason, S.J.: Multiple orders per job batch scheduling with incompatible jobs. Ann. Oper. Res. **159**(1), 245–260 (2008)

Gonçalves, J.F., Resende, M.G.C.: Biased random-key genetic algorithms for combinatorial optimization. J. Heur. **17**(5), 487–525 (2011). https://doi.org/10.1007/s10732-010-9143-1

Graham, R., Lawler, E., Lenstra, J., Kan, A.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: PL Hammer EJ, Korte B (eds) Discrete Optimization II Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium Co-sponsored by IBM Canada and SIAM Banff, Aha and Vancouver, Annals of Discrete Mathematics, vol. 5, pp. 287–326. Elsevier (1979)

Grosso, A., Croce, F.D., Tadei, R.: An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem. Oper. Res. Lett. **32**(1), 68–72 (2004)

Hansen, P., Mladenović, N., Brimberg, J., Pérez, J.: Handbook of Metaheuristics. Springer, Chap Variable Neighborhood Search, pp. 61–86 (2010)

Hulett, M., Damodaran, P., Amouie, M.: Scheduling non-identical parallel batch processing machines to minimize total weighted tardiness using particle swarm optimization. Comput. Ind. Eng. **113**, 425–436 (2017)

Kohn, R.: A Framework for Batch Scheduling with Variable Neighborhood Search in Wafer Fabrication. Ph.D. Thesis, Universität der Bundeswehr München (2015)

Kress, D., Barketau, M., Pesch, E.: Single-machine batch scheduling to minimize the total setup cost in the presence of deadlines. J. Sched. **21**(6), 595–606 (2018)

Kurz, M.E., Mason, S.J.: Minimizing total weighted tardiness on a batch-processing machine with incompatible job families and job ready times. Int. J. Prod. Res. **46**(1), 131–151 (2008)

Lee, C.Y.: Minimizing makespan on a single batch processing machine with dynamic job arrivals. Int. J. Prod. Res. **37**(1), 219–236 (1999)

Lee, C.Y., Uzsoy, R., Martin-Vega, L.A.: Efficient algorithms for scheduling semiconductor burn-in operations. Oper. Res. **40**(4), 764–775 (1992)

Li, S., Li, G., Wang, X., Liu, Q.: Minimizing makespan on a single batching machine with release times and non-identical job sizes. Oper. Res. Lett. **33**(2), 157–164 (2005)

Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search: framework and applications. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, pp. 129–168. Springer International Publishing, Cham (2019)

Lu, S., Feng, H., Li, X.: Minimizing the makespan on a single parallel batching machine. Theoret. Comput. Sci. **411**(7), 1140–1145 (2010)

Martins, I.C., Pinheiro, R.G., Protti, F., Ochi, L.S.: A hybrid iterated local search and variable neighborhood descent heuristic applied to the cell formation problem. Expert Syst. Appl. **42**(22), 8947–8955 (2015)

