Task scheduling for dual-arm industrial robots through constraint programming

MiniZinc modeling and solver comparison

Tommy Kvant

Institute of Computer Science
Lund University

February 22, 2015

Task scheduling for dual-arm industrial robots through constraint programming
MiniZinc modeling and solver comparison
Tommy Kvant
Institute of Computer Science
Lund University
February 22, 2015

Outline

Introduction - YuMi®

- Dual-armed robot
- Flexible - multiple tools
- Fine motor skills

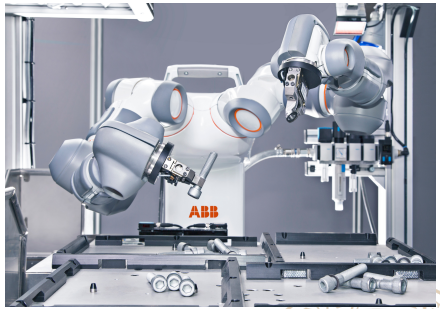
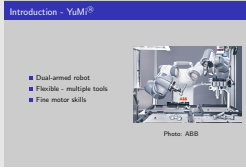


Photo: ABB

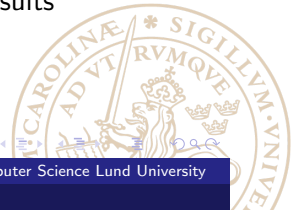


Mycket av produktion i dagens samhälle automatiseras & många produkter serier har kort livsspann → produktionen av nya serier behöver anpassas relativt snabbt → YuMi

1. Montera endast mellan maskiner utan tray

Introduction - Project goal

- Constraint Programming model for dual-armed robots such as YuMi®
  - Change tools
  - Carry only one component at the time
  - Same duration for tool changes, regardless of direction
  - Use trays, fixtures and outputs
- Implement the model in MiniZinc
- Test the model with 6 solvers and compare the results



Introduction - Project goal
Constraint Programming model for dual-armed robots such as YuMi®
Change tools
Carry only one component at the time
Same duration for tool changes, regardless of direction
Use trays, fixtures and outputs
Implement the model in MiniZinc
Test the model with 6 solvers and compare the results

Introduction - MiniZinc

- Declarative language
- Medium level
- Translates to FlatZinc
- Aims to be standard
- Many solvers can read FlatZinc

Introduction - MiniZinc
Declarative language
Medium level
Translates to FlatZinc
Aims to be standard
Many solvers can read FlatZinc

- Declarative language + declarative paradigm = good fit
- Many CP languages available, based on imperative languages = messy syntax
- Medium level: Högt nog för att formulera komplicerade uttryck, tex. for-loopar, lågt nog för att läät kunna översättas till FlatZink
- FlatZink låg nivå → läses lätt in av solvers
- Siktar på att bli standard, lätt att läsa in FlatZinc → relativt många solvers kan läsa in modeller

Case Study

Case Study

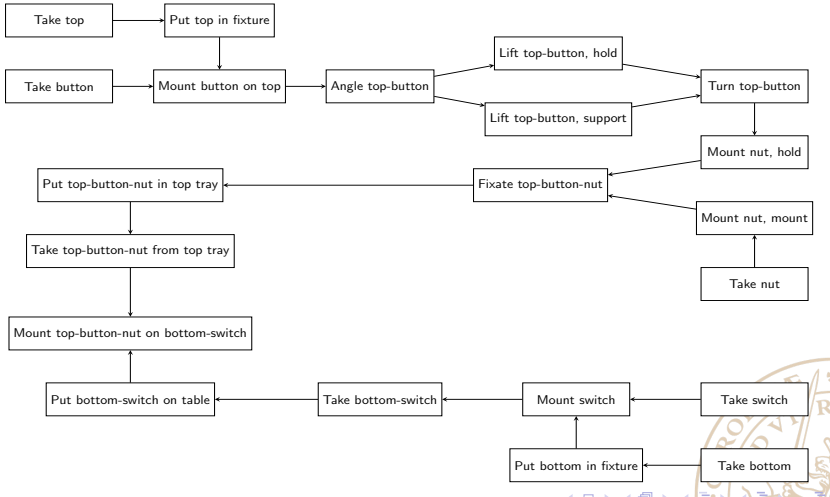
Skruvarna inte med

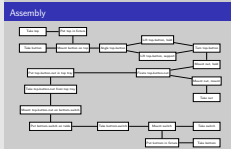
Case Study

Physical Entities

- Machines
- Tools
- Components
- Tray
- Fixture
- Output

Assembly


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Put top-button-nut in top tray
Fixate top-button-nut
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom

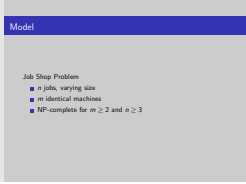

Assembly

Model

Job Shop Problem

- n jobs, varying size
- m identical machines
- NP-complete for $m \geq 2$ and $n \geq 3$

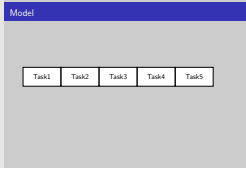
- Vill schemalägg tasks som ett job shop problem
- I literatur jobs innehåller operations, här tittar vi på 1 job och operations kallar vi tasks
- Varje jobb kan hanteras av vilken maskin som helst → Flexible Job Shop Scheduling



Model

Task1	Task2	Task3	Task4	Task5

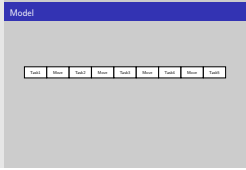
- En task kommer efter den andra
- Längden på en task, *duration*, tillhandahålls av den som skapar assemblyn



