IEEE SSCI2011

APRIL 11-15, 2011
PARIS, FRANCE

SYMPOSIUM SERIES ON COMPUTATIONAL INTELLIGENCE

www.ieee-ssci.org

CIPLS 2011

2011 IEEE Workshop on
Computational Intelligence in Production and Logistics Systems

Table of Contents
Technical Sessions
Author Index

ISBN: 978-1-61284-332-2
IEEE Catalog Number: CFP1144N-CDR

Technical Support:
Chris Dyer
Conference Catalysts, LLC
Phone: +1 785 341 3583
cdyer@conferencecatalysts.com

IEEE
IEEE Computational Intelligence Society

Organized and sponsored by the IEEE Computational Intelligence Society

2011 IEEE

2011 IEEE Workshop On Computational Intelligence In Production And Logistics Systems

(CIPLS 2011) Proceedings

Additional copies may be ordered from:
IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08855-1331 USA

+1 800 678 IEEE (+1 800 678 4333)
+1 732 981 1393
+1 732 981 9667 (FAX)
email: customer-service@ieee.org

IEEE Catalog Number: CFP1144N-CDR
ISBN: 978-1-61284-332-2

TABLE OF CONTENTS

15:20 - 16:30

PS11: CIPLS - 2011

Poster Session

IEEE CIPLS 2011 Committee

Workshop on Computational Intelligence in Production and Logistics Systems (IEEE CIPLS 2011)

The management of production and logistics systems in today's fierce competition environment is a difficult task and has become progressively complex. Major changes in products, processes, technologies, and societies bring along remarkable challenges and increasing market demands. Modelling and optimisation of the complex problems arising in production and logistics systems is of paramount importance in surviving and achieving competitive gains in productivity and quality. In recent years, the advancements in computer technology have allowed researchers to tackle large-scale problems and to develop and integrate efficient optimisation techniques for solving them. Within this context, CIPLS aims at addressing issues related to the design, planning, control, and continuous improvement of production and logistics systems using computational intelligence, including local search methods, evolutionary algorithms and other nature-inspired optimisation techniques. The intention is to cover various aspects of production from aggregate planning to shop-floor execution systems and modelling, planning and control of logistics systems. Studies incorporating real-world applications are highly encouraged.

Workshop Co-Chairs
Bülent Çatay, Sabanci University, Turkey
Raymond Chiong, Swinburne University of Technology, Australia
Patrick Siarry, Université Paris XII Val de Marne, France

Program Committee
Tolga Bektas, University of Southampton, UK
Héctor Cancela, University of the Republic, Uruguay
Maurice Clerc, http://mauriceclerc.net, France
Oscar Cordón, European Centre for Soft Computing, Spain
Moussa Diaf, University of Tizi-Ouzou, Algeria
Deniz Türsel Eliiyi, Izmir University of Economics, Turkey
Mourad Fakhfakh, University of Sfax, Tunisia
Martin Grunow, Technische Universität München, Germany
Joerg Laessig, International Computer Science Institute (ICSI), UC Berkeley, USA
Guohua Ma, Wentworth Institute of Technology, USA
Zbigniew Michalewicz, University of Adelaide, Australia
Nicolas Monmarché, University of Tours, France
Luc Muyldermans, University of Nottingham, UK
Antonio J Nebro, University of Málaga, Spain
Ceyda Oguz, Koc University, Turkey
Erwin Pesch, Universität Siegen, Germany
Anna Piwońska, Technical University of Bialystok, Poland
Ruhul A Sarker, University of New South Wales, Australia
Özgür Toy, Turkish Naval Academy, Turkey
Joaquín Bautista Valhondo, Technical University of Catalonia, Spain
Thomas Weise, University of Science and Technology of China, China

A Bounded Dynamic Programming algorithm for the Blocking Flow Shop problem

Joaquín Bautista, Alberto Cano
Nissan Chair, Barcelona
Escola Tècnica Superior d'Enginyeria de Barcelona.
Universitat Politècnica de Catalunya
Barcelona, Spain
joaquin.bautista@upc.edu, alberto.cano-perez@upc.edu

Ramon Companys, Imma Ribas
Departament d'Organització d'Empreses
Escola Tècnica Superior d'Enginyeria de Barcelona.
Universitat Politècnica de Catalunya
Barcelona, Spain
ramon.companys@upc.edu, imma.ribas@upc.edu

***Abstract*— We present some results attained with two variants of the bounded dynamic programming algorithm to solve the Fm|block|C_{max} problem using as experimental data the well-known Taillard instances. We have improved the best-known solutions for four of the Taillard's instances.**

Keywords: Scheduling, BDP algorithms, blocking flow shop

I. INTRODUCTION

This work deals with the permutation flow-shop scheduling problem without storage space between stages. If there is enough storage space between machine *j* and machine *j*+1, the job *i* can wait there for the next operation, machine *j* is released and can work on another job. But, if there is no storage space between stages, then intermediate queues of jobs waiting in the system for their next operation are not allowed. If operation on machine *j* for a job *i* is finished and the next machine, *j*+1, is still busy on the previous job, the completed job *i* has to be blocked into machine *j*. For simplicity purposes we call BFSP (*Blocking Flow Shop Problem*) the problem considered and PFSP (*Permutation Flow Shop Problem*) the equivalent case with unlimited storage space.

The most common criterion, here considered, is the minimization of the makespan or maximum completion time. Using the notation proposed by [1] the problem BFSP is denoted as Fm|block|C_{max} (and the PFSP as Fm|prmu|C_{max}).

Reference [2] published a review on flow shop with blocking and no-wait in-process. If the number of machines is two, in [3] is showed that the problem F2|block|C_{max} can be reduced to a travelling salesman problem (TSP) with $n + 1$ towns (0, 1, 2, ..., n) which can be solved in polynomial time using the algorithm proposed in [4,5] . The sequence of towns in an optimal path corresponds to an optimal permutation for the original problem. Reference [2] showed, using results from [6], that Fm|block|C_{max} problem for $m \geq 3$ machines is strongly NP-hard. In [7] a branch-and-bound algorithm, which uses an elaborated lower bound, is presented. In [8], a competitive double branch-and-bound algorithm, which uses the reversibility property of the problem, is proposed. Both references [7,8] reported solutions for the well known Taillard's benchmarks [9].

Since the problem is NP-hard several heuristic procedures for Fm|block|C_{max} has been presented. Reference [10] studied a special case and proposed the constructive profile fitting heuristic to solve it. Reference [11] adapted some procedures proposed for the PFSP and concludes that NEH heuristic, proposed by [12], has a good behavior. Reference [13] proposed four heuristics procedures, two of them based on NEH heuristic, [14] proposed a genetic algorithm (GA). Reference [15] presented two tabu search algorithms (TS), which are evaluated using the solutions obtained by [7]. Reference [16] proposed a hybrid genetic algorithm (HGA), [17] proposed an algorithm based on particle swarm optimization (HPSO), [18] proposed an algorithm based on differential optimization (DE). More recently, [19] proposed an hybrid discrete differential evolution algorithm (HDDE) and compared their results with the obtained by the tabu search algorithm proposed in [15]. Finally, in [20], an iterated greedy algorithm (IGA), which outperforms the HDDE, is presented. In this reference it can be found an updated list of best known solutions for the Taillard's benchmark.

