FOCUS: A Constraint for Concentrating High Costs

Thierry Petit

CP2012, Quebec City, 8-12th October 2012

TASC (Mines Nantes – LINA – INRIA – CNRS)

Motivations

- Many optimization problems involve side constraints that are complementary to an optimization criterion
  - Fair distribution of values and balanced solutions
    - Petit et al, ICTAI'2000, Pesant and Régin CP2005, Schaus et al, CPAIOR2007, Schaus et al, CPAIOR2007, Petit and Régin IJAIT2011
  - Reverse concept?

Motivations

- In this paper: we capture the concept of concentrating high costs values in a limited number of areas
- Global constraint with an optimal time complete filtering algorithm

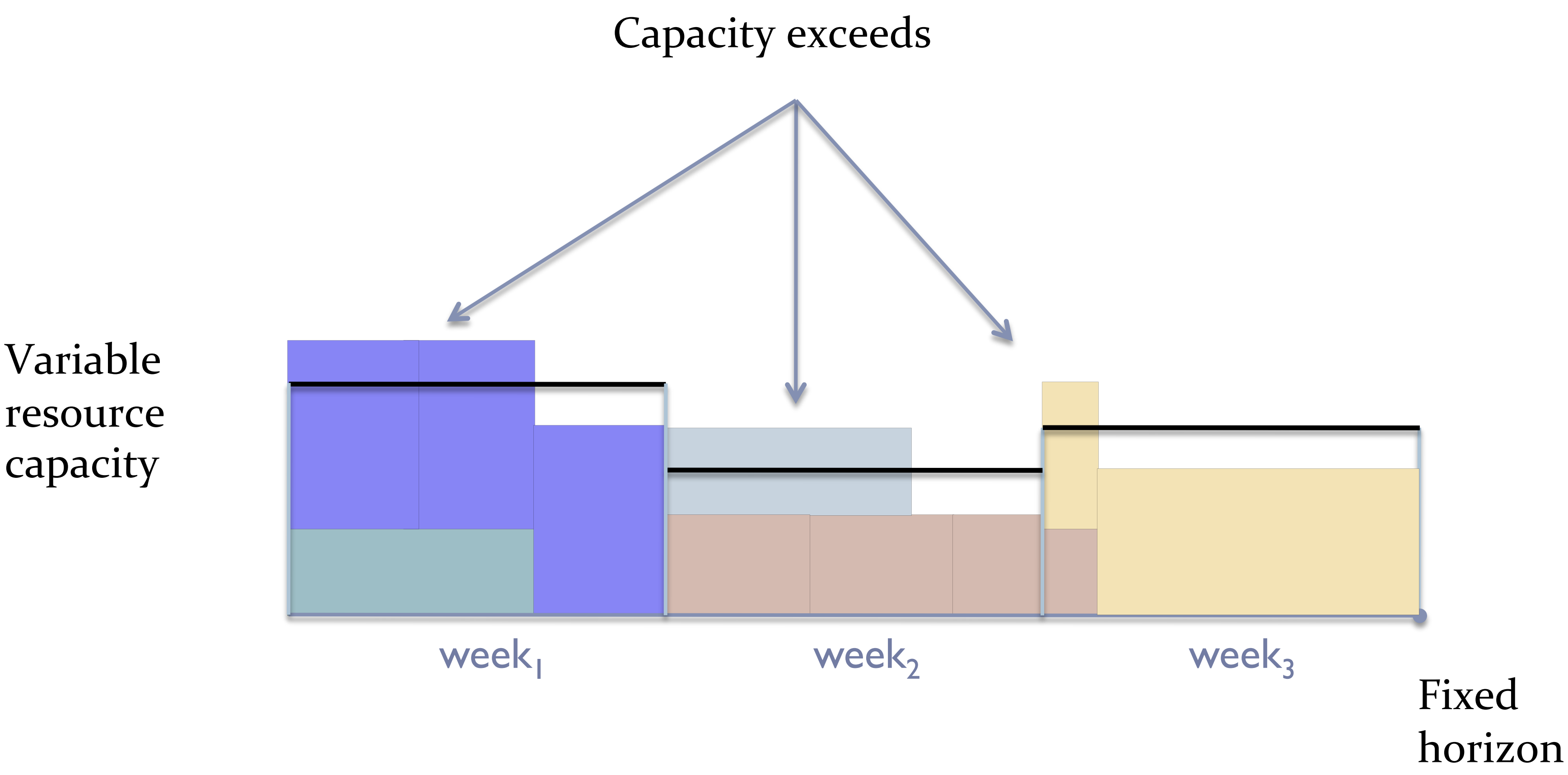
Outline

1. Two examples of use
2. The Focus constraint
3. The O(n) complete filtering algorithm
4. Experiments

Musical Problems

- Generation of chords sequences with rules
  - Sorting Chords problem (TSP)
    - A Chord is made of k notes
    - Generate a set of sorted chords which are at most as possible successively pairwise distinct (Pachet et al, Constraint'2001, Truchet and Codognet, SoftComp'2004)