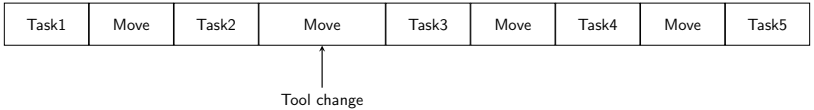
Model

Task1	Move	Task2	Move	Task3	Move	Task4	Move	Task5

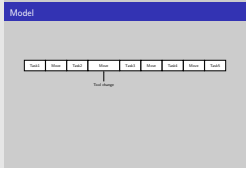
- Men tasks:en sker på olika ställen i rummet → det tar tid att flytta sig mellan dem → måste räkna med det
- Tider för move mellan tasks tillhandahålls genom en tidsmatris av den som vill schemalägga



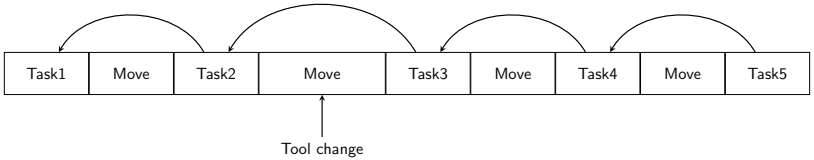
Model



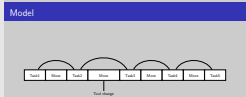
- Tasks behöver olika tools → måste utföra tool change
- utförs mellan två tasks → tiden att röra sig mellan två tasks tar längre tid → bakar in tool change tiden i move
- Det förekommer ett tool change om tiden för move tar längre tid än det egentligen skulle göra



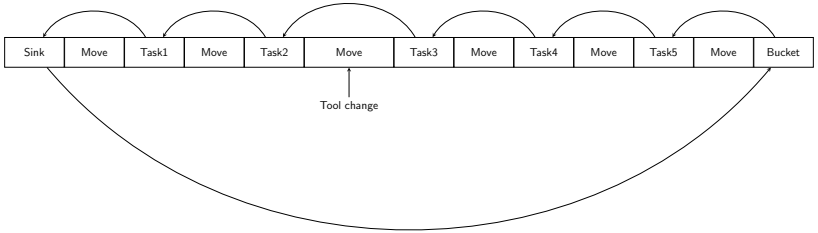
Model



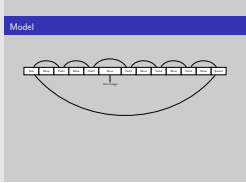
- Hur lång tid det tar beror på vilken task som kommer innan → vi måste veta vilken task som kommer innan, *predecessor*
- Tidsmatrisen tillsammans med tiderna för att byta mellan tools = ny matris med alla möjliga moves inkl. tool change
- Detta = Job Shop Problem with sequence-dependent setup times
- För att se till att detta uppfylls kan constraintet `circuit` användas
- Skapar en Hamiltonian circuit
- Uppnår det genom att koppla ihop första och sista noden.
- Constraint som säger att task måste komma efter sin predecessor → Första och sista task:en kan inte kopplas ihop



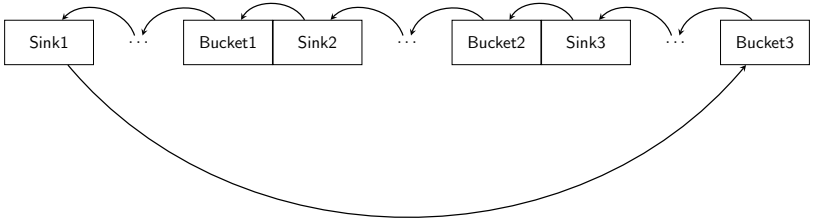
Model



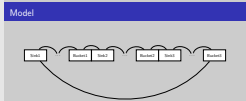
- Introducerar sink node/startTask & bucket node/goalTask
- Men detta måste göras för varje maskin, tasks måste fördelas $\rightarrow$ *usingMachine*$(t) \rightarrow$ task och predecessor måste vara på samma maskin



Model



- För att göra det för flear maskiner: Istället för lika många circuits som maskiner, kopplar ihop alla curcuits till en lång circuit
- Vi har alltså 4 variabler att schemalägga: tasks, moves, predecessors & usingMachine



