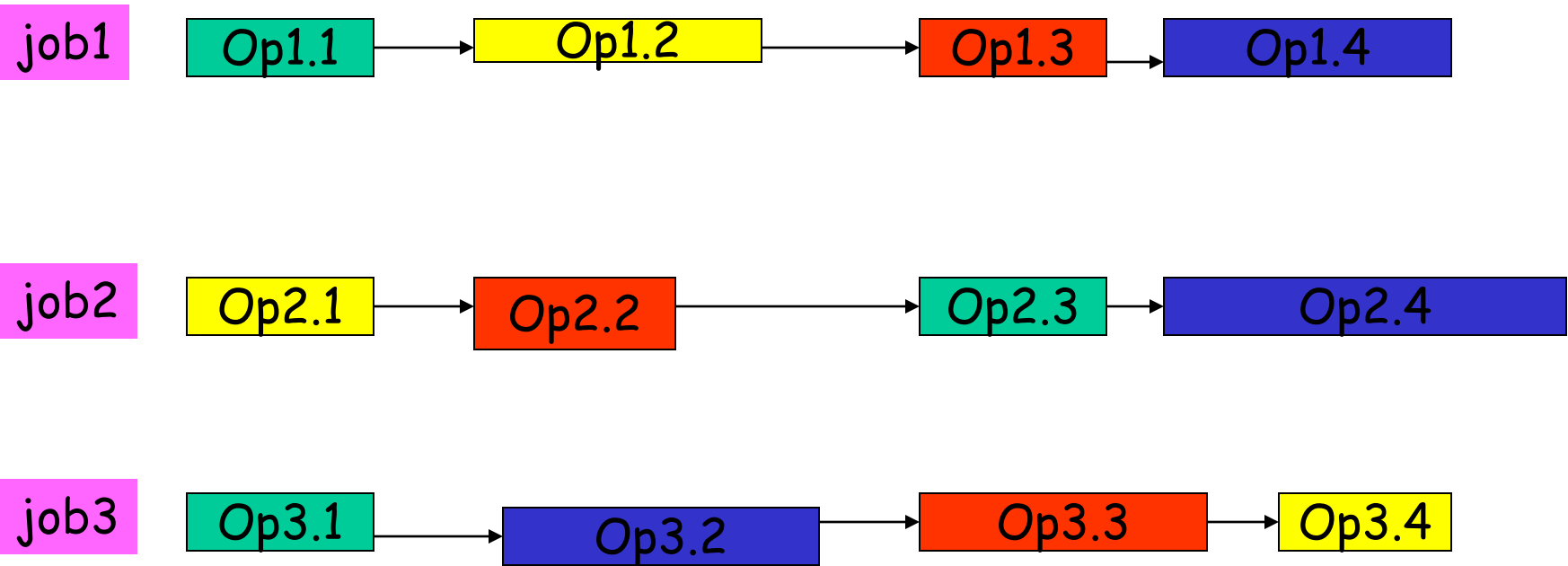


jobshop scheduling

We have

- a set of resources
- a set of jobs
 - a job is a sequence of operations/activities
- sequence the activities on the resources

An example: 3 x 4



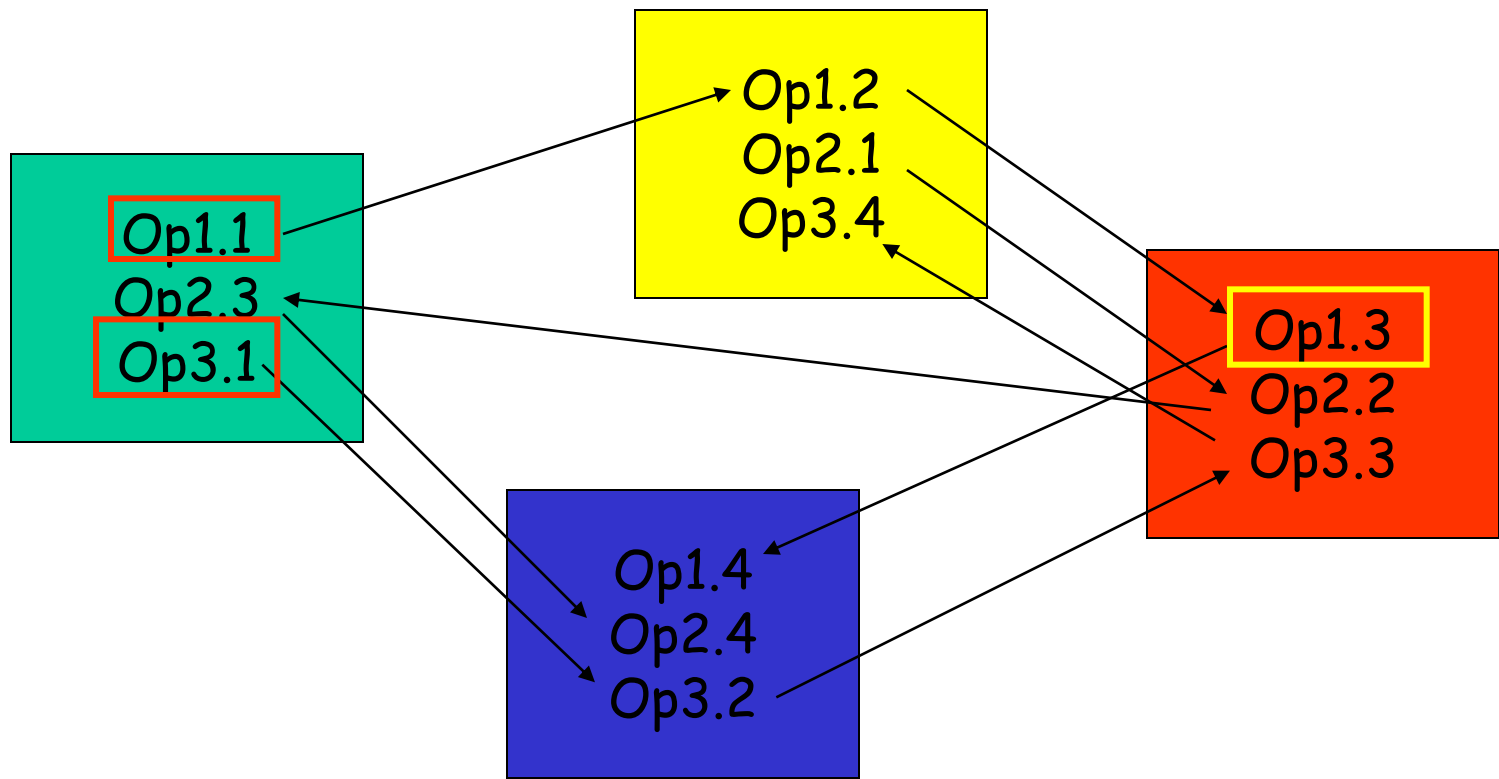
- We have 4 resources: green, yellow, red and blue
- a job is a sequence of operations (precedence constraints)
- each operation is executed on a resource (resource constraints)
- each resource can do one operation at a time
- the duration of an operation is the length of its box
- we have a due date, giving time windows for operations (time constraints)

An example: 3 x 4

Op1.1 Op1.2 Op1.3 Op1.4

Op2.1 Op2.2 Op2.3 Op2.4

Op3.1 Op3.2 Op3.3 Op3.4



Assign a start time to each operation such that

- (a) no two operations are in process on the same machine at the same time and
- (b) temporal constraints are respected