- The musicians want large sequences of successive chords that do not violate their rules
- Concentrate high violations of their rules in a small number of areas in the music score

Cumulative Scheduling with Rentals



Cumulative Scheduling with Rentals

- Rental of an additional machine to produce the resource

Cumulative Scheduling with Rentals

- Rental of an additional machine to produce the resource
  - Packaged rentals are less expensive
    - If you rent the machine during 3 consecutive weeks, the price will be lesser that the price of 3 separated rentals

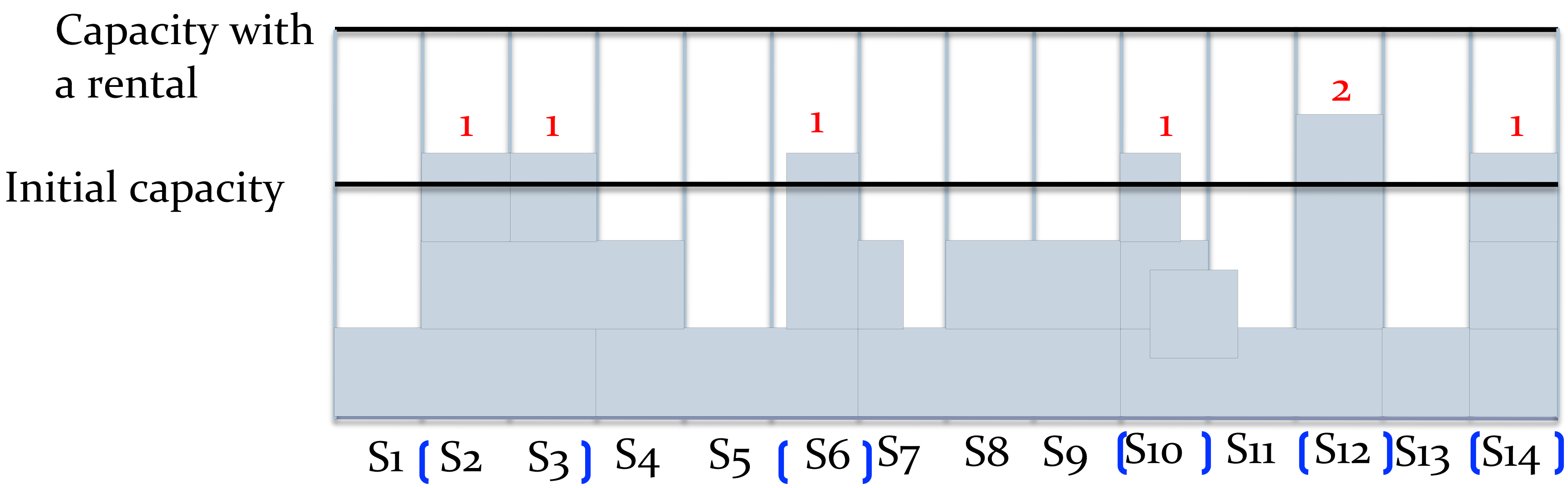
Cumulative Scheduling with Rentals

- Rental of an additional machine to produce the resource
  - Packaged rentals are less expensive
    - If you rent the machine during 3 consecutive weeks, the price will be lesser that the price of 3 separated rentals
  - The maximum duration of one package is generally limited
    - If you rent the machine during more than 3 weeks, for instance 4 weeks, then you have to sign two separate rental contracts