Model

Components

- Primitive components
- Sub-assemblies

Model
Components
Primitive components
Sub-assemblies

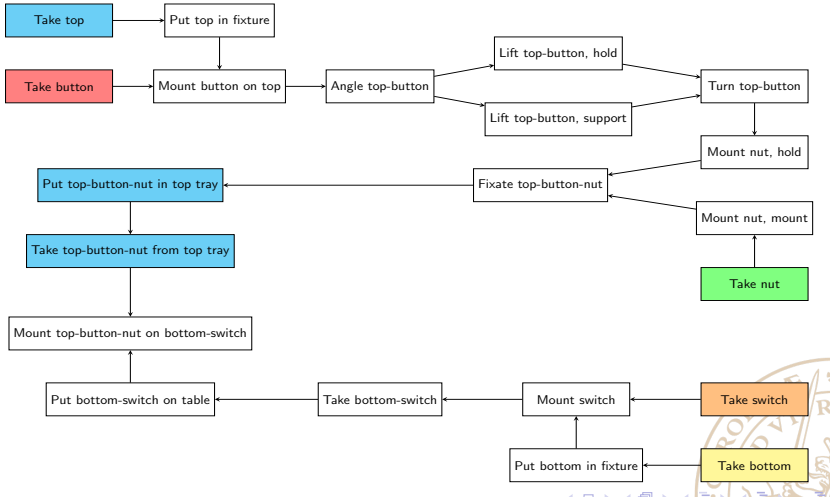
Model

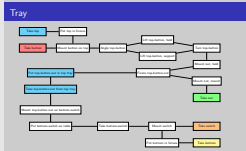
Storage mediums

- Tray - Top tray, Button tray, etc.
- Fixture
- Output

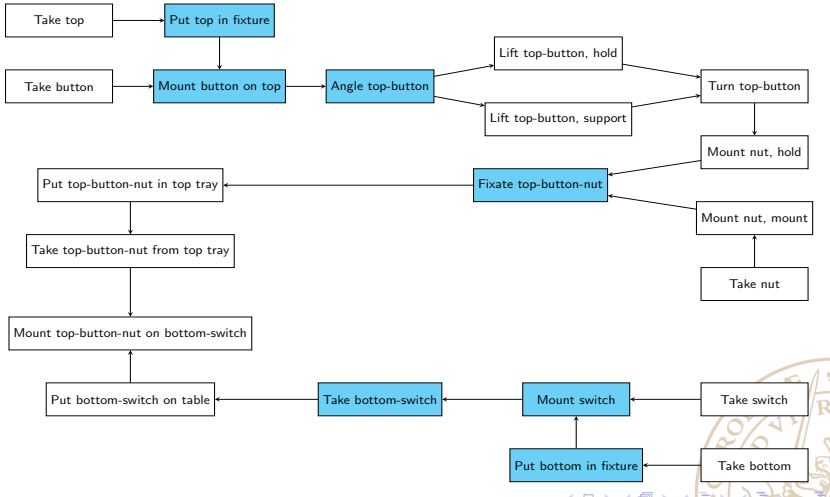
Model
Storage mediums
Tray - Top tray, Button tray, etc.
Fixture
Output

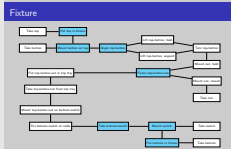
Tray


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Put top-button-nut in top tray
Fixate top-button-nut
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom


Tray

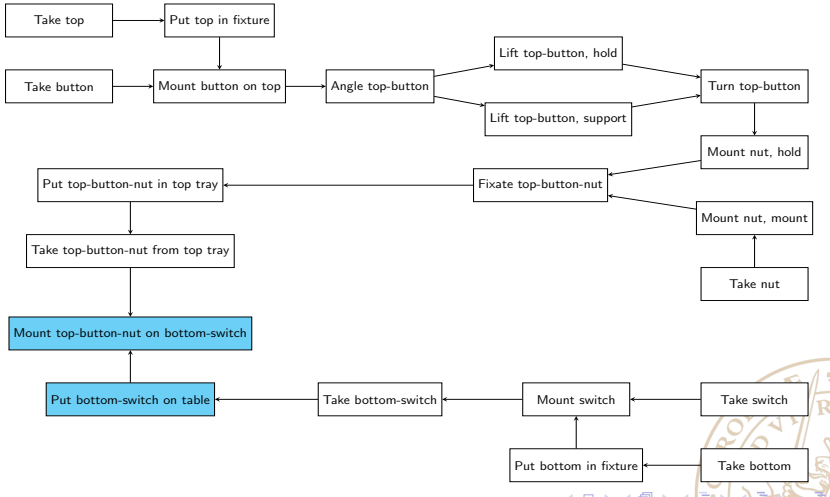
Fixture

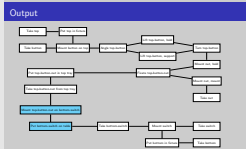




- Individuella tasks får inte överlappa på fixtures
- Tiden då fixtures är upptagna får inte överlappa, identifiera put och take för en komponent och komponent som har put komponenten som en del i dess sub-assembly

Output


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Put top-button-nut in top tray
Fixate top-button-nut
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom


Output

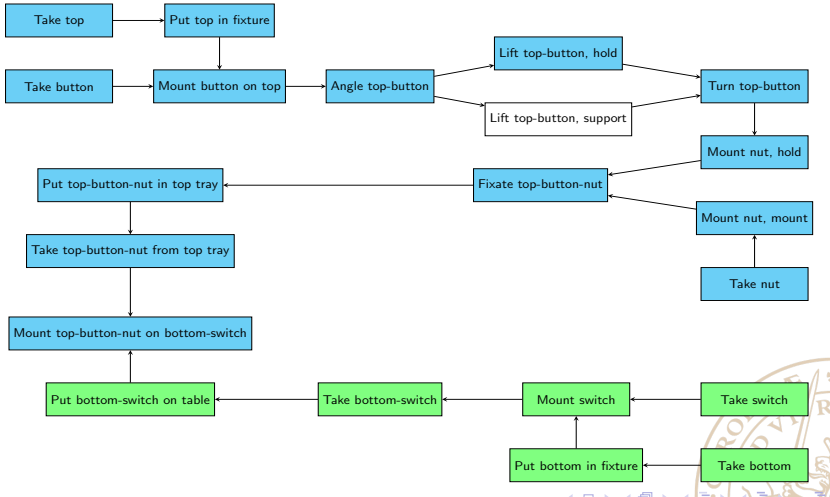
Tools

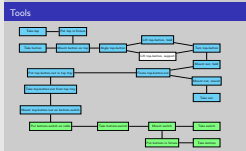
- Tools available
- Tool used by tasks

Tools
Tools available
Tool used by tasks

Antar att det finns en uppsättning av sagda tools för varje maskin

Tools





- Notera att "Lift top-button" inte har tool specificerad

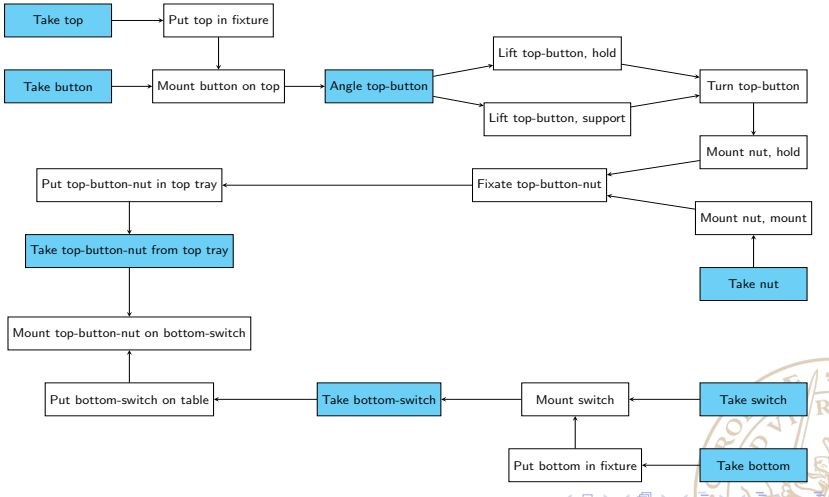
Model

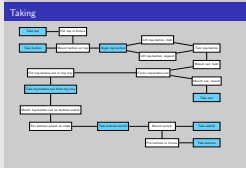
Label tasks

- Taking
- Mounting
- Putting
- Moving

Model
Label tasks
Taking
Mounting
Putting
Moving

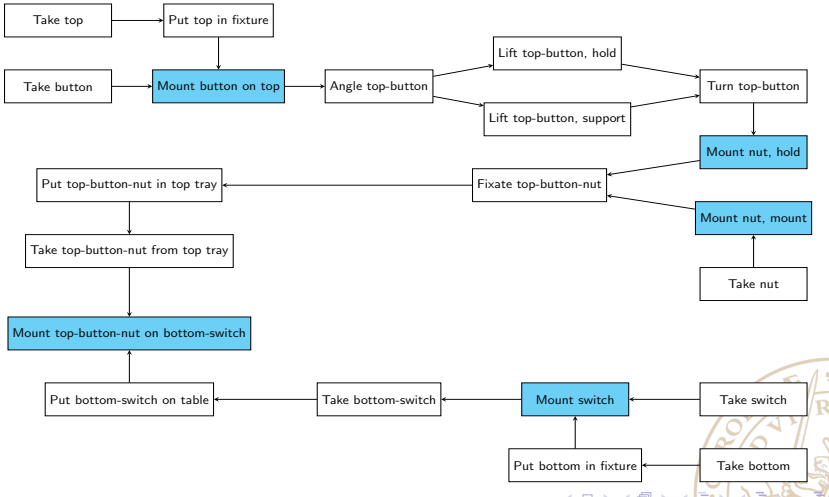
Taking

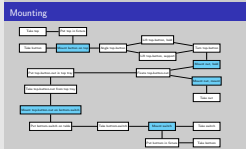




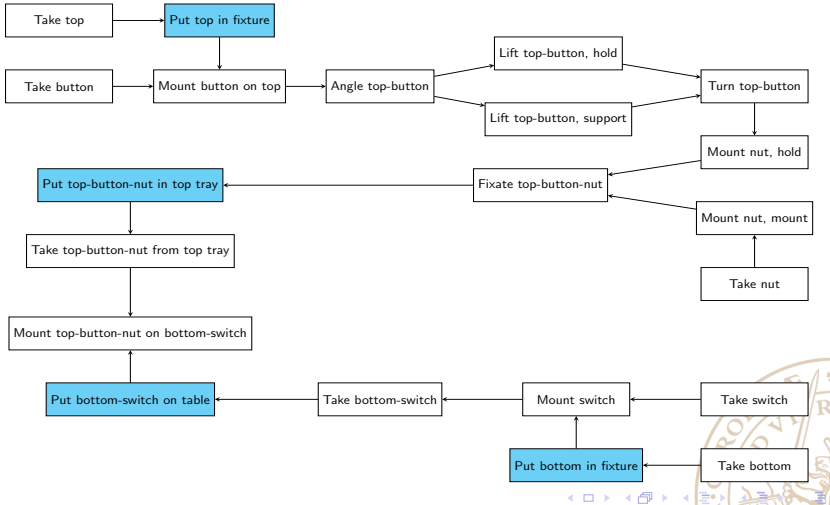
Angle skulle kunna delas in i två tasks, en taking och en moving

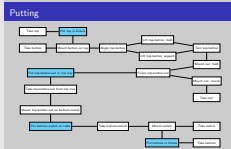
Mounting


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Put top-button-nut in top tray
Fixate top-button-nut
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom

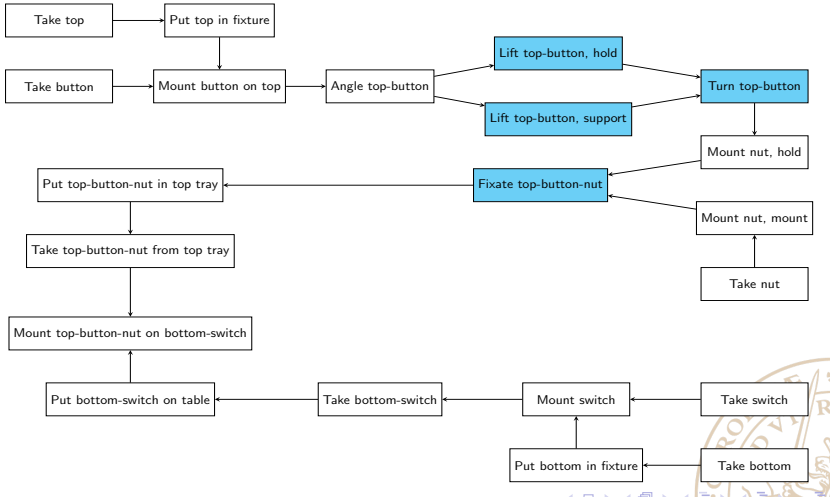

Mounting

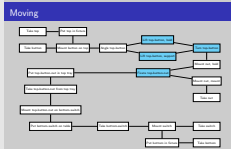
Putting


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Put top-button-nut in top tray
Fixate top-button-nut
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom


Putting

Moving


Take top
Put top in fixture
Take button
Mount button on top
Angle top-button
Lift top-button, hold
Lift top-button, support
Turn top-button
Mount nut, hold
Fixate top-button-nut
Put top-button-nut in top tray
Mount nut, mount
Take top-button-nut from top tray
Take nut
Mount top-button-nut on bottom-switch
Put bottom-switch on table
Take bottom-switch
Mount switch
Take switch
Put bottom in fixture
Take bottom


Moving

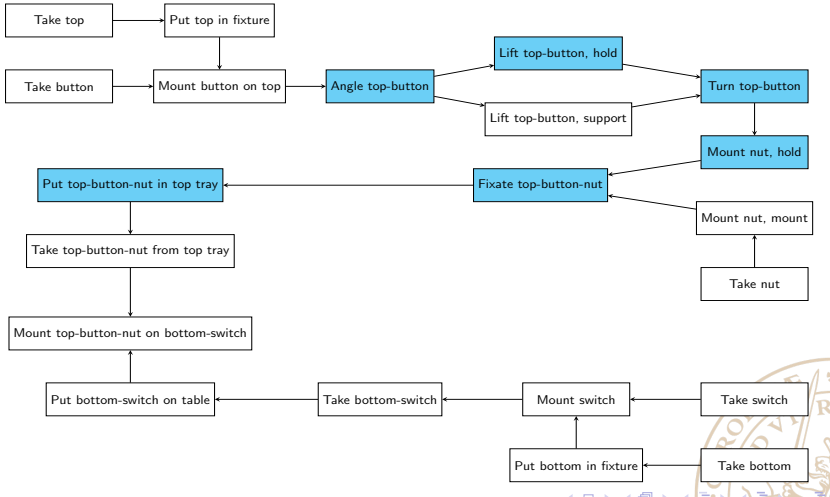
Model

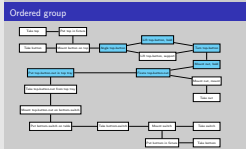
Group tasks

- Ordered group
- Concurrent group

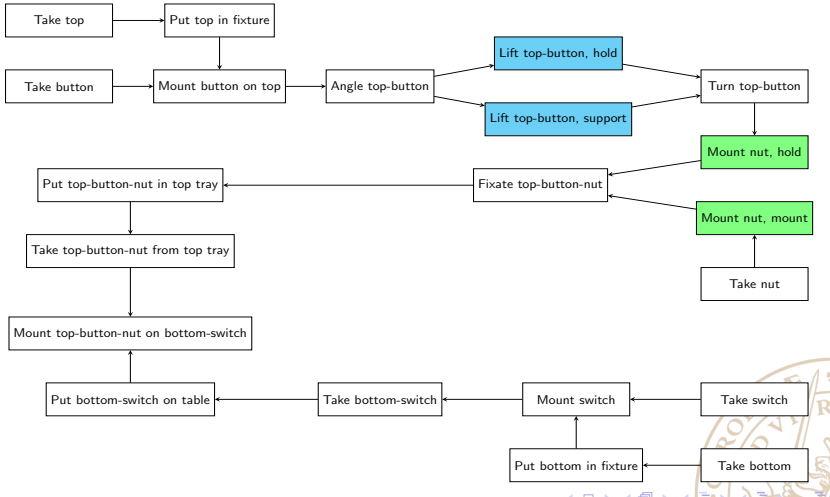
Model
Group tasks
Ordered group
Concurrent group

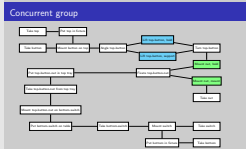
Ordered group





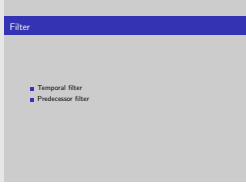
Concurrent group





Filter

- Temporal filter
- Predecessor filter



Temporal filter:

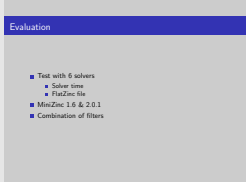
- Matris med alla möjliga moves → vi kan räkna ut värsta och bästa fallet för hela assemblyn
- mha. detta kan vi begränsa startTime för tasks

Predecessor filter:

- Vi vet att put och mount tasks inte kan komma först, då komponenten måste plockas upp först
  → $pred(putTask/mountTask) \neq startTask$
- Då allting måste sitta i outputs i slutet av assemblyn
  → $pred(goalTask) \neq takeTask$
- Tasks som använder components som är sub-components i en annan task måste ske innan den tasken → inte ha den som predecessor

Evaluation

- Test with 6 solvers
  - Solver time
  - FlatZinc file
- MiniZinc 1.6 & 2.0.1
- Combination of filters



FlatZinc file:

- Antal variabler
- Antal constraints
- Andel av constraints som är reified constraints

Reified:

- Kopplar ihop ett constraint med en variable("sanningsvärde") → är variabeln sann gäller constraint:et, är varibeln falsk gäller inte constraint:et, och vise versa
- Används ofta för implikationer och ekvivalenser
- Kan göra söktiden längre
- Aktivt unvikit direkta användningar

Criteria

- FlatZinc parser
- Free

Criteria
FlatZinc parser
Free

Solvers Tested

- G12/FD
- JaCoP
- Gecode
- or-tools
- Opturion CPX
- Choco3

Solvers Tested
G12/FD
JaCoP
Gecode
or-tools
Opturion CPX
Choco3

G12/FD

- G12 Team, NICTA
- Mercury
- Default solver for MiniZinc

G12/FD
G12 Team, NICTA
Mercury
Default solver for MiniZinc

- NICTA: National ICT Australia, Australia's Information Communications Technology (ICT) Research Centre, störst