Mathirajan, M., Sivakumar, A.: A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. Int. J. Adv. Manuf. Technol. **29**(9), 990–1001 (2006)
Mathirajan, M., Bhargav, V., Ramachandran, V.: Minimizing total weighted tardiness on a batch-processing machine with non-agreeable release times and due dates. Int. J. Adv. Manuf. Technol. **48**(9), 1133–1148 (2010)
Mönch, L., Roob, S.: A matheuristic framework for batch machine scheduling problems with incompatible job families and regular sum objective. Appl. Soft Comput. **68**, 835–846 (2018)
Mönch, L., Balasubramanian, H., Fowler, J.W., Pfund, M.E.: Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. Comput. Oper. Res. **32**(11), 2731–2750 (2005)
Mönch, L., Fowler, J., Dauzère-Pérès, S., Mason, S., Rose, O.: A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. J. Sched. **14**(6), 583–599 (2011)
Penna, P.H.V., Subramanian, A., Ochi, L.S.: An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. J. Heur. **19**(2), 201–232 (2013)
Perez, I.C., Fowler, J.W., Carlyle, W.: Minimizing total weighted tardiness on a single batch process machine with incompatible job families. Comput. Oper. Res. **32**(2), 327–341 (2005)
Pessoa, A., Bulhões, T., Nesello, V., Subramanian, A.: Exact approaches for single machine total weighted tardiness batch scheduling, Working Paper (2020)
Pinedo, M.L.: Scheduling: Theory, Algorithms and Systems. Springer International Publishing, Cham (2016)
Potts, C.N., Kovalyov, M.Y.: Scheduling with batching: a review. Eur. J. Oper. Res. **120**(2), 228–249 (2000)
Potts, C.N., Wassenhove, L.N.V.: Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity. J. Oper. Res. Soc. **43**(5), 395–406 (1992)
Subramanian, A., Farias, K.: Efficient local search limitation strategy for single machine total weighted tardiness scheduling with sequence-dependent setup times. Comput. Oper. Res. **79**, 190–206 (2017)
Subramanian, A., Battarra, M., Potts, C.: An iterated local search heuristic for the single machine total weighted tardiness problem with sequence-dependent setup times. Int. J. Prod. Res. **52**(9), 2729–2742 (2014)
Tangudu, S.K., Kurz, M.E.: A branch and bound algorithm to minimise total weighted tardiness on a single batch processing machine with ready times and incompatible job families. Prod. Plan. Control **17**(7), 728–741 (2006)
Toso, R., Resende, M.: A C++ application programming interface for biased random-key genetic algorithms. Optim. Methods Sotfw. **30**(1), 81–93 (2015)
Velez-Gallego, M., Damodaran, P., Rodríguez, M.: Makespan minimization on a single batch processing machine with unequal job ready times. Int. J. Ind. Eng. Theory Appl. Pract. **18**(10), (2011)
Vélez-Gallego, M.C., Maya, J., Damodaran, P.: Minimizing total weighted tardiness on a batch-processing machine with non-identical job sizes and non-zero ready times. In: IIE Annual Conference on Proceedings, Institute of Industrial and Systems Engineers (IISE), p. 1 (2011)
Vepsalainen, A.P.J.: Priority rules for job shops with weighted tardiness costs. Manage. Sci. **33**(8), 1035–1047 (1987)
Wang, H.M.: Solving single batch-processing machine problems using an iterated heuristic. Int. J. Prod. Res. **49**(14), 4245–4261 (2011)
Wang, H.M., Chou, F.D.: Bi-criteria single batch-processing machine with job release time and non-identical job sizes. Int. J. Inf. Educ. Technol. **3**(5), 536 (2013)
Xu, R., Chen, H., Li, X.: Makespan minimization on single batch-processing machine via ant colony optimization. Comput. Oper. Res. **39**(3), 582–593 (2012)
Zee, D.J.V.D.: Dynamic scheduling of batch-processing machines with non-identical product sizes. Int. J. Prod. Res. **45**(10), 2327–2349 (2007)
Zhou, S., Chen, H., Xu, R., Li, X.: Minimising makespan on a single batch processing machine with dynamic job arrivals and non-identical job sizes. Int. J. Prod. Res. **52**(8), 2258–2274 (2014)
Zinouri, N., Muskeyvalley, R., Damodaran, P., Ghrayeb, O.: Scheduling a batch processing machine to minimize total weighted tardiness. In: 62nd IIE Annual Conference and Expo 2012, pp. 692–701. Institute of Industrial and Systems Engineers (IISE), Institute of Industrial Engineers (2012)

Affiliations

Eduardo Queiroga[1] · Rian G. S. Pinheiro[2] · Quentin Christ[3] · Anand Subramanian[4] · Artur A. Pessoa[5]

✉ Anand Subramanian
anand@ci.ufpb.br

Eduardo Queiroga
eduardoqueiroga@id.uff.br; eduardovqueiroga@gmail.com

Rian G. S. Pinheiro
rian@ic.ufal.br

Quentin Christ
quentin.christ@st.com

Artur A. Pessoa
artur@producao.uff.br

1 Instituto de Computação, Universidade Federal Fluminense, Rua Passo da Pátria, São Domingos, Niterói, RJ 24210-240, Brazil

2 Instituto de Computação, Universidade Federal de Alagoas, Avenida Lourival Mota, Cidade Universitária, Maceió, AL, Brazil

3 Department of Manufacturing Sciences and Logistics, CMP, Ecole des Mines de Saint-Etienne, CNRS UMR 6158 LIMOS, 13541 Gardanne, France

4 Departamento de Sistemas de Computação, Centro de Informática, Universidade Federal da Paraíba, Rua dos Escoteiros, Mangabeira, João Pessoa, PB 58055-000, Brazil

5 Departamento de Engenharia de Produção, Universidade Federal Fluminense, Rua Passo da Pátria, São Domingos, Niterói, RJ 24210-240, Brazil