For this manuscript, we used a procedure based on BDP (*Bounded Dynamic Programming*). This procedure combines features of dynamic programming (determination of extreme paths in graphs) with features of branch and bound algorithms. The principles of Bounded Dynamic Programming have been described by [21] and [22]. Previous work on similar approaches has been done by [23] and [24], and extended by [25].

This manuscript is organized as follows. Section II presents the problem description. Section III describes the graph associated with the problem under consideration and establishes dominance properties between their vertices. Section IV proposes overall and partial bounds on the C_{max} value shown by the sequences. Section V introduces a procedure based on BDP to solve the problem under consideration. Section VI describes the computational experiments performed and presents the results. Finally, Section VII shows the conclusions of the study.

II. PROBLEM DESCRIPTION

At time zero, n jobs must be processed, in the same order, on each of m machines. Each job goes from machine 1 to machine m. The processing time for each operation is $p_{i,k}$,

where $k \in K = \{1, 2,.., m\}$ denotes a machine and $i \in I = \{1, 2,.., n\}$ a job. Setup times are included in processing times. These times are fixed, known in advance and positive. The objective function considered is the minimization of the makespan ($C_{\max}$).

Given a permutation, $\boldsymbol{\pi}$, of the n jobs, $[t]$ indicates the job that occupies position t in the sequence. For example, in $\boldsymbol{\pi}$ = (3, 1, 2) [1] = 3, [2] = 1, [3] = 2. For this permutation, in every machine, job 2 occupies position 3. In a feasible schedule associated to a permutation, let $s_{k,t}$ be the beginning of the time destined in machine k to job that occupies position t and $e_{k,t}$ the time of the job that occupies position t releases machine k. The Fm|prmu|$C_{\max}$ problem can be formalized as follows:

$$s_{k,t} + p_{[t],k} \le e_{k,t} \qquad k=1,2,..,m \qquad t=1,2,..,n \qquad (1)$$

$$s_{k,t} \ge e_{k,t-1} \qquad k=1,2,..,m \qquad t=1,2,..,n \qquad (2)$$

$$s_{k,t} \ge e_{k-1,t} \qquad k=1,2,..,m \qquad t=1,2,..,n \qquad (3)$$

$$C_{\max} = e_{m,n} \qquad (4)$$

Being $e_{k,0} = 0 \ \forall k$, $e_{0,t} = 0 \ \forall t$, the initial conditions.

The schedule is semi-active if equation (1) is written as $s_{k,t} + p_{[t],k} = e_{k,t}$ and equations (2) and (3) are summarized as $s_{k,t} = \max\{e_{k,t-1}, e_{k-1,t}\}$.

When there is no storage space between stages, Fm|block|$C_{\max}$ problem, if a job i finishes its operation on a machine k and if the next machine, $k + 1$, is still busy on the previous job, the completed job i has to remain on the machine k blocking it. This condition requires an additional equation (5) in the formulation of the problem.

$$e_{k,t} \ge e_{k+1,t-1} \qquad k=1,2,..,m \qquad t=1,2,..,n \qquad (5)$$

The initial condition $e_{m+1,t} = 0 \quad t=1,2,..,n$ must be added.

The schedule obtained is semi-active if equation (1) and (5) is summarized as (6):

$$e_{k,t} = \max\{s_{k,t} + p_{[t],k}, e_{k+1,t-1}\} \quad \forall k, \forall t \qquad (6)$$

Consequently, the Fm|prmu|$C_{\max}$ problem can be seen as a relaxation of the Fm|block|$C_{\max}$ problem.

A mathematical formulation associated with the problem Fm|prmu|$C_{\max}$ can be found in [26]. For the Fm|block|$C_{\max}$, the restrictions resulting from the expression (5) should be added to this formulation.

III. Graph associated with the problem

Similar to [22] and [27], we can build a linked graph without loops or direct cycles of T+1 stages. The set of vertices in level t ($t = 0,...,T$) will be noted as $J(t)$. $J(t, j)$ ($j = 1,...,|J(t)|$) being a vertex of level t, which is defined by the triad ($\vec{q}(t, j), \vec{e}(t, j), C_{\max}(t, j)$), where:

- $\vec{q}(t, j) = (q_1(t, j),..., q_n(t, j))$, where $q_i(t, j) \ \forall i$ takes the value 1 if the job i has been completed and the value 0 in the opposite case.
- $\vec{e}(t, j) = (e_1(t, j),..., e_m(t, j))$ represents the vector of completion times of the operations at the stations.
- $C_{\max}(t, j)$ represents the completion time of the last programmed job, at stage t and vertex j.

The vertex $J(t, j)$ has the following properties:

$$\sum_{i=1}^{n} q_i(t, j) = t \qquad (7)$$

$$q_i(t, j) \in \{0,1\} \ \ \forall i \qquad (8)$$

$$C_{\max}(t, j) = e_m(t, j) \qquad (9)$$

In short, a vertex $J(t, j)$ will be represented as follows:

$$J(t, j) = \{(t, j), \vec{q}(t, j), \vec{e}(t, j)\} \qquad (10)$$

At level 0 of the graph, there is only one $J(0)$ vertex. Initially, we may consider that at level t, $J(t)$ contains the vertices associated with all of the sub-sequences that can be built with t jobs that satisfy properties (7) and (8). However, it is easy to reduce the cardinal that $J(t)$ may present a priori, establishing the following relationship of dominance and equivalence:

$$J(t,j) \prec J(t,j') \Leftrightarrow [\vec{q}(t,j) = \vec{q}(t,j')] \wedge [\vec{e}(t,j) \prec \vec{e}(t,j')] \qquad (11)$$

$$J(t,j) \equiv J(t,j') \Leftrightarrow [\vec{q}(t,j) = \vec{q}(t,j')] \wedge [\vec{e}(t,j) = \vec{e}(t,j')] \qquad (12)$$

With these relationships, we can reduce the search space for solutions in the graph. Therefore, at level t of the graph, $J(t)$ will contain the vertices associated with non-dominated and non-equivalent sub-sequences, and at level T, $J(T)$ will contain all the vertices associated with non-equivalent and non-dominated completed sequences.

A transition arc through the type of job i exists between vertices $J(t, j)$ of level t and vertex $J(t+1, j_i)$ of level t+1 in the following case:

$$\vec{q}(t, j) \prec \vec{q}(t+1, j_i) \qquad (13)$$

For vertex $J(t+1, j_i)$ to be completely defined through the transition from $J(t, j)$, it is necessary to determine:

$$J(t+1, j_i) = \{(t+1, j_i), \vec{q}(t+1, j_i), \vec{e}(t+1, j_i)\} \qquad (14)$$

as follows:

$$q_i(t+1, j_i) = 1 \qquad (15)$$

$$q_h(t+1, j_i) = q_h(t, j) \quad \forall h : h \neq i \qquad (16)$$

$$e_k(t+1, j_i) = \max\begin{Bmatrix} e_k(t, j) + p_{i,k}, \\ e_{k-1}(t+1, j_i), \\ e_{k+1}(t, j) \end{Bmatrix} \quad \forall k \in K \qquad (17)$$

where $e_0(t+1, j_i) = 0$.

Indirectly, contribution to the partial $C_{\max}$ generated in the transition from $J(t, j)$ to $J(t+1, j_i)$ may be calculated by incorporating the job *i* to the latter vertex, as follows:

$$a((t, j) \rightarrow (t+1, j_i)) = e_m(t+1, j_i) - e_m(t, j) \tag{18}$$

Under these conditions, finding a sequence that optimises the total $C_{\max}$ is equivalent to finding an optimum path from vertex $J(0)$ to the set of vertices $J(T)$ of stage *T*.