JaCoP

- Java Constraint Programming solver
- Open Source
- Developed since 2001 - Krzysztof Kuchcinski & Radoslaw Szymanek
- Silver medal

JaCoP
- Java Constraint Programming solver
- Open Source
- Developed since 2001 - Krzysztof Kuchcinski & Radoslaw Szymanek
- Silver medal

Gecode

- C++
- Open Source
- Christian Schulte
- Parallel searches - utilising multiple cores
- All gold medals 2008-2012

Gecode
C++
Open Source
Christian Schulte
Parallel searches - utilising multiple cores
All gold medals 2008-2012

1. Christian Schulte: lett utvecklingen, många andra som bidragit
2. All gold medals 2008-2012: i alla kategorier

or-tools

- C++
- Google - Operational Research
- Open Source
- Utilising multiple cores
- Gold medals 2013-2014

1. Utilising multiple cores: Inte säker om parallell sökning, nämns i dokumentationen som "parrallel sovling", explicit utesluten ur dokumentationen

or-tools
C++
Google - Operational Research
Open Source
Utilising multiple cores
Gold medals 2013-2014

Opturion CPX

- Opturion Pty Ltd
- Commercial
- SAT combo
- Gold medals 2013, all silver medals 2014

Opturion CPX
Opturion Pty Ltd
Commercial
SAT combo
Gold medals 2013, all silver medals 2014

1. Opturion Pty Ltd: Härstammar från G12
2. Commercial: kostar, akademisk licens
3. SAT combo: FD + SAT, SAT = satslogik, väldigt effektiv på att lösa stora problem, sägs att satslogik $\rightarrow$ sökning inte slöas ner av stora domäner

Choco3

- Java
- Open Source
- Developed since early 2000 - Jean-Guillaume Fages & Charles Prud'homme
- Not same as predecessor Choco2

Choco3
Java
Open Source
Developed since early 2000 - Jean-Guillaume Fages & Charles Prud'homme
Not same as predecessor Choco2

Assembly Times

Manual Time
516 t.u.

Assembly Times
Manual Time
516 t.u.

Assembly Times

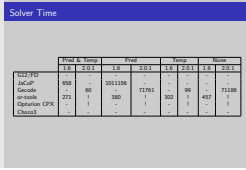
Manual Time
516 t.u.

Solver Time
512 t.u.

Assembly Times
Manual Time 516 t.u.
Solver Time 512 t.u.

Solver Time

	Pred & Temp		Pred		Temp		None	
	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1
G12/FD	-	-	-	-	-	-	-	-
JaCoP	658	-	1011156	-	-	-	-	-
Gecode	-	60	-	71761	-	99	-	71186
or-tools	271	!	380	!	302	!	457	!
Opturion CPX	-	!	-	!	-	!	-	!
Choco3	-	-	-	-	-	-	-	-

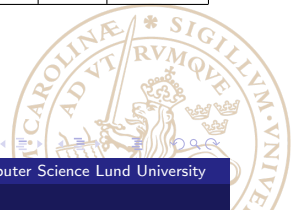


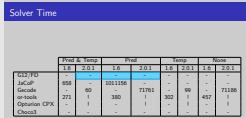
- 1011156 - 0:16:51
- 71761 - 0:01:11
- 71186 - 0:01:11

Vi kan se att i nästan alla fall hjälper filtrena i någon grad
Vilket filter bäst är svårt att säga, temp verkar bäst i flesta fall

Solver Time

	Pred & Temp		Pred		Temp		None	
	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1
G12/FD	-	-	-	-	-	-	-	-
JaCoP	658	-	1011156	-	-	-	-	-
Gecode	-	60	-	71761	-	99	-	71186
or-tools	271	!	380	!	302	!	457	!
Opturion CPX	-	!	-	!	-	!	-	!
Choco3	-	-	-	-	-	-	-	-

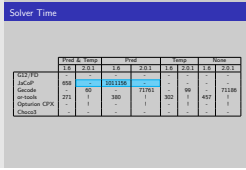




Hittar lösning, inte optimal

Solver Time

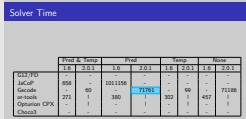
	Pred & Temp		Pred		Temp		None	
	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1
G12/FD	-	-	-	-	-	-	-	-
JaCoP	658	-	1011156	-	-	-	-	-
Gecode	-	60	-	71761	-	99	-	71186
or-tools	271	!	380	!	302	!	457	!
Opturion CPX	-	!	-	!	-	!	-	!
Choco3	-	-	-	-	-	-	-	-



Hittar hittar alla lösningar, inklusive optimala

Solver Time

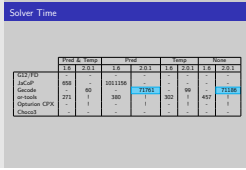
	Pred & Temp		Pred		Temp		None	
	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1
G12/FD	-	-	-	-	-	-	-	-
JaCoP	658	-	1011156	-	-	-	-	-
Gecode	-	60	-	71761	-	99	-	71186
or-tools	271	!	380	!	302	!	457	!
Opturion CPX	-	!	-	!	-	!	-	!
Choco3	-	-	-	-	-	-	-	-



Hittar 3 lösningar, inklusive optimala

Solver Time

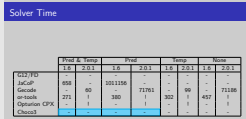
	Pred & Temp		Pred		Temp		None	
	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1
G12/FD	-	-	-	-	-	-	-	-
JaCoP	658	-	1011156	-	-	-	-	-
Gecode	-	60	-	71761	-	99	-	71186
or-tools	271	!	380	!	302	!	457	!
Opturion CPX	-	!	-	!	-	!	-	!
Choco3	-	-	-	-	-	-	-	-