Alternatively ... sequence operations on resources

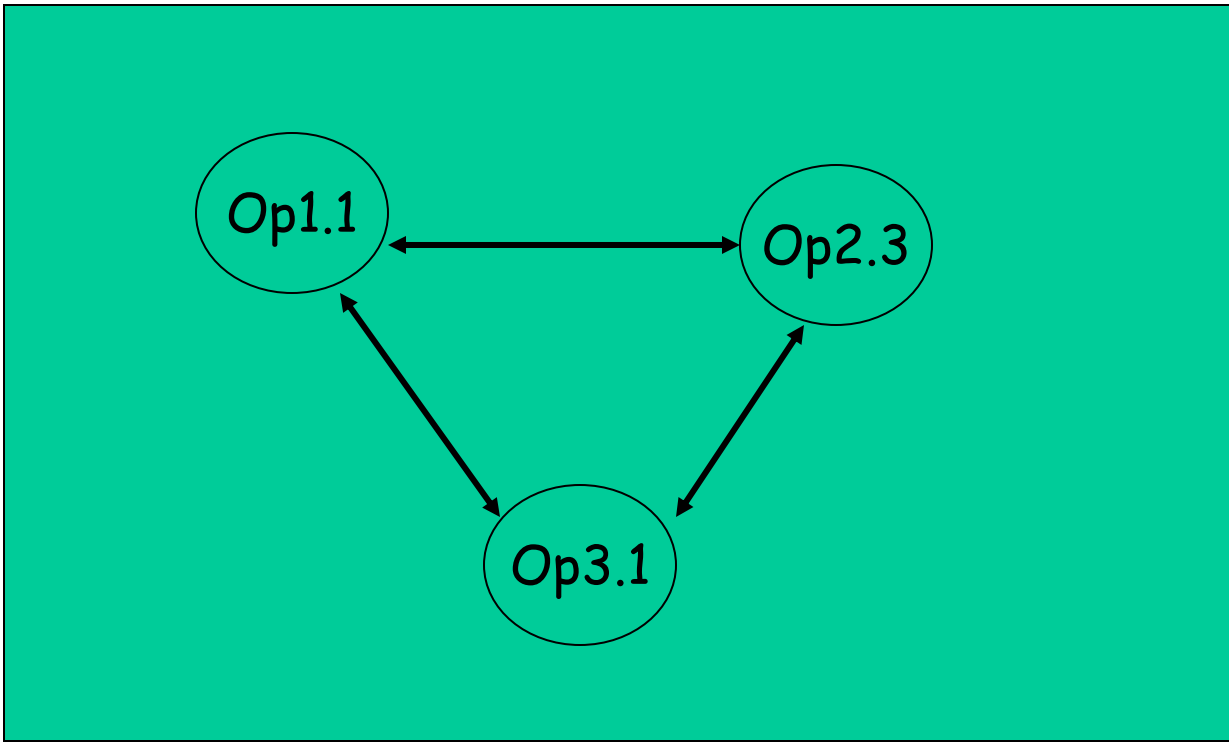
This gives a set of solutions, and might be considered a "least commitment approach"

An example: 3 x 4

Op1.1 Op1.2 Op1.3 Op1.4

Op2.1 Op2.2 Op2.3 Op2.4

Op3.1 Op3.2 Op3.3 Op3.4



On the "green" resource, put a *direction* on the arrows

A disjunctive graph

An example: 3 x 4

Op1.1 Op1.2 Op1.3 Op1.4

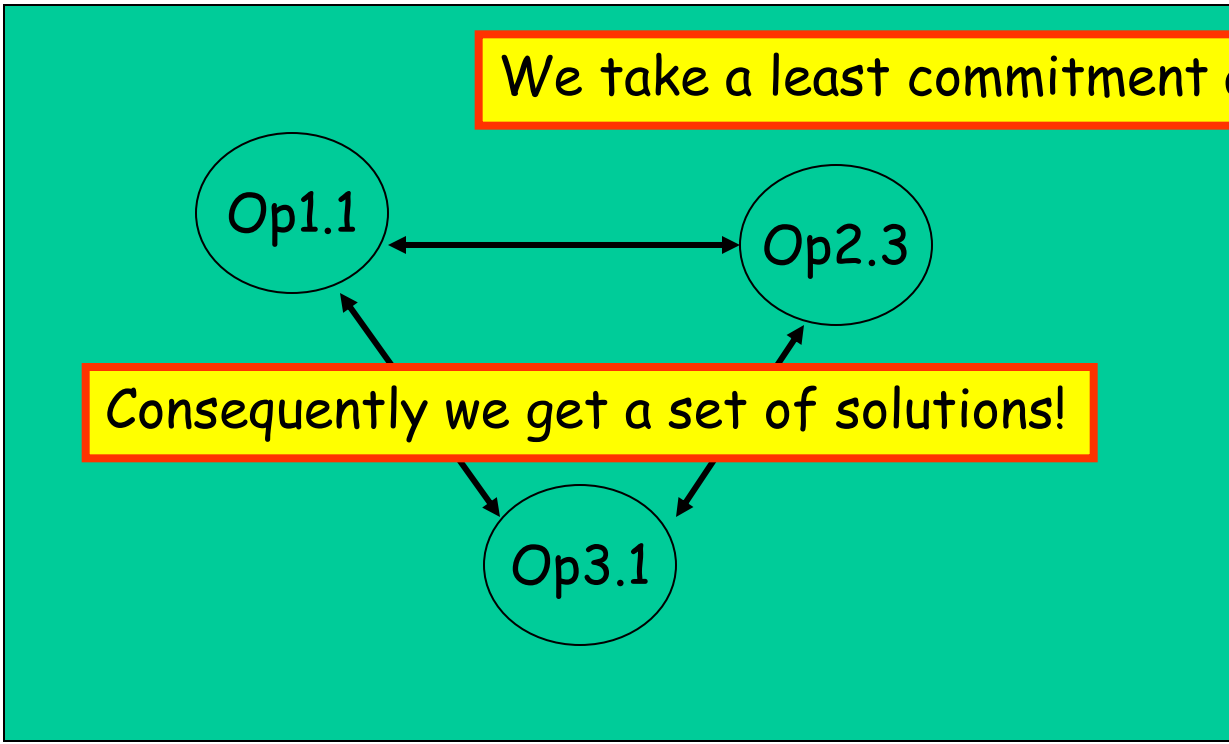
Op2.1 Op2.2 Op2.3 Op2.4

We do not bind operations to start times

Op3.1 Op3.2 Op3.3 Op3.4

We take a least commitment approach

Consequently we get a set of solutions!



On the "green" resource, put a *direction* on the arrows

A disjunctive graph

7x6

76

```
210316375346
182540500034
253458091147
150525334859
291345540331
133359004421
244402375213
```

What is makespan!

```
//
// a 7x6 job problem
// each row (task) is a job
// each job has 6 operations (activities in pair)
// each operation requires
// - an activity (to 5)
// - a duration in that activity (to 0)
//
// The problem is to find the shortest makespan
// and the span is the time required to complete all jobs
//
// minimum makespan is 57
```

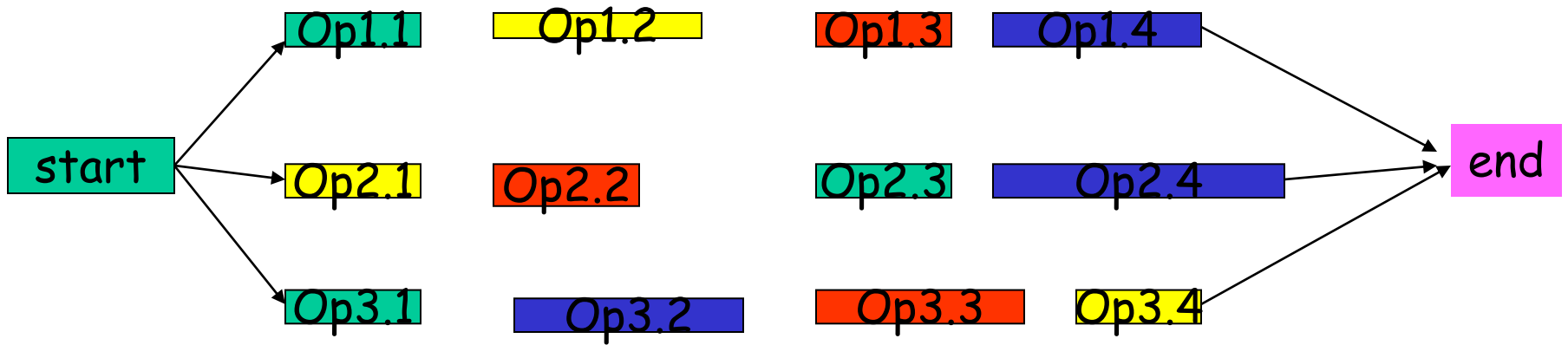
10

11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

For a long time, unsolved

Why bother?