Therefore, any algorithm of extreme paths in the graphs is valid for finding solutions to the proposed problem. However, realistic industrial problems where *n* and *m* are large give rise to graphs with a large number of vertices. Therefore, we recommend resorting to procedures that do not explicitly require the presence of all of the vertices for calculation.

IV. Bounding the Values of the Sequences

First, we establish overall bounds for $C_{\max}$, and then we establish the bounds associated with the path for building (complement) when a segment or subsequence of *t* members has already been built.

A. General bounds for C_{max}

If we account for the machines independently, then we can write the following:

$$LB1(k) = \sum_{i=1}^{n} p_{i,k} + \\ + \min_{(i,h)\in I:\, i\neq h} \left\{ \sum_{k'=1}^{k-1} p_{i,k'} + \sum_{k'=k+1}^{m} p_{h,k'} \right\} \forall k \in K \tag{19}$$

which is a bound of $C_{\max}$, through the machine *k*.

Therefore, considering all machines, we have the following:

$$LB1 = \max_{k\in K} \{ LB1(k) \} \tag{20}$$

In the same manner, we can also consider a bound for $C_{\max}$ through the job *i*:

$$LB2(i) = \sum_{k=1}^{m} p_{i,k} + \sum_{h\in I: h\neq i} \min_{k\in K} \{ p_{h,k} \} \; \forall i \in I \tag{21}$$

Considering all the jobs, then we have:

$$LB2 = \max_{i\in I} \{ LB2(i) \} \tag{22}$$

B. Bound of C_{max} through a given segment

Reference [26] proposed a directed graph for the computation of the makespan on the Fm|block|$C_{\max}$ problem, where $C_{\max}$ is the length of the critical path. The length of any path from the initial node to the final one is a lower bound of the critical path length. We use in our bound the simplest paths proposed by [28] in his pioneering work.

Let us assume that we have built a path from $J(0)$ to vertex $J(t, j)$, and thus we have the information $\vec{q}(t, j)$ and $\vec{e}(t, j)$.

To complete a sequence up to stage *T*, we will need to link with $J(t, j)$, *T-t* vertices, associated each of them with a different unscheduled job.

Under these conditions, we can delimit $C_{\max}$ through the vertex $J(t, j)$ adapting the overall bound *LB1*.

$$LB1(t, j) = \max_{k\in K} \left\{ \begin{array}{l} e_k(t, j) + \sum_{\substack{i\in I: \\ q_i(t,j)=0}} p_{i,k} + \\ + \min_{\substack{i\in I: \\ q_i(t,j)=0}} \left\{ \sum_{k'=k+1}^{m} p_{i,k'} \right\} \end{array} \right\} \tag{23}$$

If we focus on the jobs, we will have:

$$LB2(t, j) = e_1(t, j) + \max_{\substack{i\in I: \\ q_i(t,j)=0}} \left\{ \begin{array}{l} \sum_{k=1}^{m} p_{i,k} + \\ + \sum_{\substack{h\in I-\{i\}: \\ q_h(t,j)=0}} \min_{k\in K} \{ p_{h,k} \} \end{array} \right\} \tag{24}$$

V. The Use of Bounded Dynamic Programming

The procedure we propose (from [22], [27] and [29]) is called BDP (*bounded dynamic programming*) and consists of generating a part of the graph described in section III from level 0 to level *T*, one level at a time.

The generated vertices may potentially form a part of an optimum path (from 0 to *T*) that is based on the construction of an optimum segment of *t* stages, from $J(0)$ to $J(t, j)$, and on the evaluation of the bound of $C_{\max}$ to reach stage *T*, for example $LB1(t, j)$.

The procedure only keeps the information of two consecutive stages in memory, *t* and *t*+1 ($t = 0, ..., T-1$), for which it uses the following lists $\Lambda(t)$ and $\Lambda(t+1)$, respectively:

- List $\Lambda(t)$ contains information about the vertices consolidated in stage *t* that can potentially form part of an optimum or good quality path.
- List $\Lambda(t+1)$ contains the vertices that are tentatively generated one-by-one from each vertex of list through the possible transitions between stages *t* and *t*+1.

A record $\lambda(J(t, j))$ of list $\Lambda(t)$, $\lambda(J(t, j)) \in \Lambda(t)$, is composed of three elements:

$$\lambda(J(t, j)) = \{ J(t, j), LB1(t, j), \Gamma^{-}(J(t, j)) \} \tag{25}$$

where $\Gamma^{-}(J(t, j))$ is the vertex of stage *t*-1 ancestor of $J(t, j)$.

Although the use of $\Lambda(t)$ and $\Lambda(t+1)$ notably reduces memory needs, the number of vertices that can be generated for a stage can be very large. Therefore, we impose a limitation on

the number of $H(t)$ vertices stored in stage *t*. This limitation, called window width, is represented as *H*, $H(t) \le H$ $(t = 1, ..., T)$. In addition, we set the maximum number of transitions from a vertex in stage *t* to the value *n-t*.

Evidently, some vertices tentatively generated in stage *t* will not be recorded in list $\Lambda(t+1)$, as described in [27].

To obtain an initial solution with value Z_0 (the upper bound of the value of $C_{\max}$), it is sufficient to use a Greedy procedure, a local search, or *BDP* with a small window width, e.g., $H = 1$.

We have developed two variants based on BDP:

1. The ordered pair of values $(LB1(t,j), e_m(t,j))$ is used as priority rule or guide (*GZ*) to obtain solutions. A partial solution is more promising than another when it has a best bound for $C_{\max}$. In case of tie between two partial solutions (equal bound for $C_{\max}$), the partial solution with less $e_m(t,j)$ will be considered the best.
2. In the variant 2, the ordered pair of values $(e_m(t,j), LB1(t,j))$ is used as priority rule or guide (*GZ*). A partial solution is more promising than another when it has less value for his partial $C_{\max}$. In case of tie between two partial solutions (equal partial $C_{\max}$), the partial solution with less bound for total $C_{\max}$ will be considered the best.

Under these conditions, we can write the following algorithm (Variant 1 and 2):

