Applying Constraint Programming to Multi-Skill Project Scheduling

Kenneth Young

Supervisors: Andreas Schutt and Thibaut Feydy

Intro: The Problem

Intro: The Problem

What is the Multi-Skill Project Scheduling Problem (MSPSP)?

Intro: The Problem

What is the Multi-Skill Project Scheduling Problem (MSPSP)?

- Activites

Intro: The Problem

What is the Multi-Skill Project Scheduling Problem (MSPSP)?

- Activites
- Workers

Intro: The Problem

What is the Multi-Skill Project Scheduling Problem (MSPSP)?

- Activites
- Workers
- Skills

Intro: The Problem

What is the Multi-Skill Project Scheduling Problem (MSPSP)?

- Activites
- Workers
- Skills

Aim: Find the fastest way to complete all the activities

Intro: The Problem

What is the Multi-Skill Project Scheduling Problem (MSPSP)?

- Activites
- Workers
- Skills

Aim: Find the fastest way to complete all the activities

Constraints

Intro: The Problem

What is the Multi-Skill Project Scheduling Problem (MSPSP)?

- Activites
- Workers
- Skills

Aim: Find the fastest way to complete all the activities

Constraints

- Activity constraint: Precedence relations between activities

Intro: The Problem

What is the Multi-Skill Project Scheduling Problem (MSPSP)?

- Activites
- Workers
- Skills

Aim: Find the fastest way to complete all the activities

Constraints

- Activity constraint: Precedence relations between activities
- Skill constraint: Activities require skills

Intro: The Problem

What is the Multi-Skill Project Scheduling Problem (MSPSP)?

- Activites
- Workers
- Skills

Aim: Find the fastest way to complete all the activities

Constraints

- Activity constraint: Precedence relations between activities
- Skill constraint: Activities require skills
- Worker constraint: Workers each have a variety of skills

Intro: Example

Table: Workers' Skills

	Alice	Bob	Carl	Dora
Programmer	-	✓	✓	✓
DB Designer	✓	-	-	-
Webmaster	✓	✓	-	✓

Intro: Example

Table: Workers' Skills

	Alice	Bob	Carl	Dora
Programmer	-	✓	✓	✓
DB Designer	✓	-	-	-
Webmaster	✓	✓	-	✓

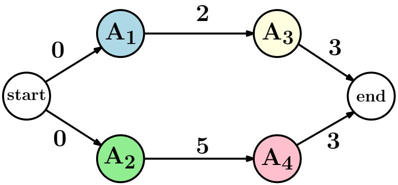


Figure: Precedence Graph

Intro: Example

Table: Workers' Skills

	Alice	Bob	Carl	Dora
Programmer	-	✓	✓	✓
DB Designer	✓	-	-	-
Webmaster	✓	✓	-	✓

Table: Skill Requirement

	A_1	A_2	A_3	A_4
Programmer	-	1	2	1
DB Designer	1	-	-	1
Webmaster	1	1	-	-

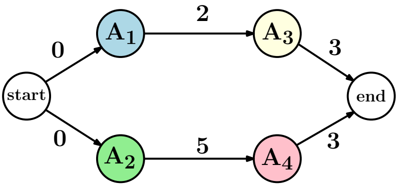


Figure: Precedence Graph

Intro: Example

Table: Workers' Skills

	Alice	Bob	Carl	Dora
Programmer	-	✓	✓	✓
DB Designer	✓	-	-	-
Webmaster	✓	✓	-	✓

Table: Skill Requirement

	A_1	A_2	A_3	A_4
Programmer	-	1	2	1
DB Designer	1	-	-	1
Webmaster	1	1	-	-

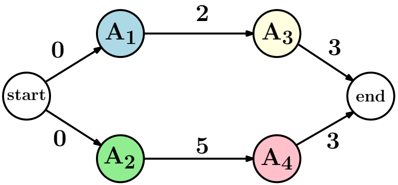


Figure: Precedence Graph

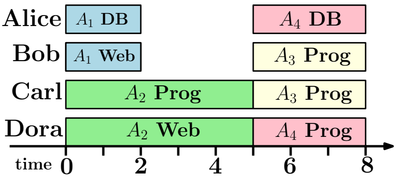


Figure: Schedule

Intro: Constraint Programming

Domain propagation

- Variables have domains of possible values
- Constraints reduce the size of these domains