Cumulative Scheduling with Rentals

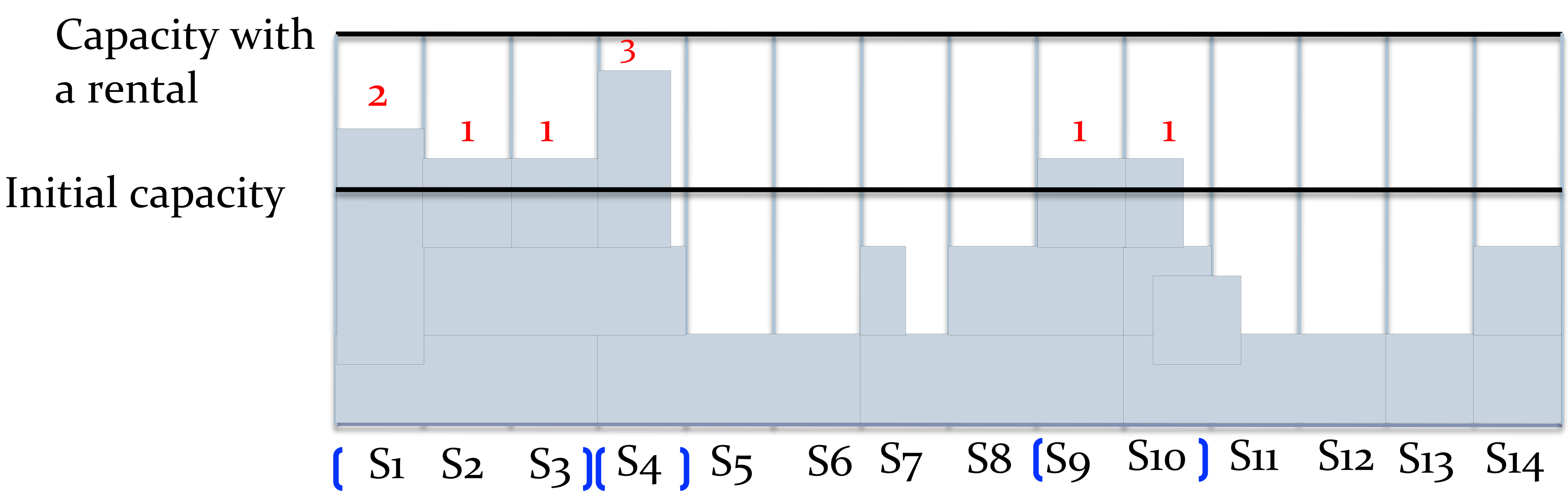
- Rental of an additional machine to produce the resource
  - Packaged rentals are less expensive
    - If you rent the machine during 3 consecutive weeks, the price will be lesser that the price of 3 separated rentals
  - The maximum duration of one package is generally limited
    - If you rent the machine during more than 3 weeks, for instance 4 weeks, then you have to sign two separate rental contracts
- Concentrate excess of resource in a small number of packages

Without a Focus Constraint



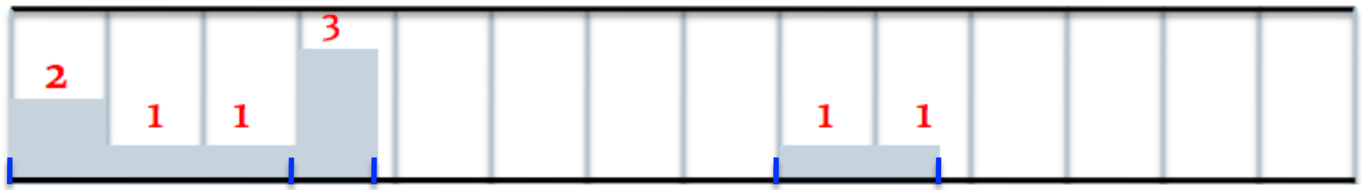
Hypothesis: the maximum duration of one package is 3 weeks.
5 distinct rental contracts
obj = 7

With a Focus Constraint



Hypothesis: the maximum duration of one package is 3 weeks.
3 distinct rental contracts
obj = 9

