A 3/2-Approximation Algorithm for the Jump Number of Interval Orders*

Stefan Felsner

TU Berlin

The jump number of a partial order P is the minimum number of incomparable adjacent pairs in some linear extension of P. The jump number problem is known to be NP–hard in general. However some particular classes of posets admit easy calculation of the jump number.

The complexity status for interval orders still remains unknown. Here we present a heuristic that, given an interval order P, generates a linear extension Λ, whose jump number is less than 3/2 times the jump number of P.

1 Introduction

A *partial order* will be denoted by $P = (X, <_P)$, where X is a finite base set and $<_P$ is the order relation. In the special case of a linear order we will write $L = x_1 x_2 \dots x_n$, where the left to right ordering defines the relation $<_L$, i.e. $x_i <_L x_j$ iff $i < j$. A *linear extension* of a partial order $P = (X, <_P)$ is a linear order $L = x_1 x_2 \dots x_n$ respecting the order relations of P, i.e. $x_i <_P x_j$ implies $i < j$ for all $x_i, x_j \in X$. A more dynamic definition would be that L is a linear extension of P iff $x_i \in Min(Q_i)$ for $1 \leq i \leq n$, where $Min(Q_i)$ denotes the set of minimal elements of $Q_i = P \setminus \{x_1, \dots, x_{i-1}\}$.

Let L be a linear extension of P. Two consecutive elements $x_i x_{i+1}$ of L are separated by a *jump* iff x_i is incomparable to x_{i+1} in P; sometimes we

*This work was supported by the Deutsche Forschungsgemeinschaft (DFG).

call this the 'jump after x_i'. If $x_i < x_{i+1}$ the pair $x_i x_{i+1}$ is called a *bump*. The total number of jumps of L is denoted by $s_P(L)$ or $s(L)$. The *jump number* $s(P)$ *of* P is the minimum number of jumps in some linear extension, i.e.

$$s(P) = \min\{s_P(L) : L \text{ is a linear extension of } P\}$$

A linear extension L of P with $s_P(L) = s(P)$ will be called *optimal*.

The jump number has been introduced by Chein and Martin [1972]. The minimization problem, 'determine $s(P)$ and find an optimal linear extension', has been shown to be NP-hard even for bipartite orders (Pulleyblank [1982]). Nevertheless efficient methods for jump-minimization have been found for large classes of partially ordered sets such as N-free orders (Rival [1983]), cycle-free orders (Duffus, Rival, Winkler [1982]) or orders with bounded decomposition width (Steiner [1985]). For some further wellknown classes, i.e. the 2–dimensional and the interval orders the complexity of the problem remains open. In both cases, however, computation of the jump number is easy on the subclass of bipartite orders (Steiner, Stewart [1987]).

Probably the most interesting result on jump numbers is the one concerning the N-free orders. For these any linear extension generated by the greedy algorithm is optimal.

GREEDY algorithm

$N = \emptyset, L = \emptyset,$
while $P \neq \emptyset$ **do**
  if $N \neq \emptyset$ **do**
    choose$(x \in N)$: bump
  else
    choose$(x \in Min(P))$**,** : jump
  $L := L + x$**,**
  $P := P \setminus \{x\}$**,**
  $N := Suc(x) \cap Min(P)$.

A great amount of work has been invested to extend this result (Sysło [1984]; Faigle, Gierz, Schrader [1985]).

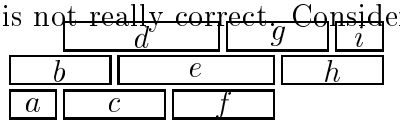
An *interval order* is an ordered set $P = (X, <_P)$, whose elements are in a

one to one correspondence with intervals on the real axis ($x \leftrightarrow I_x$) such that $x <_P y$ iff $sup\, I_x \leq inf I_y$.

For $x \in P$ let $Suc(x) = \{y \in P : x <_P y\}$ be the *successorset of* x. If P is an interval order, then it is clear from the interval representation that for any two elements $x, y \in P$ we have $Suc(x) \subseteq Suc(y)$ or $Suc(x) \supseteq Suc(y)$. This and putting $x \lhd y$ iff $Suc(y) \subset Suc(x)$ defines a new order $P_\lhd = (X, \lhd)$ on the same base set. The order $P_\lhd$ extends the given order P, i.e. $x <_P y$ implies $x \lhd y$, and $P_\lhd$ is already quite close to a linear order, in fact, it is a weak order. Another trivial but important property of interval orders is that the induced order on the subset of an interval order is an interval order too.