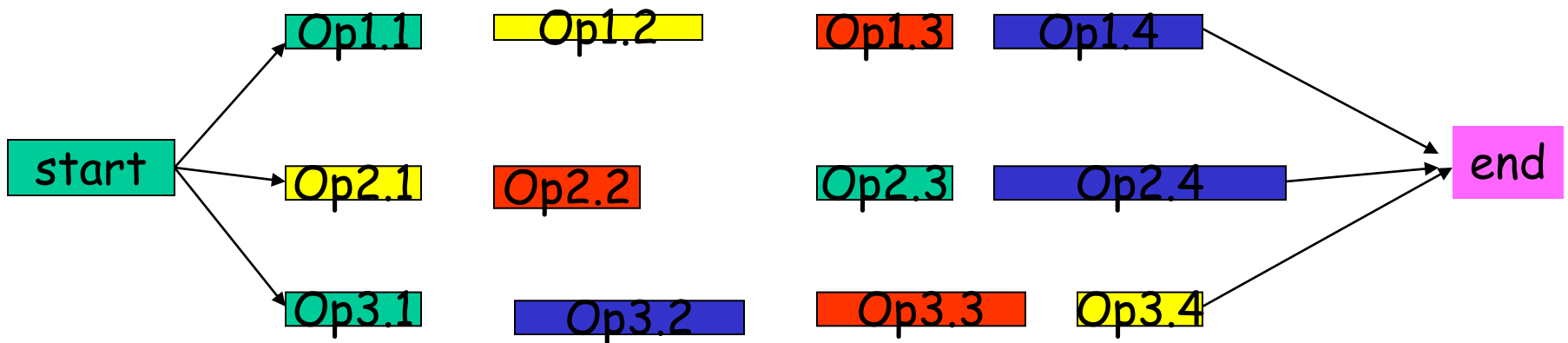
- Minimise makespan
 - what is makespan?
- Maximise start
 - JIT, minimise inventory levels
- minimise idle time on resources
 - maximise ROI
- ...



Find the smallest value for end
minimise makespan

How can we view this as a csp?

Each operation is a variable
domain is set of start times
there are precedence constraints between operation in a job
operations on a resource have *disjunctive constraints*



Complexity

What is the complexity of this problem?

- Assume we have m resources and n jobs
- on each resource we will have n operations
- we can order these in $n!$ ways
- therefore we have $O(n!^m)$ states to explore

$$O(n!^m)$$

But we want to optimise, not satisfy

How do you optimise with CP?

A sequence of decision problems

- Is there a solution with makespan 395?
 - Yip!
-
-
- Is there a solution with makespan 300?
 - Let me think about that ...
 - Yes
- Is there a solution with makespan 299?
 - Hold on, ... , hold on
 - NO!
- Minimum makespan is 300.

When optimising, via a sequence of decision problems, will all decisions be equally difficult to answer?

What does branch and bound (BnB) do ?

Who cares about jobshop scheduling?

Manufacturing inc.

Is CP any good for this class of problem?

- Main competitor is OR technology
- Mark Fox 1980's
 - ISIS constraint directed search(CMU)
- OPIS
 - Steve Smith, Peng Si Ow (CMU)
- DAS
 - Burke et al (SU)
- MicroBoss
 - Norman Sadeh (CMU)
- Edge-finding
 - a novel technique to infer constraints on a resources
 - J Carlier & E. Pinson 1994
 - CP solves open benchmarks, beats OR
- Texture based heuristics
 - J-C Beck and Mark Fox 1990's
- ILOG-Scheduler
 - Claude Le Pape, Wim Nuijten, Philippe Babbiste, J-Chris Beck
 - and many others
- 2000s
 - ILOG buy CPLEX

Is CP any good for this class of problem?

- 2000

- ILOG buy CPLEX

- ...

- 2009

- P. Vilim, edge-finding filtering

- 2010

- IBM buy ILOG

File Edit View History Bookmarks Tools Help

Guy Martin: what it's lik... x Index of /-pat/cpM... x job shop schedul... x Limited discrepanc... x

cpaior2015.uconn.edu

UConn UNIVERSITY OF CONNECTICUT

CPAIOR 2015

Home Conference Schedule Conference Details Program Local Info

CPAIOR 2015

The Twelfth International Conference on Integration of Artificial Intelligence (AI) and Operations Research

The aim of the conference series is to bring together interested researchers from constraint programming in the intersection of these fields and to provide an opportunity for researchers in one area to learn about the opportunity to show how the integration of techniques from different fields can lead to interesting res approaches from more than one of the areas are especially solicited. High quality papers from a single ar challenging applications or experience reports on such applications are strongly encouraged.

After a successful series of five CPAIOR international workshops in Ferrara (Italy), Paderborn (Germany), A conference. More than 100 participants attended the first meeting held in Nice (France). In the subsequent Pittsburgh (USA), Bologna (Italy), Berlin (Germany), Nantes (France) and Yorktown Heights (USA).

See the official webpage of the CPAIOR conference series for more information.

BARCELONA, SPAIN

File Edit View History Bookmarks Tools Help

Guy Martin: wh... x Index of /-pat/... x job shop sched... x Limited discrep... x JEA_16_1-6.dvi - a1... x Petr Vilim Homepage x Journal of Sche... x Home | CPAIOR 2015 x 25th Internation... x

icaps15.icaps-conference.org

icaps15
Jerusalem 2015
Jerusalem, Israel
June 7-11, 2015

Home Technical Program Workshops Tutorials Special Tracks Demos Doctoral Consortium Registration Accommodation

Conference Info Co-located Events Committees

Welcome

ICAPS 2015, the 25th International Conference on Automated Planning and Scheduling will take place in Jerusalem, Israel, June 7-11, 2015.

ICAPS 2015 is part of the ICAPS conference series. ICAPS is the premier forum for exchanging news and research results on theory and applications of intelligent planning and scheduling technology.

The conference features a pre-conference program of workshops and tutorials on current research topics. The main technical program consists of invited talks by leading scientists working in the area, presentations of technical papers, as well as system demonstrations. For graduate students the pre-conference program includes a Doctoral Consortium.

The detailed conference schedule is available here.

The program leaflet (pdf) is available here.

The social events include a reception on Monday night at the Tower of David followed by the Night Spectacular sound and lights show, a walking tour of the old city of Jerusalem Tuesday night, and the Banquet at Beit-Shmuel's banquet hall overlooking the old city of Jerusalem.

ICAPS-2015 will be co-located with the 8th Annual Symposium on Combinatorial Search (SOCS-15) including a joint session on Thursday June, 11.

A free shuttle bus will take SOCS participants attending the joint ICAPS-SOCS session to the SOCS venue in Ein Gedi on Thursday afternoon.

An advertising flier (PDF) for the conference is available here.

Important Dates

News and Updates

[01 June 2015] The program leaflet (pdf) is available.

[22 May 2015] All workshops proceedings are now available.

[27 May 2015] The DC schedule is now available.

[22 May 2015] All workshops schedules are now available.

[20 May 2015] The prereception optional walking tours registration is now open. See here the details.

[20 May 2015] Detailed information on the welcome reception and prereception optional walking tours is now available here.

[20 May 2015] The list of posters and demos is now available in the detailed conference schedule. Please note poster board size is 90cm x 150cm.