BDP-Fm|block|C_{max}
Input: $T, n, m, p_{i,k} (i = 1,...,n; k = 1,...,m), Z_0, H$
Output: $\Lambda(T)$
Initialization: $t = 0; \Lambda(t) = \{J(0), LB1, -\}; LBZ_{\min} = \infty$
while ($t < T$) *do*
  $\Lambda(t+1) = \{\varnothing\}; j = 1$
  while ($j \le H(t)$) *do*
    i=1
    while ($i \le n$) *do*
      if $q_i(t,j) = 0$ *then*
        generate vertex $J(t+1, j_i)$ (see (15), (16) and (17))
        determine $LB1(t+1, j_i)$ (*see* (23))
        $LBZ = LB1(t+1, j_i)$
        Variant 1:
          $GZ = (LB1(t+1, j_i), e_m(t+1, j_i))$
        Variant 2:
          $GZ = (e_m(t+1, j_i), LB1(t+1, j_i))$
        if $LBZ \ge Z_0$ *then*
          remove $J(t+1, j_i)$
        else if $\left\{ \begin{array}{l} \exists \lambda(J(t+1,h)) \in \Lambda(t+1): \\ J(t+1,h)(\prec \vee \equiv) J(t+1, j_i) \end{array} \right\}$ *then*
          reject $J(t+1, j_i)$
        else
          Let $\left\{ \begin{array}{l} L(t+1) = \{\lambda(t+1,h) \in \Lambda(t+1): \\ \quad J(t+1, j_i) \prec J(t+1,h)\} \end{array} \right\}$
          if $L(t+1) \ne \{\varnothing\}$ *then*
            $\Lambda(t+1) \leftarrow \Lambda(t+1) - L(t+1);$
            $H(t+1) \leftarrow H(t+1) - |L(t+1)|$
        end if
        if $H(t+1) < H$ *then*
          $\Lambda(t+1) \leftarrow \Lambda(t+1) +$
          $+\{J(t+1, j_i), LB1(t+1, j_i), J(t,j)\}$
          $H(t+1) \leftarrow H(t+1) + 1$
        else
          $g_1 = LB1(t+1,h)$
          $g_2 = e_m(t+1,h)$
          Variant 1:
            $\lambda(t+1, h_{\max}) = \underset{\lambda(t+1,h) \in \Lambda(t+1)}{\arg\max} (g_1, g_2)$
            $GZ_{\max} = (LB1(t+1, h_{\max}), e_m(t+1, h_{\max}))$
          Variant 2:
            $\lambda(t+1, h_{\max}) = \underset{\lambda(t+1,h) \in \Lambda(t+1)}{\arg\max} (g_2, g_1)$
            $GZ_{\max} = (e_m(t+1, h_{\max}), LB1(t+1, h_{\max}))$
          $LBZ_{\max} = LB1(t+1, h_{\max})$
          if $GZ_{\max} \le GZ$ *then*
            discard $J(t+1, j_i)$
          else
            replace $J(t+1, h_{\max})$ with $J(t+1, j_i)$:
            $\Lambda(t+1) \leftarrow \Lambda(t+1) - \lambda(t+1, h_{\max})$
            $\Lambda(t+1) \leftarrow \Lambda(t+1) +$
            $+\{J(t+1, j_i), LB1(t+1, j_i), J(t,j)\}$
            if $LBZ_{\min} > LBZ_{\max}$ *then*
              $LBZ_{\min} = LBZ_{\max}$
            end if
          end if
        end if
      end if
      $i \leftarrow i + 1$
    end while
    $j \leftarrow j + 1$
  end while
  Save $\Lambda(t+1)$; $t \leftarrow t + 1$
end while
Save $\Lambda(T)$
end BDP- Fm|block|C_{max}

When the procedure ends, we can initially find two possible situations:

- List $\Lambda(T)$ is empty, which means that we are unable to find a solution with a value less than Z_0.
- List $\Lambda(T)$ is not empty, which means that the records contained in $\Lambda(T)$, $\lambda(T,h) \in \Lambda(T)$, are associated with vertices, $J(T,h)$, whose $C_{\max}$ is $e_m(T,h) < Z_0$. In this case, we can regressively reconstruct a sequence from any of these vertices with a better value than Z_0 using the $\Lambda(t)$ list and the ancestors of the vertices.

In addition, we can guarantee that we are able to build an optimum sequence from the $\lambda(T,h) \in \Lambda(T) \ne \{\varnothing\}$ records in any of the following cases:

$$\text{Case 1}: \max_{0 \le t \le T} \{H(t)\} < H \tag{26}$$

$$\text{Case 2}: \left(\max_{0 \le t \le T} \{H(t)\} = H \right) \wedge (e_m(T,h) \le LBZ_{\min}) \tag{27}$$

In any other case, the procedure is heuristic.

VI. COMPUTATIONAL EXPERIENCE

We have performed an operation test with the first 11 sets from Taillard's benchmark instances [9]. The Taillard's benchmark instances consist of 120 instances, grouped in 12 sets. Each set has 10 instances, each of them with the same number of jobs and machines. In our computational experience, we have experimented with the first 110 instances from the 120 supplied. The number of jobs goes from 20 (set 1) to 200 (set 11) and the number of machines goes from 5 (set 1) to 20 (set 11).

To obtain solutions, we have used two variants of BDP programmed in C++, compiled with gcc v. 4.01, running on an Apple Macintosh iMac computer with an Intel Core i7 2.93 GHz processor and 8 GB RAM using MAC OS X 10.6.4. Neither the implementation nor the compiler used threads or any type of parallel code; therefore, the computer can be considered a single 2.93 GHz processor. The 110 instances were solved using seven correlative window widths, H_1 to H_7 with values 1, 10, 50, 100, 250, 500 and 750, respectively.

For the initial solution Z_0, we used the value of the solution obtained with the previous width $H_{\alpha-1}$ for each window width H_α ($\alpha = 1,..,7$), except for the case with width $H_1 = 1$ in which Z_0 was fixed at ∞.

To analyze the experimental results, we used the relative percentage deviation (RPD) calculated as follows:

$$RPD = \frac{BDP_{Best} - Best_{solution}}{Best_{solution}} 100 \qquad (28)$$

Tables I to XI shows, for the sets 1 to 11 from Taillard's instances, the best results reported in the literature (*column "Best"*, obtained from [20]) for these instances and the best results obtained ($C_{\max}$) for each window width (H_α) and for each variant (1 and 2) of the BDP algorithm. We also report the best value for RPD for each instance and BDP variant. The best results for both variants for $C_{\max}$ and RPD are show in italics and bold face.

TABLE I. SOLUTIONS FOR TAILLARD'S SET 1

Set 1	Ins.	Best from lit.	BDP Variant 1 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			$C_{\max}$								
n =20	1	1374	1544	1448	1429	1413	1407	1402	1397	1397	1.67
m=5	2	1408	1588	1494	1475	1475	1465	1433	1433	1433	1.78
	3	1280	1455	1365	1326	1312	1312	1311	1311	1311	2.42
	4	1448	1563	1562	1543	1466	1456	1456	1456	***1456***	***0.55***
	5	1341	1512	1424	1400	1369	1367	1357	1350	***1350***	***0.67***
	6	1363	1515	1470	1395	1393	1393	1385	1385	***1385***	***1.61***
	7	1381	1467	1407	1407	1396	1396	1396	1395	***1395***	***1.01***
	8	1379	1608	1524	1392	1392	1386	1386	1386	***1386***	***0.51***
	9	1373	1461	1432	1410	1410	1403	1403	1389	***1389***	***1.17***
	10	1283	1421	1311	1307	1307	1293	1293	1293	***1293***	***0.78***
	Ins.	Best from lit.	BDP Variant 2 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			$C_{\max}$								
	1	1374	1640	1513	1441	1423	1390	1380	1380	***1380***	***0.44***
	2	1408	1725	1549	1474	1459	1450	1442	1432	***1432***	***1.70***
	3	1280	1667	1471	1353	1330	1326	1314	1302	***1302***	***1.72***
	4	1448	1791	1607	1554	1516	1513	1499	1493	1493	3.11
	5	1341	1566	1456	1381	1381	1381	1381	1374	1374	2.46
	6	1363	1690	1504	1439	1414	1414	1409	1409	1409	3.37
	7	1381	1595	1483	1436	1428	1428	1424	1404	1404	1.67
	8	1379	1664	1478	1467	1450	1419	1414	1409	1409	2.18
	9	1373	1712	1539	1475	1454	1443	1439	1412	1412	2.84
	10	1283	1616	1457	1334	1312	1309	1309	1304	1304	1.64