Intro: Constraint Programming

Domain propagation

- Variables have domains of possible values
- Constraints reduce the size of these domains

Nogood learning

- Learn from failures
- Record these failures as constraints
- Use these constraints to make inferences

Intro: The Literature

- French research group
  - Principal researchers: Odile Belleguez-Morineau, Emmanuel Néron, Carlos Montoya
  - Exact branch and bound methods
  - Lower bounds
  - Adapted data from PSPLib

Intro: The Literature

- French research group
  - Principal researchers: Odile Belleguez-Morineau, Emmanuel Néron, Carlos Montoya
  - Exact branch and bound methods
  - Lower bounds
  - Adapted data from PSPLib
- Portuguese research group
  - Principal researchers: Bernardo Almeida, Isabel Correia, Francisco Saldanha-da-Gama
  - Constructive heuristics
  - Randomised search heuristics
  - Generated their own data

Intro: The Literature

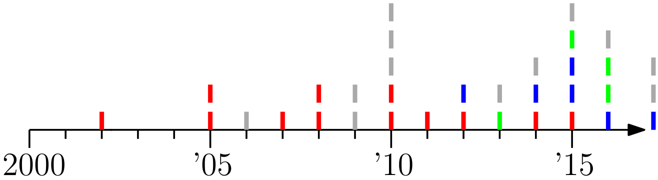
- French research group
  - Principal researchers: Odile Belleguez-Morineau, Emmanuel Néron, Carlos Montoya
  - Exact branch and bound methods
  - Lower bounds
  - Adapted data from PSPLib

- Portuguese research group
  - Principal researchers: Bernardo Almeida, Isabel Correia, Francisco Saldanha-da-Gama
  - Constructive heuristics
  - Randomised search heuristics
  - Generated their own data

- Polish research group
  - Principal researchers: Myszkowski, Skowronski
  - Randomised search heuristics
  - Generated their own data

Intro: Timeline of the Literature

- French
- Portuguese
- Polish
- Other



Model: Overview

Model: Overview

- Objective

Model: Overview

- Objective
  - Minimise the total project duration

Model: Overview

- Objective
  - Minimise the total project duration

- Two main decisions

Model: Overview

- Objective
  - Minimise the total project duration

- Two main decisions
  1. Scheduling decisions
     - Activity start times

Model: Overview

- Objective
  - Minimise the total project duration

- Two main decisions
  1. Scheduling decisions
     - Activity start times
  2. Assignment decisions
     - Workers to activities
     - Skill contribution of workers

Model: Constraints

Model: Constraints

- Precedence relations are respected

Model: Constraints

- Precedence relations are respected
- Workers perform only one activity at a time

Model: Constraints

- Precedence relations are respected
- Workers perform only one activity at a time
- Workers cannot multi-task

Model: Constraints

- Precedence relations are respected
- Workers perform only one activity at a time
- Workers cannot multi-task
- Skill requirement is satisfied
  - A worker for each skill must be present to perform the activity

Model: Constraints

- Precedence relations are respected
- Workers perform only one activity at a time
- Workers cannot multi-task
- Skill requirement is satisfied
  - A worker for each skill must be present to perform the activity
- Redundant constraints