[13 May 2015] The detailed conference schedule is now available in an online

File Edit View History Bookmarks Tools Help

Guy Martin: what it's lik... x Index of /-pat/cpl/cho... x job shop scheduling co... x Limited discrepancy sea...

link.springer.com/journal/10951

Springer Link

Search

Home Contact Us

Browse Volumes & Issues

Journal of Scheduling

ISSN: 1094-6136 (Print) 1099-1425 (Online)

Description

The Journal of Scheduling provides a global forum for the publication of all forms of scheduling research. It is the only peer reviewed journal with broad coverage of the techniques and applications of scheduling that spans several distinct disciplines.

Readers facing complex scheduling problems can turn to the journal to find the latest advances in the field. Each issue features new and novel techniques, applications, there ... [show all](#)

Find your Volume or Issue

Volume Issue

Browse All Content

Latest Articles

OriginalPaper
An efficient algorithm for semi-online multiprocessor scheduling with given total processing time
Hans Kellerer, Vladimir Kotov, Michael Gabay (December 2015)
[Download PDF \(538KB\)](#) [View Article](#)

OriginalPaper
Identical coupled task scheduling: polynomial complexity of the cyclic case
Vangelis L. Labouliere, Maria Branner, Gerd Ertl (December 2016)

Impact Factor 1.028 Available 2003 - 2015

Volumes 13 Issues 77

Articles 592 Open Access 24 Articles

Other actions

Register for Journal Updates [↗](#)

About This Journal [↗](#)

Share

nature publishing group
language editing

Eliminate language errors from your academic writing

Submit your work online & get it back in as little as 48 hours

百尺竿头，更进一步
Address us in arabic: [عزز فرصك](#)
Address us in Chinese: [艾文思](#)
Address us in Spanish: [Aumentar tus posibilidades](#)

GEORGE BOOLE* 200

CP 15 ^ ICLP 15

Cork, Ireland

Variants of jsp

- openness:
 - variety of resources can perform an operation
 - processing time dependant on resource used
- set up costs, between jobs (transition cost)
- consumable resources
 - such as gas, oil, etc
- pre-emption
 - can stop and restart an operation
- resource can perform multiple operations simultaneously
 - batch processing
- secondary resources
 - people, tools, cranes, etc
- etc

Chris Beck (2006) "The jssp has never been spotted in the wild."

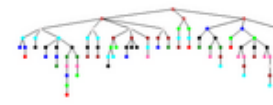
Why might CP be technology of choice for scheduling?

- can model rich real-world problems
 - addition of side constraints etc
- incorporate domain knowledge
 - in the form of variable and value ordering heuristics
- powerful reasoning/inference allied to novel search techniques

We can get a solution up and running quickly



Constraint Programming M



Home Page

News

Schedule

Contrib

Papers

Choco



Links

Exercises

MiniZinc

- choco4
 - [choco web site](#)
 - [our files](#)
 - [choco user guide](#)
 - [choco4 jar file](#)
 - [choco4 api](#)
 - [some of my files](#)
- First Simple Examples
 - [Start 2017](#)
- The Crystal Maze
 - We present [a simple encoding of the crystal maze problem](#)
- The N-queens
 - We present [the n-queens problem \(CP "Hello World!"\)](#),
- Jobshop scheduling.
 - An introduction to [jobshop scheduling](#)
 - Code etc to minimise makespan [jssp](#)
- Number Partitioning
 - [Code, data sets, and slides](#)
- Bin Packing
 - [jchoco code and data sets binPack](#)
 - Slides presenting 1D bin packing as [a case study](#)
- Graph Colouring
 - Three models for [graph colouring](#)

Index of /~pat/cpM/choco4/jssp

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 code/	26-Oct-2017 12:01	-	
 papers/	25-Oct-2017 11:30	-	
 problems/	25-Oct-2017 11:30	-	
 results-2015/	25-Oct-2017 11:30	-	
 results/	26-Oct-2017 12:01	-	
 slides/	25-Oct-2017 11:30	-	

Apache/2.2.3 (CentOS) Server at www.dcs.gla.ac.uk Port 80

Operation

```
Operation - Notepad
File Edit Format View Help
import org.chocosolver.solver.Model;
import org.chocosolver.solver.variables.IntVar;
import org.chocosolver.solver.constraints.Constraint;

public class operation {
    String id;
    String resId;
    int duration;
    IntVar start;
    Model model;

    operation(String id,String resId,int duration,int earliestStart,int latestStart,Model model){
        this.id = id;
        this.resId = resId; // resource id
        this.duration = duration;
        this.model = model;
        start = model.intVar(id,earliestStart,latestStart);
    }

    public String toString(){
        return "{" + id + " " + resId + " " + duration + " [" + start.getLB() + "," + start.getUB() + "]}";
    }

    Constraint before(operation op2){
        return model.arithm(op2.start,">=",start,"+",duration);
    }
    //
    // this operation is before operation op2
    // Therefore, if this operation starts at time t and has duration d
    // it finishes at time t+d. Therefore operation op2 can start immediately
    // at time t+d or any time after that
    //
}

```

```

File Edit Format View Help
import org.chocosolver.solver.Model;
import org.chocosolver.solver.variables.IntVar;
import org.chocosolver.solver.constraints.Constraint;

public class operation {
    String id;
    String resId;
    int duration;
    IntVar start;
    Model model;

    operation(String id,String resId,int duration,int earliestStart,int latestStart){
        this.id = id;
        this.resId = resId; // resource id
        this.duration = duration;
        this.model = model;
        start = model.intVar(id,earliestStart,latestStart);
    }

    public String toString(){
        return "{" + id + " " + resId + " " + duration + " [" + start.getLB() + "," + start.getUB() + "]}";
    }

    Constraint before(operation op2){
        return model.arithm(op2.start,">=",start,"+",duration);
    }
    //
    // this operation is before operation op2
    // Therefore, if this operation starts at time t and has duration d
    // it finishes at time t+d. Therefore operation op2 can start immediately
    // at time t+d or any time after that
    //
}

```

Operation has

- a duration
- a scheduled start time
- is on a resource

```
Operation - Notepad
File Edit Format View Help
import org.chocosolver.solver.Model;
import org.chocosolver.solver.variables.IntVar;
import org.chocosolver.solver.constraints.Constraint;

public class operation {
    String id;
    String resId;
    int duration;
    IntVar start;
    Model model;

    operation(String id,String resId,int duration,int earliestStart,int latestStart,Model model){
        this.id = id;
        this.resId = resId; // resource id
        this.duration = duration;
        this.model = model;
        start = model.intVar(id,earliestStart,latestStart);
    }

    public String toString(){
        return "{" + id + " " + resId + " " + duration + " [" + start.getLB() + "," + start.getUB() + "]}";
    }

    constraint before(operation op2){
        return model.arithm(op2.start,">=",start,"+",duration);
    }
    //
    // this operation is before operation op2
    // Therefore, if this operation starts at time t and has duration d
    // it finishes at time t+d. Therefore operation op2 can start immediately
    // at time t+d or any time after that
    //
}
}
```

```
Operation - Notepad
File Edit Format View Help
import org.chocosolver.solver.Model;
import org.chocosolver.solver.variables.IntVar;
import org.chocosolver.solver.constraints.Constraint;

public class operation {
    String id;
    String resId;
    int duration;
    IntVar start;
    Model model;

    operation(String id,String resId,int duration,int earliestStart,int latestStart,Model model){
        this.id = id;
        this.resId = resId; // resource id
        this.duration = duration;
        this.model = model;
        start = model.intVar(id,earliestStart,latestStart);
    }

    public String toString(){
        return "{" + id + " " + resId + " " + duration + " [" + start.getLB() + "," + start.getUB() + "]}";
    }

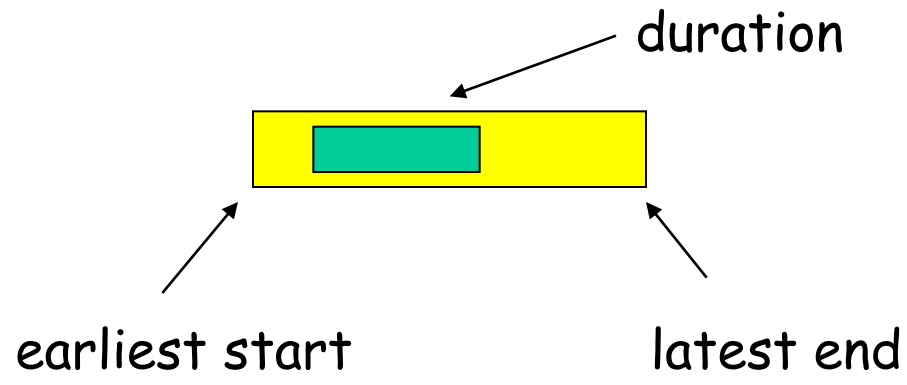
    Constraint before(operation op2){
        return model.arithm(op2.start,">=",start,"+",duration);
    }
    //
    // this operation is before operation op2
    // Therefore, if this operation starts at time t and has duration d
    // it finishes at time t+d. Therefore operation op2 can start immediately
    // at time t+d or any time after that
    //
}
}
```

```
Constraint before(Operation op2){  
    return model.arithm(op2.start,">=",start,"+",duration);  
}  
///  
///  
/// this operation is before operation op2  
/// Therefore, if this operation starts at time t and has duration d  
/// it finishes at time t+d. Therefore operation op2 can start immediately  
/// at time t+d or any time after that  
///  
///  
///
```