The Focus Constraint



$X = < x_0, x_1, x_2, x_3, \ldots \quad x_{12}, x_{13} >$

Focus(X, y_c = 3, len = 3, k = 0) is satisfied

At most 3 disjoint sequences

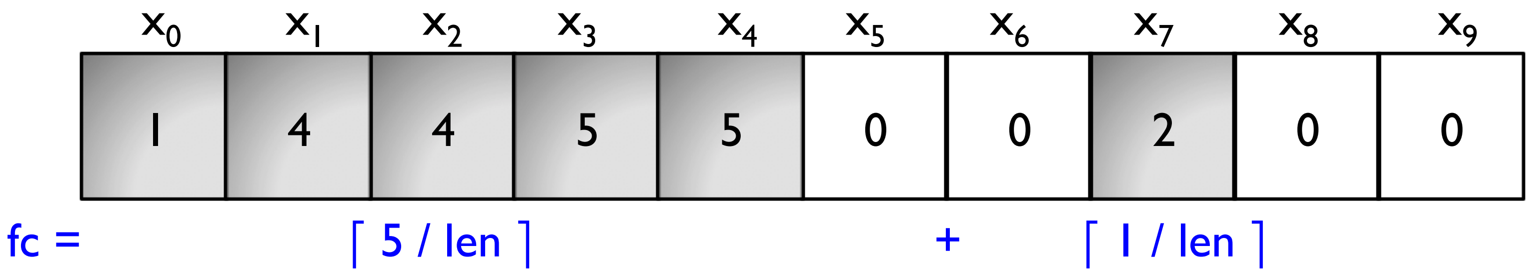
of length ≤ 3

where variables take values > 0

Checker

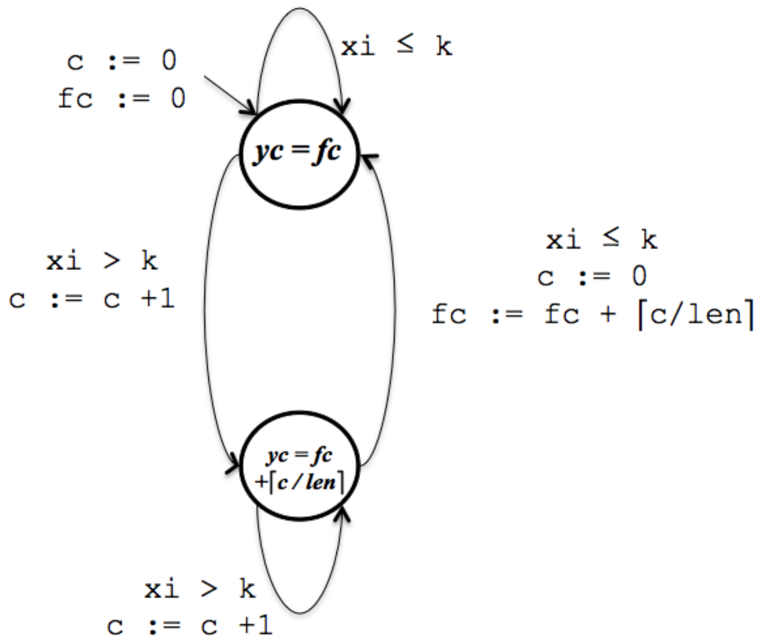
- P-sequence: Sequence where all the variables take a value v > *k*
- Focus cardinality: Minimum number of P-Sequences of length at most *len*, given the current domains of variables in X