T. Ghazal et al. [1988] characterized greedy interval orders (interval orders with the property that any optimal linear extension is greedy, i.e. can be generated by the greedy algorithm).

Faigle and Schrader [1985] gave the following greedy-heuristic for jump-minimization in interval orders: Construct a greedy linear extension L of $P_\lhd$ that ommits P-jumps whenever possible. They asserted $s_P(L) \leq 2s(P)$. The factor of two, however, is not really correct. Consider this interval order:



A linear extension of P, that is a linear extension of $P_\lhd$ too, has at least 7 jumps (e.g. $a|b|c|d|e|f|gi|h$). The optimal linear extension of P only has 3 jumps, it is $ad|be|cfh|gi$. However, it is easy to rewrite our Theorem 1, to show that the real factor to their method is less then 3.

The heuristic presented here generates (given an interval order P) a linear extension Λ, whose jump number is less than 3/2 times the jump number of P, i.e. $s_P(\Lambda) \leq \frac{3}{2} s(P)$.

2 Starting elements

Let $Start(P)$ denote the set of *starting elements*, i.e. those elements $a \in Min(P)$ that appear as minimal (first) element in some optimal linear extension of P.

First we state the following two lemmas, which are valid for the starting elements of arbitrary partial orders. If $P = (X, <)$, we will use the abbreviation P_a to denote the induced order on the set $X \setminus \{a\}$.

Lemma 1 *If* $a \in \mathrm{Min}(P)$ *and* $s(P) = s(P_a) + 1$, *then* $a \in \mathrm{Start}(P)$.

Proof. Take any optimal extension L' of P_a. Since a is a minimal element of P the concatenation $L = a + L'$ is a linear extension of P. Counting jumps proves the optimality of L. □

Lemma 2 *If* $a \in \mathrm{Start}(P)$ *and* $s(P) = s(P_a)$, *then* $\mathrm{Suc}(a) \cap \mathrm{Start}(P_a) \neq \emptyset$.

Proof. Let $L = a + b + L'$ be any optimal linear extension of P starting with a. If $b \notin \mathit{Suc}(a)$, or equivalently if (a, b) is jump, then $s(P_a) \leq s_{P_a}(b + L') = s_P(L) - 1 = s(P) - 1$ would contradict $s(P) = s(P_a)$, so $b \in \mathit{Suc}(a)$. The equalities $s(P_a) = s(P) = s(L) = s(b + L')$ then show the optimality of $b + L'$ for P_a, so $b \in \mathit{Start}(P_a)$ too. □

We now state an algorithm, which is very similar to the greedy algorithm. In contradistinction to the latter, the starty algorithm never fails to generate optimal linerar extensions.

STARTY algorithm

$N = \emptyset, L = \emptyset,$
while $P \neq \emptyset$ **do**
  if $N \neq \emptyset$ **do**
    choose$(x \in N)$: bump
  else
    choose$(x \in \mathit{Start}(P))$**,** : jump
  $L := L + x$**,**
  $P := P \setminus \{x\}$**,**
  $N := \mathit{Suc}(x) \cap \mathit{Start}(P)$.

Lemma 3 *If the linear extension* L *of* P *is generated by the starty algorithm, then* $s_P(L) = s(P)$, *i.e.* L *is optimal.*

Proof. For the proof we use induction on $|\,P\,|$. First note that if $L = a + L'$ is generated by the starty algorithm with input P, then with input P_a there is a run of the starty algorithm producing L'. Comparing the jump number of L and L' we find two possibilities:

- If $s(L) = s(L')$, then the trivial inequality $s(P) \geq s(P_a)$ and the optimality of L' for P_a implies the optimality of L.
- Otherwise, i.e. if $s(L) = s(L') + 1$, then there is a jump after a, hence $Suc(a) \cap Start(P_a) = \emptyset$ and, by Lemma 2, $s(P) = s(P_a) + 1$. Again the optimality of L is deduced from the optimality of L'. □

Corollary. The *NP*-hardness of the jump number problem implies that, in general, the identification of starting elements is hard, too.