see next slides

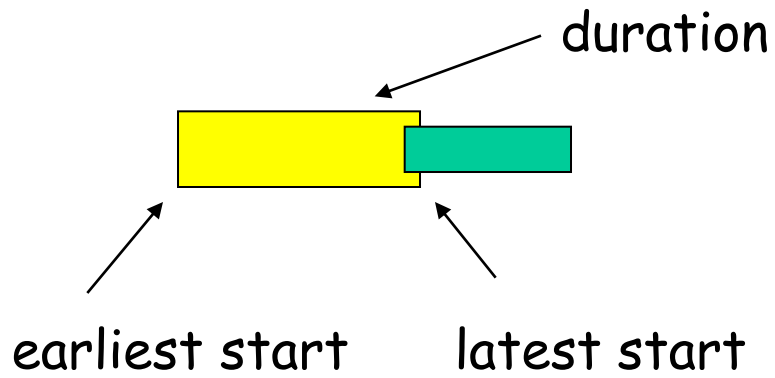
Picture of an operation

op1.before(op2)



Picture of an operation

`op1.before(op2)`



Constrained integer variable represents start time

Picture of an operation

op1.before(op2)



op1



op2

$$\text{op1.before(op2)} \longrightarrow \text{op1.start()} + \text{op1.duration()} \leq \text{op2.start()}$$

Picture of an operation

op1.before(op2)



op1

propagate



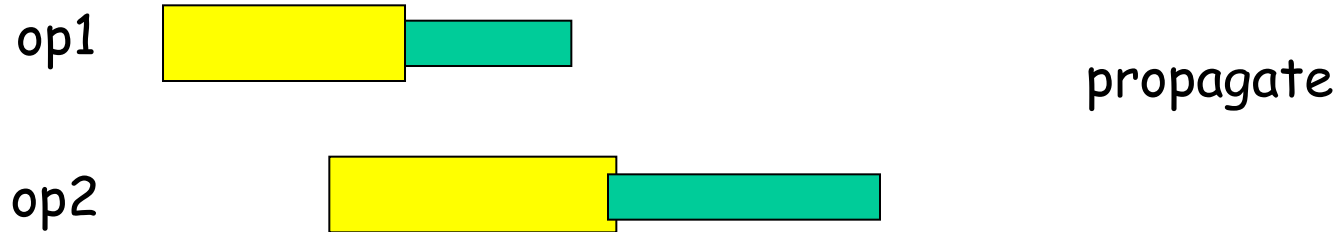
op2

$$\text{op1.before(op2)} \longrightarrow \text{op1.start()} + \text{op1.duration()} \leq \text{op2.start()}$$

Update earliest start of operation op2

Picture of an operation

op1.before(op2)



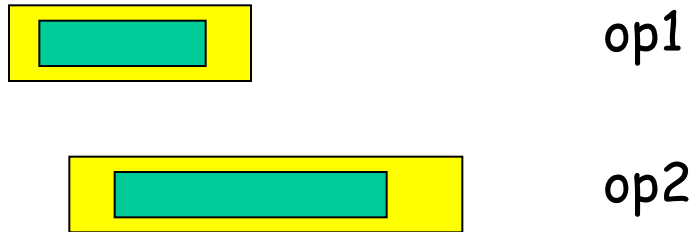
$$\text{op1.before(op2)} \longrightarrow \text{op1.start()} + \text{op1.duration()} \leq \text{op2.start()}$$

Update latest start of operation op1

No effect on this instance

Picture of an operation

op1.before(op2)



op1 and op2 cannot be in process at same time

→ op1.before(op2) **OR** op2.before(op1)

Not easy to propagate until
decision made (disjunction broken)


File Edit View History Bookmarks Tools Help

Index of /~pat/cpM/choco... x W Allen's interval algebra - W... x +

https://en.wikipedia.org/wiki/Allen's_interval_algebra Search

Create account Not logged in Talk Contributions Log in

Article Talk Read Edit View history Search



WIKIPEDIA
The Free Encyclopedia

- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia

Allen's interval algebra

From Wikipedia, the free encyclopedia

For the type of boolean algebra called interval algebra, see [Boolean algebra \(structure\)](#)

Allen's interval algebra is a [calculus](#) for [temporal reasoning](#) that was introduced by [James F. Allen](#) in 1983.

The calculus defines possible relations between time intervals and provides a composition table that can be used as a basis for reasoning about temporal descriptions of events.


File Edit View History Bookmarks Tools Help

Index of /~pat/cpM/choco... x W Allen's interval algebra - W... x +

https://en.wikipedia.org/wiki/Allen's_interval_... Search

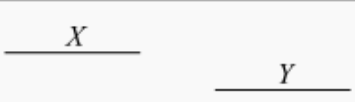
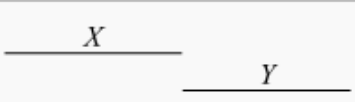
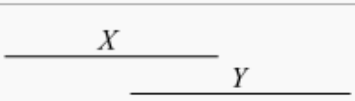
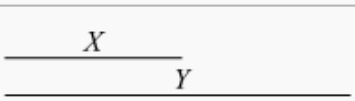
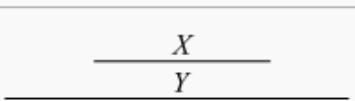
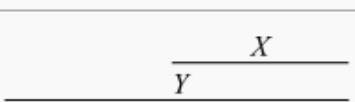
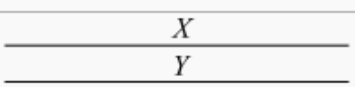
Page information
[Wikidata item](#)
[Cite this page](#)

Print/export
[Create a book](#)
[Download as PDF](#)
[Printable version](#)

Languages 
[Deutsch](#)
[Français](#)
[Edit links](#)

Relations [\[edit \]](#)

The following 13 base relations capture the possible relations between two intervals.

Relation	Illustration	Interpretation
$X < Y$ $Y > X$		X takes place before Y
$X m Y$ $Y mi X$		X meets Y (<i>i</i> stands for <i>inverse</i>)
$X o Y$ $Y oi X$		X overlaps with Y
$X s Y$ $Y si X$		X starts Y
$X d Y$ $Y di X$		X during Y
$X f Y$ $Y fi X$		X finishes Y
$X = Y$		X is equal to Y

Using this calculus, given facts can be formalized and then used for automatic reasoning. Relations between intervals are formalized as sets of base relations.