TABLE II. SOLUTIONS FOR TAILLARD'S SET 2

Set 2	Ins.	Best from lit.	BDP Variant 1 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			$C_{\max}$								
n =20	11	1698	2006	1807	1768	1762	1741	1741	1731	***1731***	***1.94***
m=10	12	1833	2116	1974	1974	1909	1909	1897	1895	***1895***	***3.38***
	13	1659	1781	1728	1695	1687	1687	1684	1684	***1684***	***1.51***
	14	1535	1791	1714	1640	1587	1587	1579	1579	***1579***	***2.87***
	15	1617	1978	1780	1738	1707	1707	1667	1667	***1667***	***3.09***
	16	1590	1830	1710	1611	1611	1610	1610	1610	***1610***	***1.26***
	17	1622	1818	1740	1725	1722	1691	1691	1681	***1681***	***3.64***
	18	1731	1904	1790	1766	1762	1756	1756	1756	1756	1.44
	19	1747	1962	1854	1854	1768	1755	1755	1755	***1755***	***0.46***
	20	1782	2100	1933	1922	1890	1829	1829	1829	***1829***	***2.64***
	Ins.	Best from lit.	BDP Variant 2 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			$C_{\max}$								
	11	1698	2210	1973	1940	1908	1870	1858	1842	1842	8.48
	12	1833	2305	2112	2022	2015	1996	1953	1953	1953	6.55
	13	1659	1951	1869	1785	1742	1716	1711	1711	1711	3.13
	14	1535	1983	1755	1711	1700	1674	1660	1639	1639	6.78
	15	1617	2083	1849	1768	1715	1683	1682	1682	1682	4.02
	16	1590	2008	1830	1718	1715	1647	1646	1646	1646	3.52
	17	1622	2046	1782	1733	1712	1692	1692	1692	1692	4.32
	18	1731	2133	1859	1796	1777	1761	1760	1752	***1752***	***1.21***
	19	1747	2035	1860	1838	1832	1822	1815	1810	1810	3.61
	20	1782	2173	2037	1962	1915	1873	1869	1869	1869	4.88

TABLE III. SOLUTIONS FOR TAILLARD'S SET 3

Set 3	Ins.	Best from lit.	BDP Variant 1 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			$C_{\max}$								
n =20	21	2436	2772	2644	2640	2622	2567	2551	2551	***2551***	***4.72***
m=20	22	2234	2760	2544	2429	2367	2350	2326	2315	***2315***	***3.63***
	23	2479	2813	2730	2705	2665	2663	2651	2644	2644	6.66
	24	2348	2733	2480	2440	2429	2419	2403	2388	***2388***	***1.70***
	25	2435	2886	2740	2621	2602	2553	2534	2534	***2534***	***4.07***
	26	2383	2744	2532	2492	2492	2492	2461	2461	***2461***	***3.27***
	27	2390	2827	2635	2584	2575	2543	2532	2531	2531	5.90
	28	2328	2792	2596	2574	2543	2522	2522	2522	***2522***	***8.33***
	29	2363	3036	2570	2545	2494	2494	2483	2483	***2483***	***5.08***
	30	2323	2698	2561	2442	2404	2404	2367	2367	***2367***	***1.89***
	Ins.	Best from lit.	BDP Variant 2 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			$C_{\max}$								
	21	2436	2868	2675	2611	2590	2579	2575	2575	2575	5.71
	22	2234	2853	2671	2544	2517	2441	2404	2404	2404	7.61
	23	2479	2857	2783	2705	2644	2636	2627	2627	***2627***	***5.97***
	24	2348	3048	2606	2523	2511	2456	2431	2431	2431	3.53
	25	2435	2945	2787	2739	2693	2641	2639	2613	2613	7.31
	26	2383	2801	2618	2559	2559	2542	2487	2476	2476	3.90
	27	2390	2956	2715	2672	2603	2603	2550	2518	***2518***	***5.36***
	28	2328	2834	2669	2618	2560	2557	2554	2529	2529	8.63
	29	2363	3130	2706	2610	2565	2560	2544	2530	2530	7.07
	30	2323	2933	2672	2576	2553	2489	2452	2425	2425	4.39

TABLE IV. SOLUTIONS FOR TAILLARD'S SET 4

Set 4	Ins.	Best from lit.	BDP Variant 1 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			$C_{\max}$								
n =50	31	3002	3276	3146	3124	3096	3078	3066	3066	***3066***	***2.13***
m=5	32	3201	3481	3341	3267	3267	3253	3253	3253	***3253***	***1.62***
	33	3011	3235	3146	3108	3081	3081	3081	3081	***3081***	***2.32***
	34	3128	3554	3261	3261	3215	3187	3181	3181	***3181***	***1.69***
	35	3166	3471	3257	3226	3226	3226	3226	3216	***3216***	***1.58***
	36	3169	3530	3365	3360	3317	3317	3284	3284	***3284***	***3.63***
	37	3013	3383	3188	3124	3098	3096	3096	3096	***3096***	***2.75***
	38	3073	3480	3180	3125	3125	3125	3125	3125	***3125***	***1.69***
	39	2908	3256	3107	3076	3005	3004	2986	2971	***2971***	***2.17***
	40	3120	3434	3277	3217	3182	3171	3171	3163	***3163***	***1.38***
	Ins.	Best from lit.	BDP Variant 2 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			$C_{\max}$								
	31	3002	3613	3366	3204	3160	3131	3111	3111	3111	3.63
	32	3201	3815	3509	3484	3426	3353	3296	3269	3269	2.12
	33	3011	3678	3301	3211	3192	3168	3168	3156	3156	4.82
	34	3128	3812	3538	3342	3308	3308	3285	3285	3285	5.02
	35	3166	3901	3492	3353	3327	3286	3242	3242	3242	2.40
	36	3169	3729	3468	3412	3349	3293	3293	3290	3290	3.82
	37	3013	3516	3289	3138	3138	3097	3097	3097	3097	2.79
	38	3073	3595	3370	3268	3223	3195	3159	3159	3159	2.80
	39	2908	3393	3078	3069	3042	3032	2995	2984	2984	2.61
	40	3120	3759	3553	3461	3406	3367	3341	3301	3301	5.80

TABLE V. SOLUTIONS FOR TAILLARD'S SET 5

Set 5	Ins.	Best from lit.	BDP Variant 1 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
n =50	41	3638	4139	3911	3837	3744	3744	3730	3730	***3730***	***2.53***
m=10	42	3507	3914	3701	3701	3647	3624	3624	3585	***3585***	***2.22***
	43	3488	3982	3848	3754	3754	3672	3642	3623	***3623***	***3.87***
	44	3656	4050	4024	3869	3869	3869	3869	3869	***3869***	***5.83***
	45	3629	4109	3836	3761	3761	3761	3720	3712	***3712***	***2.29***
	46	3621	3997	3845	3743	3743	3743	3692	3692	***3692***	***1.96***
	47	3696	4204	3940	3890	3814	3814	3814	3814	***3814***	***3.19***
	48	3572	4113	3986	3867	3718	3718	3718	3718	***3718***	***4.09***
	49	3532	3871	3742	3682	3660	3660	3636	3619	***3619***	***2.46***
	50	3624	4260	3987	3905	3905	3812	3812	3812	3812	5.19
	Ins.	Best from lit.	BDP Variant 2 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
	41	3638	4303	4058	3972	3920	3868	3853	3853	3853	5.91
	42	3507	4294	3913	3899	3804	3750	3723	3696	3696	5.39
	43	3488	4254	3922	3789	3733	3652	3652	3652	3652	4.70
	44	3656	4463	4165	4073	3991	3910	3910	3895	3895	6.54
	45	3629	4591	4152	3923	3841	3825	3776	3769	3769	3.86
	46	3621	4360	4102	3996	3963	3902	3876	3824	3824	5.61
	47	3696	4556	4246	4074	4020	3966	3966	3927	3927	6.25
	48	3572	4281	3966	3964	3929	3827	3827	3827	3827	7.14
	49	3532	4374	4170	3954	3896	3779	3779	3779	3779	6.99
	50	3624	4455	4011	3940	3883	3859	3811	3796	***3796***	***4.75***