In the case of interval orders we have slightly more information on starting elements than given in the previous lemmas.

Lemma 4 [Faigle,Schrader 85] *Let P be an interval order, $a \in \mathrm{Min}(P)$, then $a \in \mathrm{Start}(P)$ iff a is a $\lhd$-minimal element of P or $s(P) = s(P_a) + 1$.*

Proof. In an interval order there can be at most one starting element with $s(P) = s(P_a)$, since only to the (in this case unique) $\lhd$-minimal element there may exist $b \in Suc(a) \cap Min(P_a)$ as needed by Lemma 2 ($Start \subseteq Min$). Any other $a \in Start(P)$ thus fulfills the equation $s(P) = s(P_a) + 1$.

So it only remains to prove that a $\lhd$-minimal element a is a starting element. First note that if $L = x_1 x_2 \ldots x_n$ is optimal and $a = x_i$, then the set $\{x_1, \ldots, x_i\}$ is an antichain in P, since $Suc(x_{j-1}) \subseteq Suc(a)$. Let x_k be the first element of L with $x_k >_P x_1$, then $k > i$ and $g : L \mapsto g(L) = x_2 \ldots x_{k-1} x_1 x_k \ldots x_n$ is an operation, which does not increase the number of jumps. The linear extension $g^{i-1}(L)$ is optimal and starts with a. □

3 Some algorithms

In this section we will be concerned with interval orders only. The first algorithm transforms a given interval order into an auxiliary list $\mathcal{L}(P)$, which will than be the input to three further algorithms. This procedure allows us to compare the jump number of the linear extensions generated by the three algorithms and enables us to prove the 3/2 performance bound of our heuristic.

Let $min_{\lhd} M$ be the (whith respect to P) $\lhd$-minimal element of M that has the maximal set of predecessors. As in the greedy algorithm the set of admissible successors of an element x_i will be denoted by N. In the list $\mathcal{L}(P)$ the elements of P appear in $\lhd$-order, however, if after x_i, none of the

admissible successors of x_i is $\lhd$-minimal, a 'box' is generated and they are stored as candidates for further insertion. The further algorithms then use different insertion-strategies.

Auxiliary Algorithm

$N := \emptyset,\ L := \emptyset$,
while $P \neq \emptyset$ **do**
    $x := min_{\lhd} P$,
    if $N \neq \emptyset$ **do**
        if $x \not\lhd min_{\lhd} N$ **do**
            $x := min_{\lhd} N$,
            $L := L + x$: bump
        else
            $L := L + \square_N + x$, : jump
    else
        $L := L + x$, : jump
    $P := P \setminus \{x\}$,
    $N := Suc(x) \cap Min(P)$.

The algorithm gives the following list:

$$\mathcal{L}(P) = L^1 + \square_{N_1} + L^2 + \square_{N_2} + \ldots + L^\beta + \square_{N_\beta} + L^{\beta+1}$$

Remark. 1) Note that in each loop $x \in Min(P)$. So if we ommit the boxes, we may see $\mathcal{L}(P)$ as a linear extension of P. This justifies the comments given in the algorithm. However, to get things right, we will transforme $\mathcal{L}$ to the linear extension Λ_1 using a trivial algorithm, c.f. Lin_Ext_1.
2) An element y may at most once appear in a set $N = Suc(x) \cap Min(P)$, hence the N_i are pairwise disjoint.
3) If $y \in N_i$, for some i, then y is a minimal element of the remaining poset, hence in $\mathcal{L}$ there must be a jump just before y.

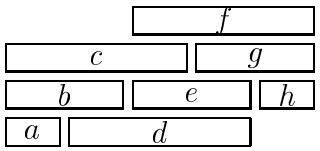
Lin_Ext_1

$\Lambda_1 := L^1 + L^2 + \ldots L^{\beta+1}$.

Note that in the auxiliary algorithm the new element x is always choosen as a $\lhd$–minimal element, hence Λ_1 is even a linear extension of $P_\lhd$.

There are two types of jumps in the linear extension Λ_1. Jumps of the first type correspond to the boxes in $\mathcal{L}(P)$, let their number be β. The other jumps arise if $N = \emptyset$, let their number be $\alpha = \sum s(L^i)$. Thus $s(\Lambda_1) = \alpha + \beta$.