Checker



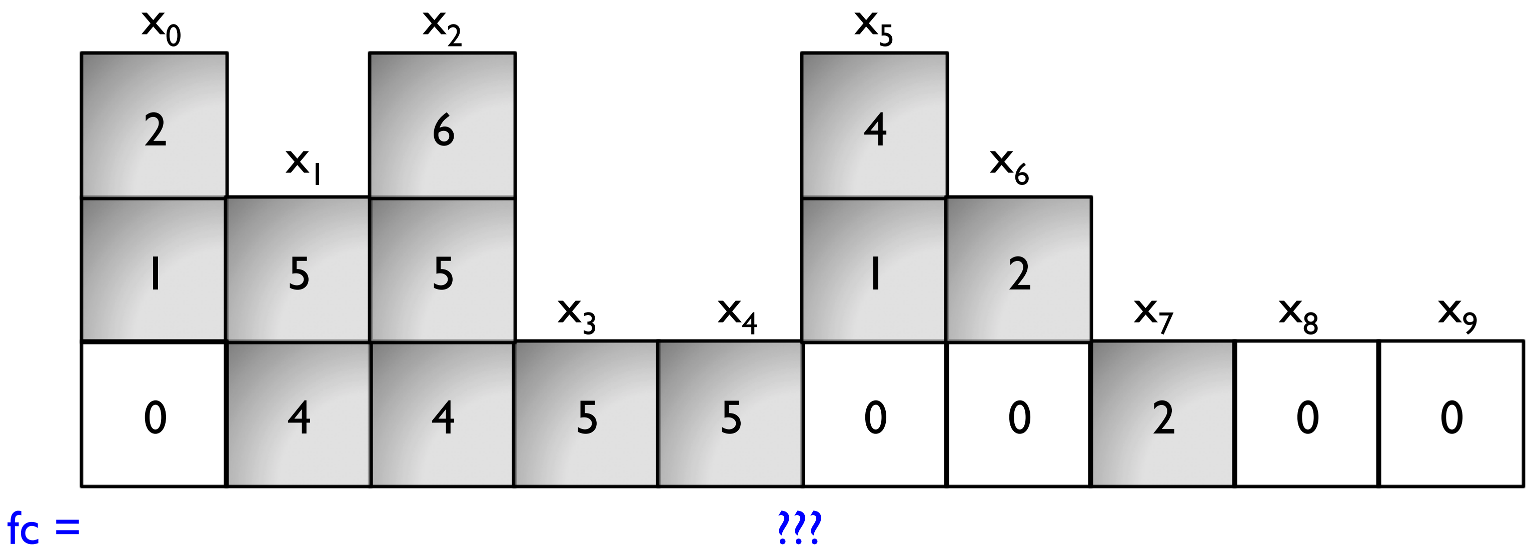
For instance if *len*=3 then the minimum number of P-sequences is 3

Checker




```
Integer nb := 0;
Integer size := 0;
Boolean prevpk := false;
for Integer i := 0; i < n; i := i + 1 do
    if min(x_i) > k then
        size := size + 1;
        prevpk := true;
    else
        if prevpk then  nb := nb + ⌈size/len⌉;
        size := 0;
        prevpk := false;
if prevpk then  nb := nb + ⌈size/len⌉;
return nb ≤ v_c; // focus cardinality of X
```

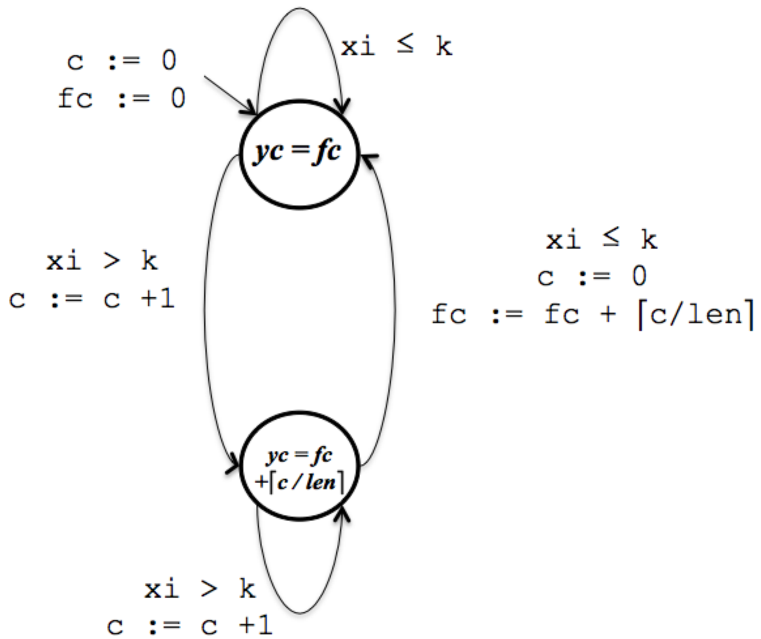
Filtering Algorithm



Three possible status for a given variable:

- $> k$
- $\leq k$
- undetermined

Checker



Complete FA based on that checker : $O(n^3)$

Filtering Algorithm

- Principle: traverse variables in X from x_0 to x_{n-1}
- For each x_i maintain two quantities
  - Focus cardinality of prefix $< x_0, x_1, \ldots, x_i >$, if $x_i \leq k$
  - Focus cardinality of prefix $< x_0, x_1, \ldots, x_i >$, if $x_i > k$

Filtering Algorithm

- Principle: traverse variables in X from x_0 to x_{n-1}
- For each x_i maintain two quantities
  - Focus cardinality of prefix $< x_0, x_1, \ldots, x_i >$, if $x_i \leq k$
  - Focus cardinality of prefix $< x_0, x_1, \ldots, x_i >$, if $x_i > k$
  - Feasibility check: $i = n - 1$

Filtering Algorithm

- Principle: traverse variables in X from x_0 to x_{n-1}
- For each x_i maintain two quantities
  - Focus cardinality of prefix < $x_0, x_1, \ldots, x_i$ >, if $x_i \leq k$
  - Focus cardinality of prefix < $x_0, x_1, \ldots, x_i$ >, if $x_i > k$
  - Feasibility check: $i = n - 1$

Algorithm 2: ISSATISFIED($X = \langle x_0, x_1, \ldots, x_{n-1} \rangle, y_c, len, k$): boolean

1 return $\min(\underline{p}(x_{n-1}, v_>), \underline{p}(x_{n-1}, v_\leq)) \leq \max(y_c)$;

$\Rightarrow$ Linear filtering algorithm ?

Filtering Algorithm

- Two incremental computations
  - → *cards*: prefix < $x_0, x_1, \ldots, x_i$ >
  - ← *sdrac*: suffix < $x_i, x_{i+1}, \ldots, x_{n-1}$ >
- Aggregation of the result on x_i

Filtering Algorithm

- The incremental computation requires an additional data
  - For each x_i the minimal length $plen_i$ of a P-sequence containing x_i within an assignment corresponding to the focus cardinality ($plen_i > 0 \Leftrightarrow x_i > k$)

Filtering Algorithm

- We thus compute three quantities :
  - cards[i][0] = Prefix focus cardinality < $x_0, x_1, \ldots, x_i$ > if $x_i \leq k$
  - cards[i][1] = Prefix focus cardinality < $x_0, x_1, \ldots, x_i$ > if $x_i > k$
  - cards[i][2] = $plen_i$

k=0, *len* = 3

$D(x_0) = D(x_1) = D(x_3) = D(x_5) = \{0,1\}$
$D(x_2) = D(x_4) = D(x_6) = \{1\}$

	x_0	x_1	x_2	x_3	x_4	x_5	x_6
cards[0]	0	0	-	1	-	1	-
cards[1]	1	1	1	1	1	2	2
cards[2]	1	1	1	2	3	1	1

Filtering Algorithm

- We thus compute three quantities :
  - cards[i][0] = Prefix focus cardinality < $x_0, x_1, \ldots, x_i$ > if $x_i \leq k$
  - cards[i][1] = Prefix focus cardinality < $x_0, x_1, \ldots, x_i$ > if $x_i > k$
  - cards[i][2] = $plen_i$

k=0, *len* = 3

$D(x_0) = D(x_1) = D(x_3) = D(x_5) = \{0,1\}$
$D(x_2) = D(x_4) = D(x_6) = \{1\}$

	x_0	x_1	x_2	x_3	x_4	x_5	x_6
cards[0]	0	0	-	1	-	1	-
cards[1]	1	1	1	1	1	2	2
cards[2]	1	1	1	2	3	1	1

Support with y_c = 2 : 0 0 1 <u>1</u> 1 0 1

Filtering Algorithm

O(n)