Model: Choice of Constraints

Unary Resource Constraint

- Each worker only performs one activity at a time

Model: Choice of Constraints

Unary Resource Constraint

- Each worker only performs one activity at a time

Three equivalent ways of modelling

Model: Choice of Constraints

Unary Resource Constraint

- Each worker only performs one activity at a time

Three equivalent ways of modelling

1. Boolean satisfiability constraint using extra decision variable

Model: Choice of Constraints

Unary Resource Constraint

- Each worker only performs one activity at a time

Three equivalent ways of modelling

1. Boolean satisfiability constraint using extra decision variable
2. Disjunctive global constraint

Model: Choice of Constraints

Unary Resource Constraint

- Each worker only performs one activity at a time

Three equivalent ways of modelling

1. Boolean satisfiability constraint using extra decision variable
2. Disjunctive global constraint
3. Cumulative global constraint

Model: Decision Variables

Decision Variables	
s_i	Start time of activity $i \in V$
o_{ij}	1 iff activities i and j overlap for $(i, j) \in U$
x_{ir}	1 iff resource $r \in \mathscr{R}$ is assigned activity $i \in V$
y_{irk}	1 iff resource $r \in \mathscr{R}$ contributes with skill $k \in \mathscr{L}$ to activity $i \in V$

Model

$$
\begin{aligned}
& & \text{Min } s_{n+1} & & & (1)\\
\text{s.t.} \quad & & s_i + p_i \leq s_j & & \forall (i,j) \in E & \quad (2)\\
& & \neg o_{ij} \Leftrightarrow (s_i + p_i \leq s_j) \vee & & & \\
& & (s_j + p_j \leq s_i) & & \forall (i,j) \in U & \quad (3)\\
& & (x_{ir} \wedge x_{jr}) \Rightarrow \neg o_{ij} & & \forall (i,j) \in U,\ r \in \mathscr{R}V_i \cap \mathscr{R}V_j & \quad (4)\\
& & \sum_{r \in \mathscr{R}V_i} y_{irk} = Req_{ik} & & \forall i \in V,\ k \in \mathscr{L}_i & \quad (5)\\
& & \sum_{k \in \mathscr{L}_i | Mast_{rk}=1} y_{irk} \leq 1 & & \forall i \in V,\ r \in \mathscr{R}V_i & \quad (6)\\
& & y_{irk} \leq Mast_{rk} & & \forall i \in V,\ r \in \mathscr{R}V_i,\ k \in \mathscr{L} & \quad (7)\\
& & y_{irk} \leq x_{ir} & & \forall i \in V,\ r \in \mathscr{R}V_i,\ k \in \mathscr{L}_i & \quad (8)\\
& & s_i \geq 0 & & \forall i \in V & \quad (9)
\end{aligned}
$$

Model: Redundent Constraints

$$\texttt{cumulative}(s,\ p,\ [\ Req_{ik} : i \in V],\ |\mathscr{R}_k|) \qquad \forall k \in \mathscr{L} \qquad (10)$$

$$\texttt{cumulative}\left(s,\ p,\ \left[\sum_{k \in \mathscr{L}} Req_{ik} : i \in V\right],\ m\right) \qquad (11)$$

$$x_{ir} = 0 \ \ \forall i \in V,\ r \in \mathscr{R} \backslash \mathscr{R} V_i \qquad (12)$$

$$y_{irk} = 0 \ \ \forall i \in V,\ r \in \mathscr{R} \backslash \mathscr{R} V_i,\ k \in \mathscr{L} \qquad (13)$$

$$y_{irk} = 0 \ \ \forall i \in V,\ r \in \mathscr{R},\ k \in \mathscr{L} \backslash \mathscr{L}_i \qquad (14)$$

$$y_{irk} = 0 \ \ \forall r \in \mathscr{R},\ k \in \mathscr{L} \backslash \mathscr{L}_r,\ i \in V \qquad (15)$$

Data: Overview

Data: Overview

- Generated our own data
  - equivalent to the Portuguese group's data

Data: Overview

- Generated our own data
  - equivalent to the Portuguese group's data
- Small data set: 216 unique instances
  - 20 activities
  - 4 skills
  - 10-30 workers

Data: Overview

- Generated our own data
  - equivalent to the Portuguese group's data
- Small data set: 216 unique instances
  - 20 activities
  - 4 skills
  - 10-30 workers
  - 13 unsolved