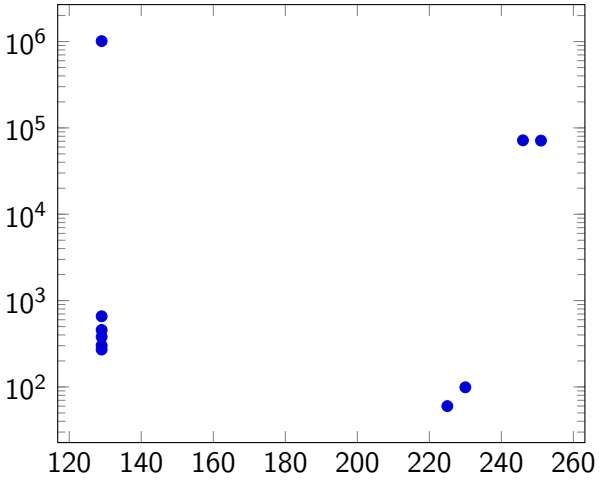
Hittar 1 lösning, den optimala, på ungefär samma tid som den tidigare

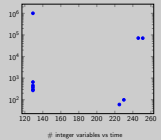
Solver Time

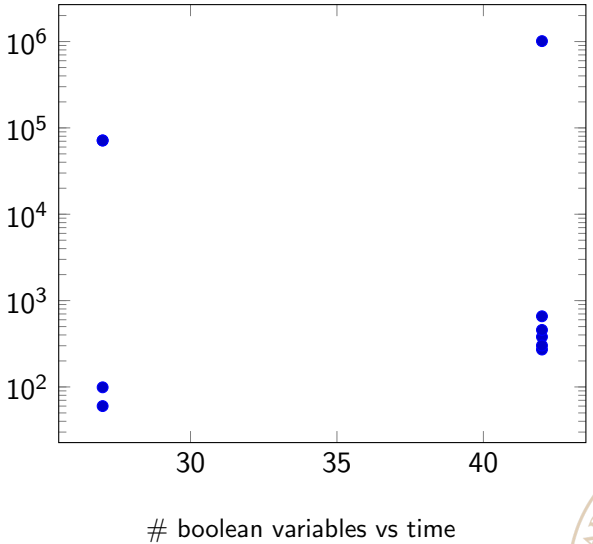
	Pred & Temp		Pred		Temp		None	
	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1	1.6	2.0.1
G12/FD	-	-	-	-	-	-	-	-
JaCoP	658	-	1011156	-	-	-	-	-
Gecode	-	60	-	71761	-	99	-	71186
or-tools	271	!	380	!	302	!	457	!
Opturion CPX	-	!	-	!	-	!	-	!
Choco3	-	-	-	-	-	-	-	-

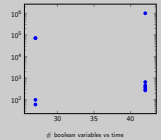


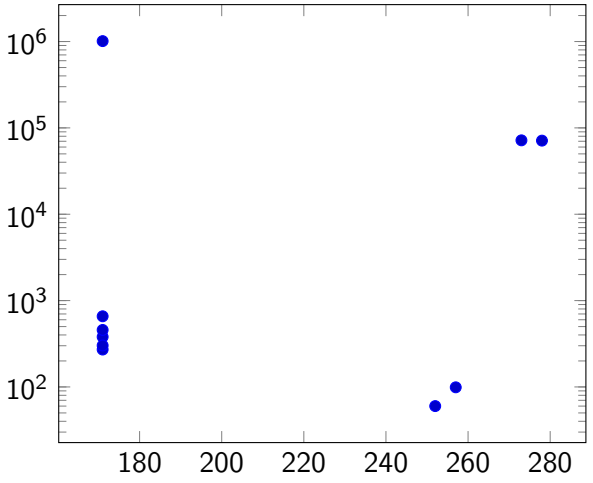
Hittar 2 lösningar direkt


10^6
10^5
10^4
10^3
10^2
120 140 160 180 200 220 240 260
integer variables vs time