Algorithm 3: MinCards($X = \langle x_0, x_1, \ldots, x_{n-1} \rangle$, len, k): Integer matrix

```
cards := new Integer[|X|][3] ;
if min(x_0) ≤ k ∧ max(x_0) > k then
    cards[0][0] := 0;
    cards[0][1] := 1;
    cards[0][2] := 1;
else
    if min(x_0) > k then
        cards[0][0] := n + 1;
        cards[0][1] := 1;
        cards[0][2] := 1;
    else
        cards[0][0] := 0;
        cards[0][1] := n + 1;
        cards[0][2] := 0;
for Integer i := 1; i < n; i := i + 1 do
    if max(x_i) > k then
        if min(x_i) > k then  cards[i][0] := n + 1;
        else  cards[i][0] := min(cards[i − 1][0], cards[i − 1][1]);
        if cards[i − 1][2] = 0 ∨ cards[i − 1][2] = len then
            cards[i][1] := min(cards[i − 1][0] + 1, cards[i − 1][1] + 1);
            cards[i][2] := 1;
        else
            cards[i][1] := min(cards[i − 1][0] + 1, cards[i − 1][1]);
            if cards[i − 1][1] < cards[i − 1][0]+1 then  cards[i][2] := cards[i−1][2]+1;
            else  cards[i][2] := 1;
    else
        cards[i][0] := min(cards[i − 1][0], cards[i − 1][1]);
        cards[i][1] := n + 1;
        cards[i][2] := 0;
return cards;
```

Filtering Algorithm

- Two incremental computations: *done in O(n)* ☺
  - *cards*: prefix < x_0, x_1, …, x_i >
  - *sdrac*: suffix < x_i, x_{i+1}, …, x_{n-1} >
- Aggregation of the result on x_i
  - Based on a regret mechanism (sum "plen" values)

Filtering Algorithm

Algorithm 4: FILTER($X = \langle x_0, x_1, \dots, x_{n-1} \rangle, y_c, len, k$): Set of variables

$cards := \text{MinCards}(X, len, k)$;
Integer $lb := \min(cards[n-1][0], cards[n-1][1])$;
if $\min(y_c) < lb$ **then** $D(y_c) := D(y_c) \setminus [\min(y_c), lb[$;
if $\min(y_c) = \max(y_c)$ **then**
    $sdrac := \text{MinCards}(\langle x_{n-1}, x_{n-2}, \dots, x_0 \rangle, len, k)$;
    for Integer $i := 0$; $i < n$; $i := i + 1$ **do**
        if $cards[i][0] + sdrac[n-1-i][0] > \max(y_c)$ **then**
            $D(x_i) := D(x_i) \setminus [\min(x_i), k]$;
        Integer $regret := 0$;
        if $cards[i][2] + sdrac[n-1-i][2] - 1 \leq len$ **then** $regret := 1$;
        if $cards[i][1] + sdrac[n-1-i][1] - regret > \max(y_c)$ **then**
            $D(x_i) := D(x_i) \setminus]k, \max(x_i)]$;
return $X \cup \{y_c\}$;

GAC in O(n)

Filtering Algorithm

Algorithm 4: FILTER($X = \langle x_0, x_1, \ldots, x_{n-1} \rangle, y_c, len, k$): Set of variables

```
cards := MinCards(X, len, k) ;
Integer lb := min(cards[n − 1][0], cards[n − 1][1]);
if min(y_c) < lb then  D(y_c) := D(y_c) \ [min(y_c), lb[;
if min(y_c) = max(y_c) then
    sdrac := MinCards(⟨x_{n−1}, x_{n−2}, ..., x_0⟩, len, k) ;
    for Integer i := 0; i < n; i := i + 1 do
        if  cards[i][0] + sdrac[n − 1 − i][0] > max(y_c) then
            D(x_i) := D(x_i) \ [min(x_i), k];
        Integer regret := 0;
        if cards[i][2] + sdrac[n − 1 − i][2] − 1 ≤ len then  regret := 1;
        if cards[i][1] + sdrac[n − 1 − i][1] − regret > max(y_c) then
            D(x_i) := D(x_i) \ ]k, max(x_i)];
return X ∪ {y_c};
```

GAC in O(n)

Intuition: we prove that changing the value of one variable modifies the focus cardinality of at most one

Impact of the GAC Algorithm of Focus (optimum solutions w.r.t the sum of the number of shared notes)