Example.



Given this interval order P we get $\mathcal{L}(P) = a\Box_d|b\Box_{ef}|c\Box_g|d|eh|f|g$ and $\Lambda_1(P) = a|b|c|d|eh|f|g$. As indicated, there are 6 jumps, 3 of them are counted by β.

For the second algorithm we will need a further notion. Let an element $b \in P$ be called *jumpreducing in* L (L is a linear extension of P) if and only if $s_{P_b}(L_b) < s_P(L)$. Here L_b is derived from L just as we get P_a from P, i.e. $L_b := x_1 \dots x_{i-1}x_{i+1} \dots x_n$ if $L = x_1 \dots x_{i-1} b\, x_{i+1} \dots x_n$. Let $Red(L)$ denote the set of all jumpreducing elements of L

Lin_Ext_2

$\mathcal{L} := \mathcal{L}(P)$

while *there is a* $\underline{last}$ $\Box$ *in* $\mathcal{L}$ **do**

  $\mathcal{L} = \mathcal{L}' + \Box_N + L$: decompose $\mathcal{L}$

  if $N \cap Red(L) \neq \emptyset$ **do**

    choose$(n \in N \cap Red(L))$,

    $\mathcal{L} := \mathcal{L}' + n + L_n$

  else

    $\mathcal{L} := \mathcal{L}' + L$

$\Lambda_2 := \mathcal{L}$.

Example. If the input of Lin_Ext_2 is $\mathcal{L}(P) = a\Box_d|b\Box_{ef}|c\Box_g|d|eh|f|g$, the list constructed in the preceding example, then there are two possible outcomes, namely $\Lambda_2 = a|be|cg|dh|f$ with 4 jumps, or $\Lambda_2 = ad|bf|cg|eh$ with 3 jumps. Which of the two alternatives occures, depends on whether we choose e or f as the jumpreducing element to be pulled into the second box. As the width of P equals 4 the linear extension with 3 jumps must be

optimal, in fact it is the unique optimal linear extension of P.

Lemma 5 *If the linear extension L of P is generated by Lin_Ext_2, i.e. $L = \Lambda_2(P)$, then $x \in \mathrm{Red}(L)$ implies $s(L) = s(L_x) + 1$.*

Proof. By definition of jumpreducing, $s(L) > s(L_x)$, and trivially $s(L_x) \geq s(L) - 2$. We thus only have to exclude the case $s(L_x) = s(L) - 2$.

Now assume $s(L_x) = s(L) - 2$, we find that $L = \ldots a|x|b \ldots$, with $b \in Suc(a)$, $b \notin Suc(x)$. Hence $a \lhd x \lhd b$ and in $\mathcal{L}(P)$ the elements a, x, b did appear in the same order, moreover $\mathcal{L}(P) = \ldots a \Box_{N_a} x \Box_{N_x} b \ldots$, with $b \in N_a$.

If Lin_Ext_2 does find a jumpreducing element in N_x, then in $\Lambda_2(P)$ there is a bump after x, if not, then $b \in N_a$ is jumpreducing and in $\Lambda_2(P)$ there is a bump after a. Together this contradicts the assumption $L = \Lambda_2(P)$. □

Lemma 6 *Let ρ count the number of times that the if condition is true in a run of algorithm Lin_Ext_2. Then $s(\Lambda_2) = s(\Lambda_1) - \rho = \alpha + \beta - \rho$.*

Proof. The algorithm searches for the last box in $\mathcal{L}$ and decomposes $\mathcal{L} = \mathcal{L}' + \Box_N + L$. Since L then is a linear extension for some subset $Q = \{L\}$ of P, in fact $L = \Lambda_2(Q)$, we can ask whether $n \in N$ is jumpreducing in L with respect to Q. If in L we find a jumpreducing $n \in N$, then we pull n foreward and thus reduce the number of jumps by one. The number ρ just counts the jumps reduced in this way.