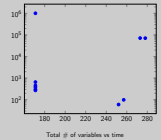


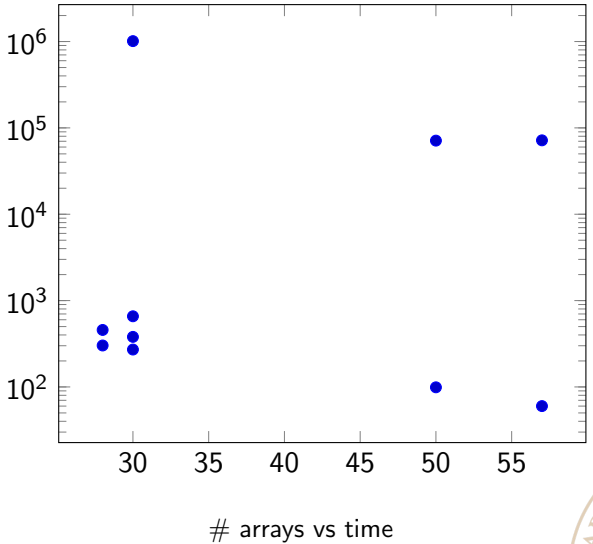

10^6
10^5
10^4
10^3
10^2
30
35
40
boolean variables vs time

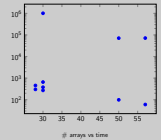


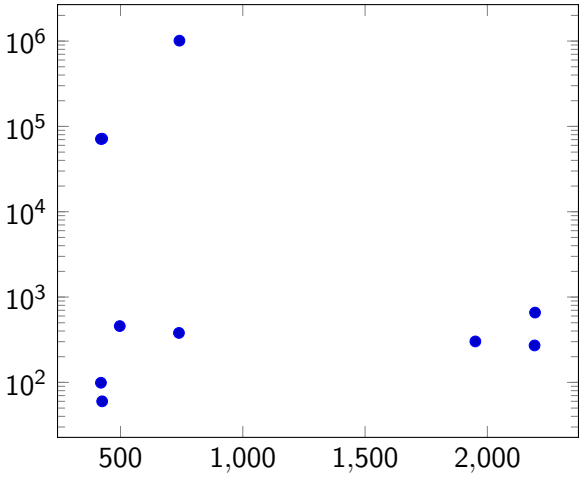


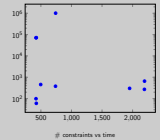
Total # of variables vs time

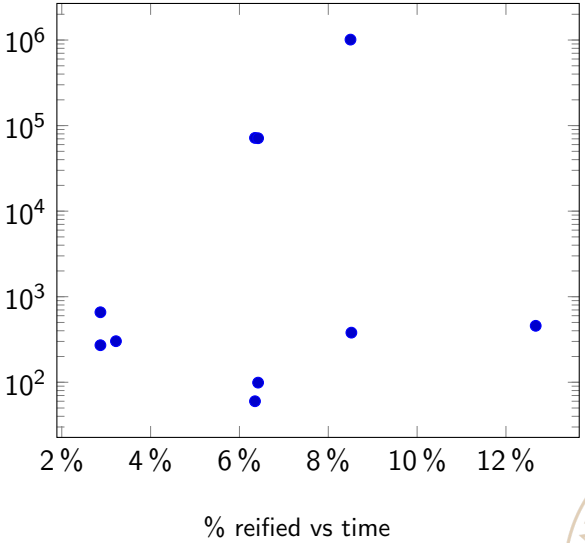


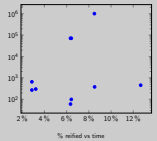

10^6
10^5
10^4
10^3
10^2
30
35
40
45
50
55
arrays vs time

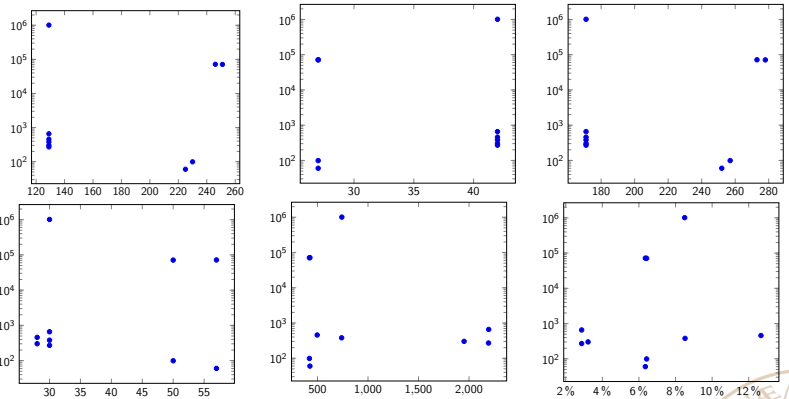


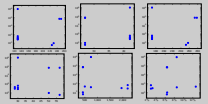

10^6
10^5
10^4
10^3
10^2
500
1,000
1,500
2,000
constraints vs time




10^6
10^5
10^4
10^3
10^2
2 %
4 %
6 %
8 %
10 %
12 %
% reified vs time




10^6
10^5
10^4
10^3
10^2
120 140 160 180 200 220 240 260
30 35 40
180 200 220 240 260 280
30 35 40 45 50 55
500 1,000 1,500 2,000
2% 4% 6% 8% 10% 12%



Conclusions

Conclusions

- Model produces solution just as good as handmade solution

Conclusions
Model produces solution just as good as handmade solution

Conclusions

- Model produces solution just as good as handmade solution
- Solver performance varies a lot

Conclusions
- Model produces solution just as good as handmade solution
- Solver performance varies a lot

Conclusions

- Model produces solution just as good as handmade solution
- Solver performance varies a lot
- Best performance: Gecode, all filters, MiniZinc 2.0.1

Conclusions
- Model produces solution just as good as handmade solution
- Solver performance varies a lot
- Best performance: Gecode, all filters, MiniZinc 2.0.1

Conclusions

- Model produces solution just as good as handmade solution
- Solver performance varies a lot
- Best performance: Gecode, all filters, MiniZinc 2.0.1
- Filters presented have a positive impact on runtime

Conclusions
- Model produces solution just as good as handmade solution
- Solver performance varies a lot
- Best performance: Gecode, all filters, MiniZinc 2.0.1
- Filters presented have a positive impact on runtime

Conclusions

- Model produces solution just as good as handmade solution
- Solver performance varies a lot
- Best performance: Gecode, all filters, MiniZinc 2.0.1
- Filters presented have a positive impact on runtime
- No relation between FlatZinc output and solver runtime

Conclusions
- Model produces solution just as good as handmade solution
- Solver performance varies a lot
- Best performance: Gecode, all filters, MiniZinc 2.0.1
- Filters presented have a positive impact on runtime
- No relation between FlatZinc output and solver runtime

Further work

- Test the result on a real robot
- Further testing of the filters
- More realistic representation of tools available

1. Kanske dyker upp problem missade. T.ex. kollisioner
2. Endast testat filter i grupp, alla filter kanske inte behövs, bygga vidare på dem som fungerade bäst
3. Vi antar att det finns en uppsättning tools till varje maskin, så kanske inte är fallet, finns kanske bara ett visst antal tools tillgängliga

Further work
Test the result on a real robot
Further testing of the filters
More realistic representation of tools available