The sentence

During dinner, Peter reads the newspaper. Afterwards, he goes to bed.

Job

Job

```
Job - Notepad
File Edit Format View Help
import java.util.ArrayList;
import org.chocosolver.solver.Model;

public class Job {
    String id;
    ArrayList<operation> operations;
    int length;
    Model model;

    Job(String id, Model model){
        this.id = id;
        operations = new ArrayList<operation>();
        length = 0;
        this.model = model;
    }

    void add(operation op){
        if (!operations.isEmpty()) operations.get(length-1).before(op).post();
        operations.add(op);
        length++;
    }
    //
    // adding operations in sequence to a job, such that
    // operation op_i is before operation op_{i+1}
    // i.e. precedence constraints between operations in a job.
    //
    operation get(int i){return operations.get(i);}

    public string toString(){
        string s = "(" + id + " ";
        for (int i=0; i<operations.size(); i++) s = s + ((operation)operations.get(i)).toString() + " ";
        return s + ")";
    }
}
```



```
public class Job {  
    String id;  
    ArrayList<Operation> operations;  
    int length;  
    Model model;  
  
    Job(string id, Model model){  
        this.id = id;  
        operations = new ArrayList<Operation>();  
        length = 0;  
        this.model = model;  
    }  
}
```

Job is a sequence of operations

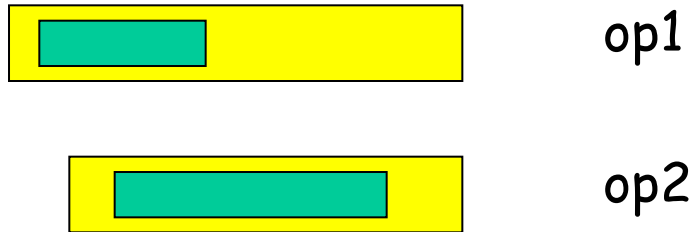
```
void add(Operation op){
    if (!operations.isEmpty()) operations.get(length-1).before(op).post();
    operations.add(op);
    length++;
}
///
/// adding operations in sequence to a job, such that
/// operation op_i is before operation op_{i+1}
/// i.e. precedence constraints between operations in a job.
///
```

Creating/building a job as a sequence of operations each one *before* the other

Decision

Picture of an operation

op1.before(op2)



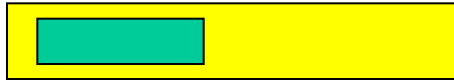
Use a 0/1 decision variable $d[i][j]$ as follows

$d[i][j] = 0 \rightarrow \text{op}[i].\text{before}(\text{op}[j])$

$d[i][j] = 1 \rightarrow \text{op}[j].\text{before}(\text{op}[i])$

Picture of an operation

op1.before(op2)



op1



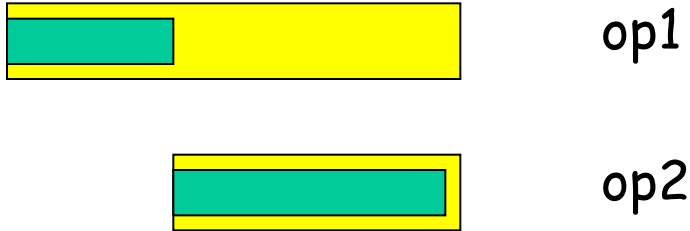
op2

$d[i][j] = 0 \rightarrow \text{op}[i].\text{before}(\text{op}[j])$

op1 before op2

Picture of an operation

op1.before(op2)

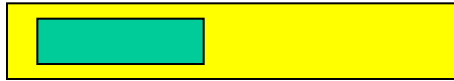


$d[i][j] = 0 \rightarrow \text{op}[i].\text{before}(\text{op}[j])$

op1 before op2

Picture of an operation

op1.before(op2)



op1



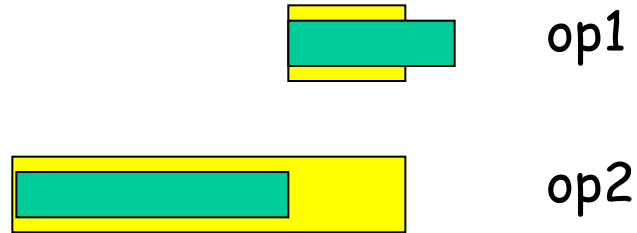
op2

$d[i][j] = 1 \rightarrow op[j].before(op[i])$

op2 before op1

Picture of an operation

op1.before(op2)



$d[i][j] = 1 \rightarrow op[j].before(op[i])$

op2 before op1

Decision

```
Decision - Notepad
File Edit Format View Help
import org.chocosolver.solver.variables.IntVar;

public class Decision {
    IntVar d;
    Operation op_i;
    Operation op_j;

    Decision(IntVar d, Operation op_i, Operation op_j){
        this.d = d;
        this.op_i = op_i;
        this.op_j = op_j;
    }

    public String toString(){return "{" + d + " " + op_i + " " + op_j + "}";}

    IntVar getD(){return d;}
    Operation getOp_i(){return op_i;}
    Operation getOp_j(){return op_j;}
}
///
/// A Decision is a triple where 0/1 variable d decides
/// the ordering between two operations on a resource
///
```

```
public class Decision {
    IntVar d;
    Operation op_i;
    Operation op_j;

    Decision(IntVar d, Operation op_i, Operation op_j){
        this.d = d;
        this.op_i = op_i;
        this.op_j = op_j;
    }
}
```

A decision is a triple:

- a zero/one variable d
- an operation op_i
- an operation op_j

Value of d decides relative order of
The two operations (before or after)

Resource

```
Resource - Notepad
File Edit Format View Help

import java.util.*;
import org.chocosolver.solver.Model;
import org.chocosolver.solver.variables.Intvar;

public class Resource {
    String id;
    ArrayList<Operation> operations;
    ArrayList<Decision> decisions;
    Model model;

    Resource(String id, Model model){
        this.id = id;
        operations = new ArrayList<Operation>();
        decisions = new ArrayList<Decision>();
        this.model = model;
    }

    void add(Operation op){
        int n = operations.size();
        for (int i=0; i<n; i++){
            Operation op_i = operations.get(i);
            Intvar decision = model.intVar("dec_" + i + "_", n, 0, 1);
            decisions.add(new Decision(decision, op_i, op));
            model.ifThen(model.arithm(decision, "=", 0), op_i.before(op)); // decision = 0 -> op_i before op
            model.ifThen(model.arithm(decision, "=", 1), op.before(op_i)); // decision = 1 -> op before op_i
        }
        operations.add(op);
    }

    public String toString(){return "Res: " + id +
        " NoOps: " + operations.size() +
        " NoDecVars: " + decisions.size();}
}
```

```
public class Resource {
    String id;
    ArrayList<Operation> operations;
    ArrayList<Decision> decisions;
    Model model;

    Resource(String id, Model model){
        this.id = id;
        operations = new ArrayList<Operation>();
        decisions = new ArrayList<Decision>();
        this.model = model;
    }
}
```

Resource is a collection of operations and decisions that will be made on their ordering/sequencing on this resource

```
void add(Operation op){
    int n = operations.size();
    for (int i=0;i<n;i++){
        Operation op_i = operations.get(i);
        IntVar decision = model.intvar("dec_"+ i +", "+ n,0,1);
        decisions.add(new Decision(decision,op_i,op));
        model.ifThen(model.arithm(decision,"=",0),op_i.before(op));
        model.ifThen(model.arithm(decision,"=",1),op.before(op_i));
    }
    operations.add(op);
}
```