A bit more formally – the following formula has to be verified:

$$s(\Lambda_1(\mathcal{L}' + n + L_n)) = s(\Lambda_1(\mathcal{L}' + \Box_N + L)) - 1.$$

The jumps in $\mathcal{L}'$agree. Since n is a successor of the last element x of $\mathcal{L}'$, we may charge the jump after x in $\Lambda_1(\mathcal{L}' + \Box_N + L)$ with the jump after n in $\Lambda_1(\mathcal{L}' + n + L_n)$. Finaly n is jumpreducing in $L = \Lambda_2(Q)$, so we may apply Lemma 5 to get $s(L_n) = s(L) - 1$. □

Lemma 7 *Let the linear extension Λ_1 and Λ_2 of the interval order P be generated by Lin_Ext_1 and Lin_Ext_2. If α and β are the 'jumpcounters' of Λ_1 introduced above. Then $s(\Lambda_2) \leq \alpha + \frac{\beta}{2}$.*

Proof. From Lemma 6 we know that $s(\Lambda_2) = \alpha + \beta - \rho$, so we only have to prove $\rho \geq \beta/2$. Let $\mathcal{L} = \mathcal{L}' + \Box_N + L$ be any decomposition occurring in the run of Lin_Ext_2. Take $n \in N$ and suppose that n is not jumpreducing. Now let $Q = \{L\}$, then $n \in Min(Q)$, hence there must be a jump between the

immediate predecessor x of n in L and n. Let y be the immediate successor of n in L. If (n, y) is a jump, or if n is the last element in L, then n is seen to be jumpreducing. Hence (n, y) must be a bump, i.e. $y \in Suc(n)$. For n not to be jumpreducing we need in addition that $y \notin Suc(x)$. Altogether this implies $Suc(x) \subset Suc(n)$ or equivalently $n \lhd x$. Since, as we saw before, Λ_1 is a linear extension of $P_\lhd$, the element x comes after n in Λ_1 and as well in $\mathcal{L}(P)$. We conclude that the element x has been pulled foreward into a box in some earlier step.

Let I be the set of indices of boxes that had to remain empty and let J be the set of indices of filled boxes, i.e. $I \cup J = \{1 \ldots \beta\}$. To a box $\square_N$ that could not be filled and each $n \in N$ we did find the immediate predecessor x of n in a box filled with x. Hence we have a injective function

$$\Theta : \bigcup_{i \in I} N_i \longrightarrow J.$$

Since $\mid N_i \mid \geq 1$ for all i we get $\mid I \mid \leq \mid J \mid = \rho$ (c.f. the definition of ρ in Lemma 6). This last inequality, together with $\mid I \mid + \mid J \mid = \beta$ gives $\rho \geq \beta/2$.

□

The last algorithm of this section converts $\mathcal{L}(P)$ into an optimal linear extension of P. This algorithm is based upon the recognition of starting elements, so it is impracticable, however, it enables us to relate $s(P)$ to α and β and will thus help us to show $s(\Lambda_2) \leq (3/2)s(P)$.

Lin_Ext_3

$\mathcal{L} := \mathcal{L}(P)$

while *there is a first* $\square$ *in* $\mathcal{L}$ **do**

  $\mathcal{L} = L + \square_N + \mathcal{L}'$: decompose $\mathcal{L}$

  $Q := \{\mathcal{L}'\}$,

  if $N \cap Start(Q) \neq \emptyset$ **do**

    choose$(n \in N \cap Start(Q))$,

    $\mathcal{L} := L + n + \mathcal{L}(Q \setminus \{n\})$

  else

    $\mathcal{L} := L + \mathcal{L}'$

$\Lambda_3 := \mathcal{L}$.

Lemma 8 *The linear extension* $\Lambda_3 = \Lambda_3(\mathcal{L}(P))$ *generated by the preceeding*

algorithm is an optimal linear extension of P.

Proof. With reference to Lemma 3 it suffices to show that there is a run of the starty algorithm with input P that generates $\Lambda_3(\mathcal{L}(P))$. Let $\Lambda_3 = x_1x_2\ldots x_n$, the element x_1 is $\lhd$-minimal, so by Lemma 4 we know $x_1 \in \mathit{Start}(P)$ and x_1 is a suitable first element for a linear extension generated by the starty algorithm.

Suppose the starty algorithm did choose $x_1 \ldots x_i$, the first i elements as in Λ_3. Let $Q = P \setminus \{x_1, \ldots, x_i\}$.