TABLE VI. SOLUTIONS FOR TAILLARD'S SET 6

Set 6	Ins.	Best from lit.	BDP Variant 1 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
n =50	51	4500	5213	4899	4844	4753	4753	4741	4705	***4705***	***4.56***
m=20	52	4276	5163	4960	4811	4720	4700	4700	4668	***4668***	***9.17***
	53	4289	5258	4879	4553	4553	4553	4553	4553	***4553***	***6.16***
	54	4377	5010	4663	4649	4643	4615	4572	4572	***4572***	***4.46***
	55	4268	5291	4888	4669	4669	4624	4594	4542	***4542***	***6.42***
	56	4280	5039	4876	4689	4651	4628	4596	4596	***4596***	***7.38***
	57	4308	5110	4853	4636	4636	4573	4505	4505	***4505***	***4.57***
	58	4326	5395	4836	4689	4627	4590	4544	4494	***4494***	***3.88***
	59	4316	5261	5044	4780	4780	4780	4732	4687	***4687***	***8.60***
	60	4428	5160	4887	4831	4719	4719	4663	4663	***4663***	***5.31***
	Ins.	Best from lit.	BDP Variant 2 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
	51	4500	5403	5252	5076	5044	4971	4945	4911	4911	9.13
	52	4276	5165	4955	4882	4811	4811	4811	4756	4756	11.23
	53	4289	5308	5050	4936	4838	4819	4751	4737	4737	10.45
	54	4377	5311	5002	4897	4814	4715	4671	4671	4671	6.72
	55	4268	5386	5054	4937	4862	4728	4728	4669	4669	9.40
	56	4280	5549	5204	5024	4963	4898	4805	4805	4805	12.27
	57	4308	5465	5119	4981	4932	4849	4792	4772	4772	10.77
	58	4326	5487	5163	5039	4912	4893	4825	4825	4825	11.53
	59	4316	5384	5115	5010	4835	4812	4786	4754	4754	10.15
	60	4428	5456	5186	5089	4939	4885	4843	4782	4782	7.99

TABLE VII. SOLUTIONS FOR TAILLARD'S SET 7

Set 7	Ins.	Best from lit.	BDP Variant 1 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
n =100	61	6151	6764	6417	6329	6270	6230	6225	6225	***6225***	***1.20***
m=5	62	6022	6537	6236	6113	6113	6108	6108	6034	***6034***	***0.20***
	63	5927	6368	6207	5975	5975	5975	5942	5942	***5942***	***0.25***
	64	5772	6190	5926	5808	5805	5782	5782	5782	***5782***	***0.17***
	65	5960	6453	6089	6070	6050	6050	6050	6016	***6016***	***0.94***
	66	5852	6471	6034	5945	5945	5876	5876	5876	***5876***	***0.41***
	67	6004	6471	6220	6111	6081	6056	6056	6050	***6050***	***0.77***
	68	5915	6397	6056	6002	5916	5916	5882	5882	***5882***	***-0.56***
	69	6123	6647	6255	6255	6255	6201	6201	6172	***6172***	***0.80***
	70	6159	6741	6274	6274	6244	6180	6154	6154	***6154***	***-0.08***
	Ins.	Best from lit.	BDP Variant 2 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
	61	6151	7233	6872	6699	6677	6618	6577	6505	6505	5.76
	62	6022	7129	6651	6452	6410	6346	6282	6282	6282	4.32
	63	5927	6995	6623	6365	6308	6285	6212	6170	6170	4.10
	64	5772	6681	6265	6212	6115	6056	5992	5985	5985	3.69
	65	5960	6960	6428	6266	6266	6247	6217	6217	6217	4.31
	66	5852	7108	6538	6370	6163	6094	6024	5969	5969	2.00
	67	6004	6980	6516	6311	6252	6222	6195	6195	6195	3.18
	68	5915	7127	6495	6410	6275	6200	6182	6151	6151	3.99
	69	6123	7178	6679	6549	6479	6447	6359	6344	6344	3.61
	70	6159	7179	6788	6603	6572	6520	6491	6422	6422	4.27

TABLE VIII. SOLUTIONS FOR TAILLARD'S SET 8

Set 8	Ins.	Best from lit.	BDP Variant 1 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
n =100	71	7042	7790	7404	7374	7283	7246	7231	7231	***7231***	***2.68***
m=10	72	6791	7547	7097	6957	6895	6866	6814	6814	***6814***	***0.34***
	73	6936	7728	7293	7165	7157	7065	7050	7050	***7050***	***1.64***
	74	7187	7925	7701	7553	7521	7482	7466	7405	***7405***	***3.03***
	75	6810	7424	7110	7008	6962	6932	6932	6932	***6932***	***1.79***
	76	6666	7427	7046	6971	6971	6934	6878	6855	***6855***	***2.84***
	77	6801	7681	7322	7117	7117	7071	6983	6983	***6983***	***2.68***
	78	6874	7415	7257	6998	6998	6998	6998	6972	***6972***	***1.43***
	79	7055	7955	7453	7344	7281	7216	7216	7216	***7216***	***2.28***
	80	6965	7705	7344	7225	7129	7129	7125	7123	***7123***	***2.27***
	Ins.	Best from lit.	BDP Variant 2 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
	71	7042	8128	7685	7574	7480	7390	7325	7306	7306	3.75
	72	6791	8032	7566	7373	7262	7183	7106	7069	7069	4.09
	73	6936	8160	7800	7573	7526	7459	7279	7206	7206	3.89
	74	7187	8483	7858	7711	7663	7645	7478	7478	7478	4.05
	75	6810	8101	7718	7421	7347	7310	7287	7208	7208	5.84
	76	6666	8078	7450	7241	7151	7058	6996	6926	6926	3.90
	77	6801	8201	7761	7551	7468	7323	7284	7216	7216	6.10
	78	6874	8066	7568	7380	7302	7248	7163	7127	7127	3.68
	79	7055	8348	7776	7694	7646	7597	7428	7382	7382	4.64
	80	6965	8124	7698	7558	7498	7421	7279	7266	7266	4.32