Instances			FOCUS(X, y_c, len, k)				CHECKER(X, y_c, len, k)			
nb. of chords	$\overline{y_c}$-len -k -$nmax$	#optimum with $sum > 0$	average #backtracks (of 100)	average #fails (of 100)	max. #backtracks (of 100)	average time (ms) (of 100)	average #backtracks (of 100)	average #fails (of 100)	max. #backtracks (of 100)	average time (ms) (of 100)
8	1-4-0-3	66	61	46	462	11	1518	951	15300	17
8	1-4-1-3	66	45	33	342	1	91	59	1724	2
8	2-4-0-3	66	47	34	247	1	58	42	455	1
8	2-4-1-3	66	45	33	301	1	44	32	349	1
8	1-6-0-4	84	198	141	2205	12	15952	10040	86778	213
8	1-6-1-4	84	114	79	767	3	1819	1117	45171	24
8	2-6-0-4	84	127	88	620	3	2069	1361	64509	28
8	2-6-1-4	84	118	81	724	3	250	168	7027	4
8	1-8-0-5	98	261	184	1533	6	39307	25787	167575	566
8	1-8-1-5	98	148	103	662	3	11821	7642	94738	168
8	2-8-0-5	98	164	113	803	4	21739	14400	173063	317
8	1-8-0-5	98	183	127	882	4	10779	6939	92560	153
8	1-8-0-6	99	290	203	1187	18	46564	30488	130058	690
8	1-8-1-6	99	238	166	1167	11	29256	19150	134882	438
8	2-8-0-6	99	221	152	1458	6	29455	19607	123857	445
8	2-8-1-6	99	209	144	1118	9	21332	14095	117768	329
9	1-9-0-4	88	415	299	4003	18	214341	133051	1095734	3244
9	1-9-1-4	88	268	185	2184	6	12731	7988	751414	203
9	2-9-0-4	88	270	188	2714	6	22107	14065	374121	337
9	2-9-1-4	88	266	182	3499	6	1364	941	92773	23
9	1-9-0-5	97	574	407	2437	26	360324	230167	1355934	6584
9	1-9-1-5	97	404	273	1677	11	62956	40277	881441	1150
9	2-9-0-5	97	451	309	3327	12	228072	147007	1124630	4263
9	2-9-1-5	97	386	260	1698	10	58421	37589	989900	1079

Hardness of Instances Using Focus (optimum solutions)

Instances				with FOCUS				without FOCUS			
nb. of chords	$\overline{y_c}$ -len -k -$nmax$	#optimum with $sum > 0$	#optimum equal with and without FOCUS	max. value of sum	average #backtracks (of 100)	max. #backtracks (of 100)	average time (ms) (of 100)	max. value of sum	average #backtracks (of 100)	max. #backtracks (of 100)	average time (ms) (of 100)
6	2-4-0-4	88	99	7	23	76	3	7	22	76	1
8	2-4-0-4	84	95	8	131	618	4	7	115	449	3
10	2-4-0-4	78	98	6	457	4579	11	6	360	3376	9
12	2-4-0-4	69	98	5	952	12277	28	5	1010	10812	27
16	2-4-0-4	43	100	4	4778	132019	153	4	6069	95531	189
20	2-4-0-4	7	100	3	15650	1316296	747	3	15970	1095399	679
6	2-4-0-6	97	96	13	37	113	1	13	37	121	1
8	2-4-0-6	99	93	11	218	1305	5	11	198	860	4
10	2-4-0-6	97	77	10	1247	5775	32	9	1159	10921	26
12	2-4-0-6	96	75	12	5092	34098	155	11	4844	54155	145
16	2-4-0-6	88	84	9	45935	724815	2002	9	73251	2517570	3407
20	2-4-0-6	79	91	8	264881	4157997	14236	6	174956	2918335	8284

First Solution

Instances			with FOCUS				without FOCUS			
nb. of chords	$\overline{y_c}$ -len -k -$nmax$	average gap in sum	min(sum) / max(sum) (of 100)	average #backtracks (of 100)	max. #backtracks (of 100)	average time (ms) (of 100)	min(sum) / max(sum) (of 100)	average #backtracks (of 100)	max. #backtracks (of 100)	average time (ms) (of 100)
50	4-6-2-4	14	45/73	0	16	84	55/94	0	0	80
100	8-6-2-4	27	100/145	1	57	1168	119/175	0	0	1413
50	5-6-1-4	30	19/69	212	12698	101	55/94	0	0	85
100	10-6-1-4	63	40/113	93	3829	1002	119/175	0	0	1407

Thank you!