If x_{i+1} is $\lhd$-minimal in Q, then the starty algorithm may choose x_{i+1}. Otherwise x_{i+1} must be an element of $N = \mathit{Suc}(x_i) \cap \mathit{Min}(Q)$ and the algorithm Lin_Ext_3 (in a certain repetition of the while loop) found the decomposition $x_1 \ldots x_i + \Box_N + \mathcal{L}(Q)$ and selected x_{i+1} as a starting element. Hence, again, x_{i+1} may be choosen by the starty algorithm. □

As announced before, we now look for an expression of $s(\Lambda_3)$ containing α and β. We begin with the introduction of new variables. Let γ count the number of times the while loop is repeated in a run of Lin_Ext_3, i.e. γ is the number of boxes the algorithm finds on its way through the list $\mathcal{L}$. Let γ_0 count the number of times the condition $N \cap \mathit{Start}(Q) \neq \emptyset$ appears false, i.e. γ_0 is the number of boxes kept empty.

We now turn to the remaining $\gamma - \gamma_0$ boxes, i.e. to the boxes filled up with some starting element n. In each of these cases the end of $\mathcal{L}$, i.e. $\mathcal{L}' = \mathcal{L}(Q)$ is replaced by $\mathcal{L}(Q_n)$. A close look at the auxiliary algorithm enables us to characterize the transition $\mathcal{L}(Q) \to \mathcal{L}(Q_n)$. Since $n \in \mathit{Min}(Q)$, but n is not $\lhd$-minimal in Q, we know that in $\mathcal{L}'$ the element n is preceded by an element x that is incomparable to n. If $\mathit{Suc}(x) \neq \mathit{Suc}(n)$, i.e. $\mathit{Suc}(x) \supset \mathit{Suc}(n)$, then in $\mathcal{L}'$ we find a box between x and n and possible patterns for the transition are:

$$x\Box_{N_i} \mid n\Box_{N_{i+1}} \mid y \;\to\; xy \tag{1}$$

$$x\Box_{N_i} \mid n\Box_{N_{i+1}} \mid y \;\to\; x\Box_{N_i \cup N_{i+1}} \mid y \tag{2}$$

$$x\Box_N \mid ny \;\to\; xy \tag{3}$$

$$x\Box_N \mid n \mid y \;\to\; x\Box_N \mid y \tag{4}$$

Remark. These transitions correspond to:
(1) $\mathit{Suc}(n) \supset \mathit{Suc}(y), y \notin \mathit{Suc}(n), y \in \mathit{Suc}(x)$.
(2) $\mathit{Suc}(n) \supset \mathit{Suc}(y), y \notin \mathit{Suc}(n), y \notin \mathit{Suc}(x)$.

(3) $Suc(n) \supset Suc(y), y \in Suc(n)$.
(4) $Suc(n) = Suc(y)$.
Note that by the definition of $min_{\lhd}$ we have $Pred(n) \supseteq Pred(x)$,
so $n \notin Suc(x)$ implies $y \notin Suc(x)$ in cases (2) and (4).

Now let α_1, β_1 be the jumpcounters for $\Lambda_1(L + \mathcal{L}(Q))$ and α_2, β_2 be those for $\Lambda_1(L + n + \mathcal{L}(Q_n))$. Depending on the pattern of the transition $\mathcal{L}(Q) \to \mathcal{L}(Q_n)$ we find

$\alpha_2 = \alpha_1$; $\beta_2 = \beta_1 - 2$ in case 1,
$\alpha_2 = \alpha_1$; $\beta_2 = \beta_1 - 1$ in case 2 and 3,
$\alpha_2 = \alpha_1 - 1$; $\beta_2 = \beta_1$ in case 4.

If $Suc(n) = Suc(x)$ then the jump $x|n$ between x and n is counted by α. When n is pulled foreward, then with respect to the rest of the list, x takes the role of n. Thus the new jumpcounters are $\alpha_2 = \alpha_1 - 1$ and $\beta_2 = \beta_1$.