TABLE IX. SOLUTIONS FOR TAILLARD'S SET 9

Set 9	Ins.	Best from lit.	BDP Variant 1 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
n =100	81	7844	9309	8982	8673	8560	8551	8479	8395	***8395***	***7.02***
m=20	82	7894	9234	8540	8380	8309	8248	8232	8232	***8232***	***4.28***
	83	7794	9016	8664	8434	8434	8382	8334	8334	***8334***	***6.93***
	84	7899	8891	8609	8604	8416	8244	8244	8240	***8240***	***4.32***
	85	7901	9024	8378	8378	8225	8225	8203	8203	***8203***	***3.82***
	86	7888	9241	8765	8553	8340	8340	8318	8318	***8318***	***5.45***
	87	7930	8936	8620	8457	8457	8364	8364	8364	***8364***	***5.47***
	88	8022	9386	8794	8681	8577	8534	8487	8449	***8449***	***5.32***
	89	7969	8995	8626	8525	8357	8357	8205	8205	***8205***	***2.96***
	90	7993	9275	8886	8771	8655	8655	8564	8432	***8432***	***5.49***
	Ins.	Best from lit.	BDP Variant 2 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
	81	7844	9493	9148	8998	8856	8670	8670	8571	8571	9.27
	82	7894	9579	9128	8868	8814	8717	8676	8676	8676	9.91
	83	7794	9395	8949	8818	8740	8716	8601	8496	8496	9.01
	84	7899	9385	8972	8854	8845	8704	8668	8628	8628	9.23
	85	7901	9350	8958	8831	8739	8622	8589	8534	8534	8.01
	86	7888	9361	8954	8799	8759	8607	8580	8580	8580	8.77
	87	7930	9670	9145	8955	8856	8800	8728	8693	8693	9.62
	88	8022	9671	9230	9102	9012	8894	8771	8755	8755	9.14
	89	7969	9457	9078	8902	8809	8732	8675	8638	8638	8.40
	90	7993	9743	9348	9226	9068	8963	8891	8749	8749	9.46

TABLE X. SOLUTIONS FOR TAILLARD'S SET 10

Set 10	Ins.	Best from lit.	BDP Variant 1 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
n =200	91	13406	14678	14089	13674	13674	13674	13557	13537	***13537***	***0.98***
m=10	92	13313	14472	13832	13592	13537	13311	13287	13287	***13287***	***-0.20***
	93	13416	14463	13944	13727	13691	13615	13503	13475	***13475***	***0.44***
	94	13344	14641	13981	13662	13644	13618	13550	13435	***13435***	***0.68***
	95	13360	14344	13962	13694	13566	13471	13426	13319	***13319***	***-0.31***
	96	13192	14115	13754	13484	13339	13304	13304	13273	***13273***	***0.61***
	97	13598	15270	14161	13866	13805	13805	13728	13728	***13728***	***0.96***
	98	13504	14831	13909	13782	13678	13629	13626	13626	***13626***	***0.90***
	99	13310	14590	13735	13511	13468	13386	13386	13358	***13358***	***0.36***
	100	13439	14861	13914	13617	13600	13535	13523	13480	***13480***	***0.31***
	Ins.	Best from lit.	BDP Variant 2 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
	91	13406	15561	14885	14484	14333	14156	14001	13985	13985	4.32
	92	13313	15404	14623	14306	14171	14135	13977	13939	13939	4.70
	93	13416	15586	14769	14445	14289	14141	14005	14005	14005	4.39
	94	13344	15629	14773	14334	14253	14067	13922	13922	13922	4.33
	95	13360	15903	14959	14555	14411	14199	13968	13968	13968	4.55
	96	13192	15288	14604	13990	13978	13782	13603	13585	13585	2.98
	97	13598	15811	15041	14711	14614	14448	14322	14312	14312	5.25
	98	13504	15674	14984	14553	14433	14226	14138	14138	14138	4.69
	99	13310	15533	14767	14252	14106	14021	13987	13839	13839	3.97
	100	13439	15597	14861	14432	14363	14220	14172	14145	14145	5.25

TABLE XI. SOLUTIONS FOR TAILLARD'S SET 11

Set 11	Ins.	Best from lit.	BDP Variant 1 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
n =200	101	14912	16617	16006	15558	15475	15331	15263	15263	***15263***	***2.35***
m=20	102	15002	16919	15860	15721	15718	15580	15501	15411	***15411***	***2.73***
	103	15186	16484	15838	15568	15409	15324	15318	15318	***15318***	***0.87***
	104	15082	16533	15762	15515	15506	15506	15430	15296	***15296***	***1.42***
	105	14970	17064	15823	15623	15569	15351	15351	15307	***15307***	***2.25***
	106	15101	16713	15968	15781	15781	15482	15482	15453	***15453***	***2.33***
	107	15099	16746	16075	15846	15840	15691	15567	15567	***15567***	***3.10***
	108	15141	16430	15897	15594	15566	15534	15405	15388	***15388***	***1.63***
	109	15034	16452	15896	15540	15540	15361	15361	15351	***15351***	***2.11***
	110	15122	16385	15796	15522	15464	15400	15370	15370	***15370***	***1.64***
	Ins.	Best from lit.	BDP Variant 2 H=1	H=10	H=50	H=100	H=250	H=500	H=750	Best solution found	Best RPD
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
	101	14912	17474	16710	16374	16191	15965	15818	15743	15743	5.57
	102	15002	17935	17081	16657	16537	16456	16272	16037	16037	6.90
	103	15186	17517	17037	16842	16606	16402	16170	16170	16170	6.48
	104	15082	17888	17104	16777	16645	16468	16238	16238	16238	7.66
	105	14970	17455	16701	16565	16481	16224	16053	16053	16053	7.23
	106	15101	17897	17469	16937	16816	16524	16357	16357	16357	8.32
	107	15099	17537	16868	16595	16480	16295	16166	16166	16166	7.07
	108	15141	17575	16965	16592	16481	16408	16118	16069	16069	6.13
	109	15034	17500	17074	16577	16395	16395	16119	16065	16065	6.86
	110	15122	17602	17077	16681	16570	16270	16270	16128	16128	6.65

In Tables I to XI we can observe that values of RPD are between -0,56% and 9,17% in all the instances. RPD negative values indicate an improvement to the best solution reported in the literature; these improvements occur in 4 instances: instances 68, 70 and 92 (with a window width H=500), and instance 95 (with a window width H=750).

We can observe that the improvement for C_{max} that occurs (on average) between consecutive window widths is clearly decreasing. In addition, C_{max} values obtained tend asymptotically to the best known value of C_{max} when we increase the window width.

The average RPD for the 110 instances is 2,66%. The average RPD for each set (1 to 11) and variant (1 and 2) are reported in Table XII. Table XII also shows average CPU time (in seconds) for both variants of the BDP and windows widths H=500 and H=750 (for the 11 sets).

TABLE XII. AVERAGE RPD AND CPU TIMES FOR THE 11 SETS

Sets	1	2	3	4	5	6	7	8	9	10	11	All sets
% average RPD 1	1.22	2.22	4.53	2.10	3.36	6.05	0.41	2.10	5.11	0.47	2.04	2.69
% average RPD 2	2.11	4.65	5.95	3.58	5.71	9.96	3.92	4.43	9.08	4.44	6.89	5.52
% average RPD (both)	1.02	2.20	4.40	2.10	3.32	6.05	0.41	2.10	5.11	0.47	2.04	2.66
average CPU 1/500	3.2	3.6	4.6	36.3	44.2	62.7	191.7	249.0	378.3	1625.2	2630.1	475.4
average CPU 2/500	2.4	3.2	4.4	39.8	46.8	63.9	251.8	299.3	425.2	2156.9	2989.2	571.2
average CPU 1/750	6.9	7.6	9.2	79.1	92.8	128.1	420.3	509.1	719.1	3190.0	4793.4	905.1
average CPU 2/750	4.9	6.3	8.4	83.5	98.6	129.7	545.0	619.7	823.6	4415.5	5554.8	1117.3

Comparing the results obtained by both variants, we can see that BDP Variant 1 outperforms BDP Variant 2, in CPU times and solutions obtained.