Data: Overview

- Generated our own data
  - equivalent to the Portuguese group's data
- Small data set: 216 unique instances
  - 20 activities
  - 4 skills
  - 10-30 workers
  - 13 unsolved
- Large data set: 216 unique instances
  - 40 activities
  - 4 skills
  - 20-60 workers

Data: Overview

- Generated our own data
  - equivalent to the Portuguese group's data
- Small data set: 216 unique instances
  - 20 activities
  - 4 skills
  - 10-30 workers
  - 13 unsolved
- Large data set: 216 unique instances
  - 40 activities
  - 4 skills
  - 20-60 workers
  - 211 unsolved

Data: Complexity Measures

1. Skill Factor
   - $SF \in \{1,\ 0.75,\ 0.5,\ variable\}$

Data: Complexity Measures

1. Skill Factor
   - $SF \in \{1,\ 0.75,\ 0.5,\ \mathit{variable}\}$
2. Network Complexity
   - $NC \in \{1.5,\ 1.8,\ 2.1\}$

Data: Complexity Measures

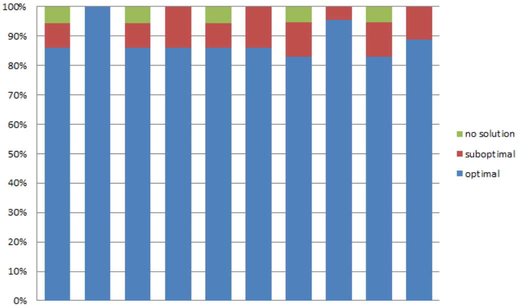
1. Skill Factor
   - $SF \in \{1,\ 0.75,\ 0.5,\ \mathit{variable}\}$
2. Network Complexity
   - $NC \in \{1.5,\ 1.8,\ 2.1\}$
3. Modified Resource Strength
   - varied over 3 values
   - $$MRS = \frac{m}{\sum_{i \in V} \sum_{k \in \mathscr{L}} Req_{ik}}$$

Experiments: Constraint Choice

Sample of 72 small instances

Experiments: Constraint Choice

Sample of 72 small instances



free search	0	1	0	1	0	1	0	1	0	1
edge finding	-	-	0	0	1	1	0	0	1	1
	bool		disjunctive				cumulative			

Experiments: Search Strategies

Experiments: Search Strategies

- Start times (s_i)

Experiments: Search Strategies

- Start times (s_i)
- Start times (s_i), then worker assignment (x_{ir})

Experiments: Search Strategies

- Start times (s_i)
- Start times (s_i), then worker assignment (x_{ir})
- Start times (s_i), then contrbution of each worker (y_{irk})

Experiments: Search Strategies

- Start times (s_i)
- Start times (s_i), then worker assignment (x_{ir})
- Start times (s_i), then contrbution of each worker (y_{irk})
- Naïve activity-based

Experiments: Small Data Set

- Tested on all 216 small instances
- Time limit of 300 seconds

Experiments: Small Data Set

- Tested on all 216 small instances
- Time limit of 300 seconds

search strategy	#no soln	%gap	#optimal	%optimal	avg. runtime
default	0	0.00	216	100.00	3.25s
start	0	2.50	215	99.54	1.26s
start then worker	0	0.00	216	100.00	2.89s
start then skill	**0**	**0.00**	**216**	**100.00**	**1.63s**
naïve activity-based	0	2.50	215	99.54	0.82s

Experiments: Small Data Set

- Tested on all 216 small instances
- Time limit of 300 seconds

search strategy	#no soln	%gap	#optimal	%optimal	avg. runtime
default	0	0.00	216	100.00	3.25s
start	0	2.50	215	99.54	1.26s
start then worker	0	0.00	216	100.00	2.89s
start then skill	**0**	**0.00**	**216**	**100.00**	**1.63s**
naïve activity-based	0	2.50	215	99.54	0.82s
activity smallest	0	0.00	216	100.00	0.45s