Now partition the $\gamma - \gamma_0$ starting elements n, which have been pulled foreward by the algorithm, according to the type of transition $\mathcal{L}(Q) \to \mathcal{L}(Q_n)$. Let γ_1 count the transitions of type 1, γ_2 count the transitions of type 2 and 3 and let γ_3 count the remaining transitions. With these definitions the validity of the following equations is obvious

$$\gamma = \gamma_0 + \gamma_1 + \gamma_2 + \gamma_3,$$
$$\beta = \gamma + 2\gamma_1 + \gamma_2 = \gamma_0 + 3\gamma_1 + 2\gamma_2 + \gamma_3.$$

At this point we can express $s(\Lambda_3)$ in terms of α, β and the γ_i.

Lemma 9 *The jumpnumber of* Λ_3 *is*
$$s(\Lambda_3) = (\alpha - \gamma_3) + (\beta - 2\gamma_1 - \gamma_2) = (\alpha - \gamma_3) + \gamma.$$

Proof. In Λ_1 we distinguish between the jumps counted by α and those counted by β. If we transform Λ_1 step by step to Λ_3, i.e. for each of the γ repetitions of the while loop in Lin_Ext_3 we considere $\Lambda_1(\mathcal{L})$ and observe, which jumps disappear. We see that only jumps on the right side of the current box are disappearing. Since number and type of dissappearing jumps depend on the transition $\mathcal{L}(Q) \to \mathcal{L}(Q_n)$ only, they are counted by the γ_i. Altogether we note the disappearence of γ_3 jumps counted by α and $2\gamma_1 + \gamma_2$ jumps counted by β. This proves the Lemma. □

Theorem 1 *If* $\Lambda_2 = \Lambda_2(\mathcal{L}(P))$, *then* $s(\Lambda_2) \leq \frac{3}{2} s(P)$.

Proof. We saw that $s(\Lambda_2) \leq \alpha + \beta/2$ (Lemma 7) as well as $s(P) = s(\Lambda_3) = \alpha - \gamma_3 + \gamma$ (Lemma 8 and Lemma 9). Using $\gamma = \gamma_0 + \gamma_1 + \gamma_2 + \gamma_3$ we may rewrite this as $s(P) = \alpha + \gamma_0 + \gamma_1 + \gamma_2$. To prove the assertion it thus suffices to show

$$\alpha + \frac{\beta}{2} \quad \leq \quad \frac{3}{2}(\alpha + \gamma_0 + \gamma_1 + \gamma_2),$$

or equivalently

$$\beta \quad \leq \quad \alpha + 3\gamma_0 + 3\gamma_1 + 3\gamma_2.$$

Note that $\gamma_3 \leq \alpha$ (c.f. the proof of Lemma 9) and trivialy $0 \leq \gamma_0, \gamma_2$. This transforms the equality

$$\beta \quad = \quad \gamma_0 + 3\gamma_1 + 2\gamma_2 + \gamma_3,$$

into the desired inequality

$$\beta \quad \leq \quad 3\gamma_0 + 3\gamma_1 + 3\gamma_2 + \alpha.$$

□

REFERENCES

M. Chein and P. Martin [1972]: *Sur le nombre de saunts d'une forêt* , C. R. Acad. Sc. Paris, t. 275, série A 159-161.

D. Duffus, I. Rival and P. Winkler [1982]: *Minimizing setups for cycle-free ordered sets*, Proc. Amer. Math. Soc. 85, 509-513.

U. Faigle and R. Schrader [1985]: *A setup heuristic for interval orders*, Oper. Res. Letters 4, 185-188.

U. Faigle, G. Gierz and R. Schrader [1985]: *Algorithmic approaches to setup minimization*, SIAM J. Comput. Vol.14, No.4, 954-965.

T. Ghazal, A. Sharary and N. Zaguia [1988]: *On Minimizing the Jump Number for Interval Orders*, Order 4, 341-350.

W. Pulleyblank [1982]: *On minimizing setups in precedence constrained scheduling*, unpublished manuscript.

I. Rival [1983]: *Optimal linear extensions by interchanging chains*, Proc. Amer. Math. Soc. 89, 387-394.

G. Steiner [1985]: *On finding the jump number of a partial order by substitution decomposition*, Order 2, 9-23.

G. Steiner and L. Stewart [1987]: *A linear time algorithm to find the jump number of 2–dimensional bipartite orders*, Order ss, 359-367.

M. M. Sysło [1984]: *Minimizing the jump number for partially ordered sets: a graph-theoretic approach*, Order 1, 7-19.