Add an operation to a resource and then constrain it ...

```
void add(Operation op){
    int n = operations.size();
    for (int i=0;i<n;i++){
        Operation op_i = operations.get(i);
        IntVar decision = model.intvar("dec_"+ i +"," + n,0,1);
        decisions.add(new Decision(decision,op_i,op));
        model.ifThen(model.arithm(decision,"=",0),op_i.before(op));
        model.ifThen(model.arithm(decision,"=",1),op.before(op_i));
    }
    operations.add(op);
}
```

decision = 0 implies op_i before op
decision = 1 implies op before op_i

JSSP


```

JSSP - Notepad
File Edit Format View Help
import java.io.*;
import java.util.*;
import org.chocosolver.solver.Model;
import org.chocosolver.solver.Solver;
import org.chocosolver.solver.variables.IntVar;
import org.chocosolver.solver.constraints.IIntConstraintFactory.*;
import org.chocosolver.solver.search.strategy.Search;
import org.chocosolver.solver.exception.ContradictionException;

public class JSSP {
    String id; // file name
    int n; // number of jobs
    int m; // number of resources
    int dueDate; // aka makespan
    ArrayList<Job> jobs; // jobs to complete
    ArrayList<Resource> resources; // resources to use
    Operation endOp; // last operation for ALL jobs!
    Model model;

    JSSP(String fname,int dueDate) throws IOException {
        Scanner sc = new Scanner(new File(fname));
        id = fname;
        n = sc.nextInt(); // number of jobs
        m = sc.nextInt(); // number of resources
        jobs = new ArrayList<Job>();
        resources = new ArrayList<Resource>();
        this.dueDate = dueDate;
        model = new Model("id");
        endOp = new Operation("endOp","nullRes",0,0,dueDate,model);
        int totalDuration = 0;
        for (int i=0;i<m;i++){ resources.add(new Resource("r_"+i,model));}
        for (int i=0;i<n;i++){
            Job job = new Job("job_"+i,model);
            for (int j=0;j<m;j++){
                Resource resource = resources.get(sc.nextInt());
                int duration = sc.nextInt();
                totalDuration = totalDuration + duration;
                Operation operation = new Operation("op_"+i+"_"+j,resource.id,duration,0,dueDate,model);
                resource.add(operation);
                job.add(operation);
            }
            job.add(endOp);
            jobs.add(job);
        }
        model.arithm(endOp.start,"<=",totalDuration).post();
        sc.close();
    }

    public String toString(){
        String s = "JSSP " + n + "x" + m + "\n";
        for (int i=0;i<jobs.size();i++){
            Job job = (Job)jobs.get(i);
            s = s + job + "\n";
        }
        return s;
    }

    IntVar getMakeSpan(){return endOp.start;}

    Decision[] getDecisions(){
        Decision[] decisions = new Decision[((n * (n-1))/2) * m];
        for (int i=0,k=0;i<m;i++){
            for (Decision decision : resources.get(i).decisions)
                decisions[k++] = decision;
        }
        return decisions;
    }
    // to be used with min-slack dvo

    IntVar[] getDecisionIntVars(){
        IntVar[] decisions = new IntVar[((n * (n-1))/2) * m];
        for (int i=0,k=0;i<m;i++){
            for (Decision decision : resources.get(i).decisions)
                decisions[k++] = decision.getD();
        }
        return decisions;
    }
    // to be used with lex dvo
}

```

JSSP

```
JSSP - Notepad
File Edit Format View Help
import java.io.*;
import java.util.*;
import org.chocosolver.solver.Model;
import org.chocosolver.solver.Solver;
import org.chocosolver.solver.variables.IntVar;
import org.chocosolver.solver.constraints.IIntConstraintFactory.*;
import org.chocosolver.solver.search.strategy.Search;
import org.chocosolver.solver.exception.ContradictionException;

public class JSSP {
    String id; // file name
    int n; // number of jobs
    int m; // number of resources
    int dueDate; // aka makespan
    ArrayList<Job> jobs; // jobs to complete
    ArrayList<Resource> resources; // resources to use
    Operation endOp; // last operation for ALL jobs!
    Model model;

    JSSP(String fname,int dueDate) throws IOException {
        Scanner sc = new Scanner(new File(fname));
        id = fname;
        n = sc.nextInt(); // number of jobs
        m = sc.nextInt(); // number of resources
        jobs = new ArrayList<Job>();
        resources = new ArrayList<Resource>();
        this.dueDate = dueDate;
        model = new Model("id");
        endOp = new Operation("endOp","nullRes",0,0,dueDate,model);
        int totalDuration = 0;
        for (int i=0;i<m;i++){ resources.add(new Resource("r_"+i,model));}
        for (int i=0;i<n;i++){
            Job job = new Job("job_"+i,model);
            for (int j=0;j<m;j++){
                Resource resource = resources.get(sc.nextInt());
                int duration = sc.nextInt();
                totalDuration = totalDuration + duration;
                Operation operation = new Operation("op_"+i+"_"+j,resource.id,duration,0,dueDate,model);
                resource.add(operation);
                job.add(operation);
            }
            job.add(endOp);
            jobs.add(job);
        }
        model.arithm(endOp.start,"<=",totalDuration).post();
        sc.close();
    }

    public String toString(){
        String s = "JSSP "+ n + "x" + m + "\n";
        for (int i=0;i<jobs.size();i++){
            Job job = (Job)jobs.get(i);
            s = s + job + "\n";
        }
        return s;
    }

    IntVar getMakeSpan(){return endOp.start;}

    Decision[] getDecisions(){
        Decision[] decisions = new Decision[((n * (n-1))/2) * m];
        for (int i=0,k=0;i<m;i++){
            for (Decision decision : resources.get(i).decisions)
                decisions[k++] = decision;
        }
        return decisions;
    }
    // to be used with min-slack dvo

    IntVar[] getDecisionIntVars(){
        IntVar[] decisions = new IntVar[((n * (n-1))/2) * m];
        for (int i=0,k=0;i<m;i++){
            for (Decision decision : resources.get(i).decisions)
                decisions[k++] = decision.getD();
        }
        return decisions;
    }
    // to be used with lex dvo
}
```

Ouch!

```
public class JSSP {  
    String id;           // file name  
    int n;              // number of jobs  
    int m;              // number of resources  
    int dueDate;        // aka makespan  
    ArrayList<Job> jobs; // jobs to complete  
    ArrayList<Resource> resources; // resources to use  
    Operation endOp;    // last operation for ALL jobs!  
    Model model;  
}
```

A jssp is a collection of jobs and resources

```

JSSP(String fname,int dueDate) throws IOException {
    Scanner sc      = new Scanner(new File(fname));
    id              = fname;
    n               = sc.nextInt(); // number of jobs
    m               = sc.nextInt(); // number of resources
    jobs            = new ArrayList<Job>();
    resources       = new ArrayList<Resource>();
    this.dueDate    = dueDate;
    model           = new Model("id");
    endop           = new Operation("endop","nullRes",0,0,dueDate,model);
    int totalDuration = 0;
    for (int i=0;i<m;i++) resources.add(new Resource("r_"+i,model));
    for (int i=0;i<n;i++){
        Job job = new Job("job_"+i,model);
        for (int j=0;j<m;j++){
            Resource resource = resources.get(sc.nextInt());
            int duration      = sc.nextInt();
            totalDuration     = totalDuration + duration;
            Operation operation = new Operation("op_"+i+"_"+j,resource.id,duration,0,dueDate,model);
            resource.add(operation);
            job.add(operation);
        }
        job.add(endop);
        jobs.add(job);
    }
    model.arithm(endop.start,"<=",totalDuration).post();
    sc.close();
}