Experiments: Large Data Set

- Tested on all 216 large instances
- Time limit of 300 seconds

Experiments: Large Data Set

- Tested on all 216 large instances
- Time limit of 300 seconds

variable selection	#nodes	#props	#no soln	%gap	#opt	%opt	opt rt.	total rt.
naive	6,568k	375m	0	60.36	13	6.02	98.15	287.85
smallest	**734k**	**71m**	**0**	**53.29**	**15**	**6.94**	**103.51**	**286.35**
smallest_largest	814k	73m	0	55.50	7	3.24	70.71	292.57
first_fail	1,037k	80m	0	67.48	8	3.70	113.33	293.09

Experiments: Large Data Set

- Tested on all 216 large instances
- Time limit of 300 seconds

variable selection	#nodes	#props	#no soln	%gap	#opt	%opt	opt rt.	total rt.
naive	6,568k	375m	0	60.36	13	6.02	98.15	287.85
smallest	**734k**	**71m**	**0**	**53.29**	**15**	**6.94**	**103.51**	**286.35**
smallest_largest	814k	73m	0	55.50	7	3.24	70.71	292.57
first_fail	1,037k	80m	0	67.48	8	3.70	113.33	293.09
smallest wth UB	715k	69m	5	54.18	16	7.41	78.88	283.62

Experiments: Large Data Set

- Tested on all 216 large instances
- Time limit of 300 seconds

variable selection	#nodes	#props	#no soln	%gap	#opt	%opt	opt rt.	total rt.
naive	6,568k	375m	0	60.36	13	6.02	98.15	287.85
smallest	**734k**	**71m**	**0**	**53.29**	**15**	**6.94**	**103.51**	**286.35**
smallest_largest	814k	73m	0	55.50	7	3.24	70.71	292.57
first_fail	1,037k	80m	0	67.48	8	3.70	113.33	293.09
smallest wth UB	715k	69m	5	54.18	16	7.41	78.88	283.62
smallest (fixed)	3,962k	409m	0	50.18	24	11.11	55.45	272.83

Experiments: Large Data Set

- Time limit of 3600 seconds

Experiments: Large Data Set

- Time limit of 3600 seconds

measure	value	#nodes	#props	%gap	#opt	%opt	opt rt.	total rt.
SF	1	36m	4,254m	48.35	7	12.96	257.55	3166.72
	0.75	47m	4,296m	49.63	4	7.41	545.44	3373.74
	0.5	32m	4,468m	42.06	20	37.04	782.20	2556.37
	var	45m	4,302m	49.82	5	9.26	130.97	3278.79
NC	1.5	41m	5,007m	57.33	8	11.11	819.48	3291.05
	1.8	42m	4,381m	46.09	11	15.28	558.06	3135.26
	2.1	36m	3,602m	38.99	17	23.61	446.41	2855.40
MRS	#1	37m	6,083m	76.93	14	19.44	327.52	2963.68
	#2	42m	4,154m	43.36	8	11.11	913.89	3301.54
	#3	39m	2,753m	23.94	14	19.44	599.07	3016.49
Overall		**40m**	**4,330m**	**47.92**	**36**	**16.67**	**563.43**	**3093.90**

Summary

- Applied the constraint programming solver chuffed to the MSPSP

Summary

- Applied the constraint programming solver chuffed to the MSPSP
- Generated a set of benchmark instances

Summary

- Applied the constraint programming solver chuffed to the MSPSP
- Generated a set of benchmark instances
- Found an effective model formulation

Summary

- Applied the constraint programming solver chuffed to the MSPSP
- Generated a set of benchmark instances
- Found an effective model formulation
- Solved all small instances

Future Work

- Apply activity-based search to the large dataset

Future Work

- Apply activity-based search to the large dataset
- Create a more structured search procedure in chuffed

Acknowledgements

- Dr. Andreas Schutt
- Dr. Thibaut Feydy
- Adrian Goldwaser

Thanks for listening!

Questions?