VII. CONCLUSIONS

In this paper a BDP (Bounded Dynamic Programming) has been proposed for solving the permutation flow shop problem with blocking. This type of procedure has been used to solve sequencing in mixed assembly lines and assembly line balancing problems but, to the best of our knowledge, it has not been used to solve the problem here considered.

The BDP combines features of dynamic programming with features of branch and bound algorithms. The main elements which define the efficiency of the BDP procedure are the graph associated to the problem, the initial solution, the bounding scheme used to prune the graph and the window width used. The window width limits the maximum number of partial solutions retained in each level, therefore it is also necessary to define the rules to decide which vertices are pruned. In our implementation two different variants has been used. The best behavior has been obtained when the priority rule keeps those vertices with a best bound of C_{max} and in case of ties those with the best partial C_{max}. Even though we have set the initial solution (Z_0) to infinite and we have used a simple bounding scheme, we have improved the best known solutions for the Taillard's instances number 68, 70 and 92 with a window width of 500, and instance number 95 with a window width of 750, in a competitive time.

Future research will focus on using an improved bounding scheme more adapted to the characteristics of the problem which, combined with a better initial solution as the MME2 proposed in [20], could help to improve the efficiency of the procedure.

ACKNOWLEDGMENT

The authors greatly appreciate the collaboration of Nissan Spanish Industrial Operations (NSIO) as well as the Nissan Chair UPC for partially funding this research. This work was also partially funded by projects PROTHIUS-II, DPI2007-63026 and PROTHIUS-III, DPI2010-16759 including EDRF fundings from the Spanish government.

REFERENCES

[1] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey", *Annals of Discrete Mathematics*, vol. 5, pp. 287-326, 1979.

[2] N.G. Hall and C. Sriskandarajah, "A survey of machine scheduling problems with blocking and no wait in process", *Operations Research*, vol. 44(3), pp. 510-525, 1996.

[3] S.S. Reddi and B. Ramamoorthy, "On the flow-shop sequencing problem with no wait in process", *Operations Research Quarterly*, vol. 23(3), pp. 323-331, 1972.

[4] P.C. Gilmore and R.E. Gomory, "Sequencing a one state-variable machine: A solvable case of the traveling salesman problem", *Operations Research*, vol. 12, pp. 655-679, 1964.

[5] P.C. Gilmore, E.L. Lawler and D.B. Shmoys, "Well-solved special cases", in *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, E.L. Lawler, K.L. Lenstra, A.H.G Rinooy Kan and D.B. Shmoys, Eds. Wiley, 1991, pp. 87-143.

[6] C.H. Papadimitriou and P.C. Kanellakis, "Flowshop scheduling with limited temporary storage", *Journal of the ACM*, vol. 27(3), pp. 533-549, 1980.

[7] D.P. Ronconi, "A Branch-and-Bound Algorithm to Minimize the Makespan in a Flowshop with Blocking", *Annals of Operations Research,* vol. 138(1), pp. 53-65, 2005.

[8] R. Companys and M. Mateo, "Different behaviour of a double branch-and-bound algorithm on Fm|prmu|Cmax and Fm|block|Cmax problems", *Computers & Operations Research*, vol. 34(4), pp. 938-953, 2007.

[9] E. Taillard, "Benchmarks for basic scheduling problems", *European Journal of Operational Research*, vol. 64(2), pp. 278-285, 1993.

[10] S.T. McCormick, M.L. Pinedo, S. Shenker and B. Wolf, "Sequencing in an Assembly Line with Blocking to Minimize Cycle Time", *Operations Research,* vol. 37, pp. 925-936, 1989.

[11] R. Leisten, "Flowshop sequencing problems with limited buffer storage", *International Journal of Production Research*, vol. 28(11), pp. 2085, 1990.

[12] M. Nawaz, Jr E.E. Enscore and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem", *Omega*, vol. 11(1), pp. 91-95, 1983.

[13] D.P. Ronconi, "A note on constructive heuristics for the flowshop problem with blocking", *International Journal of Production Economics*, vol. 87(1), pp. 39-48, 2004.

[14] V. Caraffa, S. Ianes, P. Bagchi T and C. Sriskandarajah, "Minimizing makespan in a blocking flowshop using genetic algorithms", *International Journal of Production Economics*, vol. 70(2), pp. 101-115, 2001.

[15] J. Grabowski and J. Pempera, "The permutation flow shop problem with blocking. A tabu search approach", *Omega*, vol. 35(3), pp. 302-311, 2007.

[16] L. Wang, L. Zhang and D. Zheng, "An effective hybrid genetic algorithm for flow shop scheduling with limited buffers", *Computers & Operations Research*, vol. 33(10), pp. 2960-2971, 2006.

[17] B. Liu, L. Wang and Y. Jin, "An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers", *Computers & Operations Research*, vol. 35(9), pp. 2791-2806, 2008.

[18] B. Qian, L. Wang, D.X. Huang and X. Wang, "An effective hybrid DE-based algorithm for flow shop scheduling with limited buffers", *International Journal of Production Research*, vol. 47(1), pp. 1-24, 2009.

[19] L. Wang, Q. Pan, P.N. Suganthan, W. Wang and Y. Wang, "A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems", *Computers & Operations Research*, vol. 37(3), pp. 509-520, 2010.

[20] I. Ribas, R. Companys and X. Tort-Martorell, "An iterated greedy algorithm for the flowshop scheduling problem with blocking", *Omega*, vol. 39, pp. 293-301, 2011.

[21] J. Bautista, "*Procedimientos heurísticos y exactos parala secuenciación en sistemas productivos de unidades homogéneas (contexto J.I.T.*)", Doctoral Thesis, DOE, ETSEIB-UPC, 1993.

[22] J. Bautista, R. Companys and A. Corominas, "Heuristics and exact algorithms for solving the Monden problem", *European Journal of Operational Research,* vol. 88, pp. 101-113, 1996.

[23] Th.L. Morin and R.E. Marsten, "Branch-and-bound strategies for Dynamic Programming", *Operations Research*, vol. 24(4), pp. 611-627, 1976.

[24] R.E. Marsten and Th.L. Morin, "A hybrid approach to discrete Mathematical Programming", *Mathematical Programming*, vol. 14, pp. 21-40, 1978.

[25] R.L. Carraway and R.L. Schmidt, "An improved discrete Dynamic Programming algorithm for allocating resources among interdependent projects", *Management Science*, vol. 37(9), pp. 1195-1200, 1991.

[26] M. L. Pinedo, *"Scheduling: Theory, Algorithms, and Systems"*, Springer (Third edition), 2008.

[27] J. Bautista and A. Cano, "Solving mixed model sequencing problem in assembly lines with serial workstations with work overload minimisation and interruption rules*", European Journal of Operational Research*, vol. 210(3), pp. 495-513, 2011.

[28] Z. A. Lomnicki, "A "branch-and-bound" algorithm for the exact solution of the three-machine scheduling problem", *Operations Research Quarterly*, vol. 16(1), pp. 89-100, 1965.

[29] J. Bautista and J. Pereira, "A dynamic programming based heuristic for the assembly line balancing problem", *European Journal of Operational Research*, vol. 194(3), pp. 787-794, 2009.