```

```
JSSP(String fname,int dueDate) throws IOException {
    Scanner sc      = new Scanner(new File(fname));
    id              = fname;
    n               = sc.nextInt(); // number of jobs
    m               = sc.nextInt(); // number of resources
    jobs            = new ArrayList<Job>();
    resources       = new ArrayList<Resource>();
    this.dueDate    = dueDate;
    model           = new Model("id");
    endOp           = new Operation("endOp","nullRes",0,0,dueDate,model);
}
```

```
JSSP(String fname,int dueDate) throws IOException {
    Scanner sc      = new Scanner(new File(fname));
    id              = fname;
    n               = sc.nextInt(); // number of jobs
    m               = sc.nextInt(); // number of resources
    jobs            = new ArrayList<Job>();
    resources       = new ArrayList<Resource>();
    this.dueDate    = dueDate;
    model           = new Model("JSSP");
    endOp           = new Operation("endOp","nullRes",0,0,dueDate,model);
}
```

```
int totalDuration = 0;
for (int i=0;i<m;i++) resources.add(new Resource("r_"+i,model));
for (int i=0;i<n;i++){
    Job job = new Job("job_"+i,model);
    for (int j=0;j<m;j++){
        Resource resource = resources.get(sc.nextInt());
        int duration      = sc.nextInt();
        totalDuration     = totalDuration + duration;
        Operation operation = new Operation("op_"+i+"_"+j,resource.id,duration,0,dueDate,model);
        resource.add(operation);
        job.add(operation);
    }
    job.add(endOp);
    jobs.add(job);
}
model.arithm(endOp.start,"<=",totalDuration).post();
sc.close();
```

```
int totalDuration = 0;
for (int i=0; i<m; i++) resources.add(new Resource("r_"+i,model));
for (int i=0; i<n; i++){
    Job job = new Job("job_"+i,model);
    for (int j=0; j<m; j++){
        Resource resource = resources.get(sc.nextInt());
        int duration = sc.nextInt();
        totalDuration = totalDuration + duration;
        Operation operation = new Operation("op_"+i+"_"+j,resource.id,duration,0,dueDate,model);
        resource.add(operation);
        job.add(operation);
    }
    job.add(endOp);
    jobs.add(job);
}
model.arithm(endOp.start,"<=",totalDuration).post();
sc.close();
```



```
int totalDuration = 0;
for (int i=0;i<m;i++) resources.add(new Resource("r_"+i,model));
for (int i=0;i<n;i++){
job job = new Job("job_"+i,model);
    for (int j=0;j<m;j++){
        Resource resource = resources.get(sc.nextInt());
        int duration      = sc.nextInt();
        totalDuration     = totalDuration + duration;
        Operation operation = new Operation("op_"+i+"_"+j,resource.id,duration,0,dueDate,model);
        resource.add(operation);
        job.add(operation);
    }
    job.add(endOp);
    jobs.add(job);
}
model.arithm(endOp.start,"<=",totalDuration).post();
sc.close();
```

```
int totalDuration = 0;
for (int i=0;i<m;i++) resources.add(new Resource("r_"+i,model));
for (int i=0;i<n;i++){
    Job job = new Job("job_"+i,model);
    for (int j=0;j<m;j++){
        Resource resource = resources.get(sc.nextInt());
        int duration      = sc.nextInt();
        totalDuration     = totalDuration + duration;
        Operation operation = new Operation("op_"+i+"_"+j,resource.id,duration,0,dueDate,model);
        resource.add(operation);
        job.add(operation);
    }
    job.add(endOp);
    jobs.add(job);
}
model.arithm(endOp.start,"<=",totalDuration).post();
sc.close();
```

```
int totalDuration = 0;
for (int i=0; i<m; i++) resources.add(new Resource("r_"+i, model));
for (int i=0; i<n; i++){
    Job job = new Job("job_"+i, model);
    for (int j=0; j<m; j++){
        Resource resource = resources.get(sc.nextInt());
        int duration = sc.nextInt();
        totalDuration = totalDuration + duration;
        Operation operation = new Operation("op_"+i+"_"+j, resource.id, duration, 0, dueDate, model);
        resource.add(operation);
        job.add(operation);
    }
    job.add(endOp);
    jobs.add(job);
}
model.arithm(endOp.start, "<=", totalDuration).post();
sc.close();
```

DecisionProblem

DecisionProblem

```
public class DecisionProblem {
    public static void main(String[] args) throws FileNotFoundException, IOException {
        int dueDate          = Integer.parseInt(args[1]);
        JSSP jssp             = new JSSP(args[0], dueDate);
        Solver solver         = jssp.getModel().getSolver();
        Decision[] decisions = jssp.getDecisions();
        int n                 = decisions.length;
        IntVar makespan       = jssp.getMakespan();

        solver.setSearch(Search.inputOrderUBSearch(jssp.getDecisionIntVars()));

        System.out.println("solved: " + solver.solve());

        System.out.println(makespan + " ["+ makespan.getLB() + ", "+ makespan.getUB() + "]");
        System.out.println("nodes: " + solver.getMeasures().getNodeCount() +
            " cpu: " + solver.getMeasures().getTimeCount());
    }
}
// is there a legal schedule with a make span of dueDate, or less?
//
```

```

public class Optimize {
    public static void main(String[] args) throws FileNotFoundException, IOException {
        int timeLimit      = Integer.parseInt(args[1]);
        JSSP jssp          = new JSSP(args[0],9999);
        Model model        = jssp.getModel();
        Solver solver      = model.getSolver();
        Decision[] decisions = jssp.getDecisions();
        int n              = decisions.length;
        IntVar makespan    = jssp.getMakespan();

        solver.limitTime(timeLimit*1000);

        solver.setSearch(new IntStrategy(jssp.getDecisionIntVars(),new MinSlackHeuristic(jssp.getDecisions()),new IntDomainMax()));
        //
        // attach a variable & value ordering heuristic to solver
        //
        // alternative way to optimize
        // solution solution = solver.findOptimalSolution(makespan,false);

        // trace optimisation
        model.setObjective(Model.MINIMIZE,makespan);
        while(solver.solve()) System.out.println("makespan: ["+ makespan.getLB() +"," + makespan.getUB() +"]" +
            "nodes: "+ solver.getMeasures().getNodeCount());

        System.out.println("nodes: "+ solver.getMeasures().getNodeCount() +
            "cpu: "+ solver.getMeasures().getTimeCount());
    }
}

```

Wot!? No heuristics!?!?



Maintaining Singleton Arc Consistency

Patrick Prosser¹ and Christophe Lecoutre²

<p>¹Department of Computing Science University of Glasgow Scotland pat@cs.gla.ac.uk</p>	<p>²CRIL-CNRS FRE 2499, Université d'Artois Lens, France lecoutre@cril.univ-artois.fr</p>
--	--

Abstract. Singleton Arc-Consistency (SAC) is a simple and strong level of consistency, but is expensive to enforce. To date, research has focused on improving the performance of algorithms that achieve SAC. These algorithms tend to be somewhat complex and non-trivial to implement. Furthermore, these algorithms have mostly been tested out as a preprocessing step before actually solving a problem. Here, we show how a basic and simple SAC algorithm can be incorporated into a constraint programming toolkit and then used within search. We then propose limited degrees of SAC, in particular SAC on the upper and lower bounds of variables (Bound-SAC) and SAC on only the first value in domains (First-SAC). We investigate the effects of limiting the amount of SAC maintained whilst solving a problem instance, i.e. by limiting the values in domains that are made SAC and the variables that are made SAC. Our studies show that judicious use of SAC within the search process can indeed pay off.

7

Instance	MAC	Maintaining		
		B-SAC _{dn}	B-SAC _{st}	B-SAC
la01	666	666	666	666
la02	655	655	655	655
la03	653	597	603	603
la04	628	598	590	590
la05	593	665	665	665
la06	1245	1146	1233	1237
la07	1214	897	1336	1359
la08	1161	1084	1400	1393
la09	1498	1049	1527	1520
la10	1658	972	1192	1259
la11	1453	1787	—	—
la12	1467	1504	—	—
la13	2899	2310	—	—
la14	1970	1784	—	—
la15	2368	2200	—	—

Table 1. Cost of best solution found for Lawrence job-shop scheduling instances, given 10 minutes CPU

5 A study of Golomb rulers

In [16], experiments were performed on Golomb rulers. In particular, given the length l