A Logic-Based Benders Decomposition of an Assembly Line Balancing and Scheduling Problem

Kenneth D. Young

ModRef 2017 Workshop

Masters Thesis

Supervisor: Dr. Alysson M. Costa

28th of August, 2017

Overview

1. The Problem: Assembly Line Balancing
2. The Method: Benders Decomposition
3. Results

Assembly Lines through History

Ford Motor Company, Detroit 1915

Volkswagen, Wolfsburg 1951

Assembly Lines through History

Kia Motors, Žilina 2012

The Assembly Line Balancing Problem

Production system designed to *efficiently* manufacture a product

The Assembly Line Balancing Problem

Production system designed to *efficiently* manufacture a product

1. Tasks
2. Work stations
3. Precedence Relations

The Assembly Line Balancing Problem

Production system designed to *efficiently* manufacture a product

1. Tasks
2. Work stations
3. Precedence Relations

Def$^{\underline{n}}$: The cycle time of the line is the largest workload among all stations

$$c = \max\{\, l_k \mid k \in K \,\}$$

where K is the set of stations and l_k is the load of station k

The Type-2 ALBP

- Two main kinds of assembly line problems: Type-1 and Type-2.

The Type-2 ALBP

- Two main kinds of assembly line problems: Type-1 and Type-2.
- We consider the Type-2 case:
  - Fixed number of work stations
  - Variable cycle time

The Type-2 ALBP

- Two main kinds of assembly line problems: Type-1 and Type-2.
- We consider the Type-2 case:
  - Fixed number of work stations
  - Variable cycle time
- Aim: Maximize production rate $\iff$ Minimize cycle time
  - By balancing workload across the stations

The Type-2 ALBP

- Two main kinds of assembly line problems: Type-1 and Type-2.
- We consider the Type-2 case:
  - Fixed number of work stations
  - Variable cycle time
- Aim: Maximize production rate $\iff$ Minimize cycle time
  - By balancing workload across the stations
- What are our decisions?
  - Assigning tasks to stations

Simple Assembly Line Example

4 Tasks

3 Stations

Simple Assembly Line Example

4 Tasks

3 Stations

Simple Assembly Line Example

4 Tasks

3 Stations

Sequence-Dependent Setup Times

- Theory usually assumes task sequence within a station can be arbitrary
  - Not always the case in practice!

Sequence-Dependent Setup Times

- Theory usually assumes task sequence within a station can be arbitrary
  - Not always the case in practice!
- Setup times between tasks can vary a lot
  - Walking distances
  - Tool-changes (esp. in robotic assembly lines)
  - Cooling periods

Sequence-Dependent Setup Times

- Theory usually assumes task sequence within a station can be arbitrary
  - Not always the case in practice!
- Setup times between tasks can vary a lot
  - Walking distances
  - Tool-changes (esp. in robotic assembly lines)
  - Cooling periods
- Considering setups leads to the SetUp Assembly Line Balancing and Scheduling Problem, or SUALBSP
  - Type-2 problem is denoted SUALBSP-2

The SUALBSP-2

- SUALBSP-2 can be viewed as m Travelling Salesperson Problems where the set of cities for each TSP is unknown

The SUALBSP-2

- SUALBSP-2 can be viewed as m Travelling Salesperson Problems where the set of cities for each TSP is unknown
- Two distinct types of setups/distances between tasks/cities
  - ϕ_{ij} = Forward setup between tasks on same product
  - β_{ij} = Backward setup between last task on product p and first task on $p+1$

The SUALBSP-2

- SUALBSP-2 can be viewed as m Travelling Salesperson Problems where the set of cities for each TSP is unknown
- Two distinct types of setups/distances between tasks/cities
  - ϕ_{ij} = Forward setup between tasks on same product
  - β_{ij} = Backward setup between last task on product p and first task on $p+1$
- Consider one station with tasks T_i, T_j and T_k:

Example with Setup Times

ϕ_{ij}	T_1	T_2	T_3	T_4
T_1	–	5	5	5
T_2	–	–	2	5
T_3	–	–	–	1
T_4	–	–	5	–

β_{ij}	T_1	T_2	T_3	T_4
T_1	0	–	–	–
T_2	5	0	–	–
T_3	5	5	0	5
T_4	5	5	1	0

Example with Setup Times

ϕ_{ij}	T_1	T_2	T_3	T_4
T_1	–	5	5	5
T_2	–	–	2	5
T_3	–	–	–	1
T_4	–	–	5	–

β_{ij}	T_1	T_2	T_3	T_4
T_1	0	–	–	–
T_2	5	0	–	–
T_3	5	5	0	5
T_4	5	5	1	0

Example with Setup Times

ϕ_{ij}	T_1	T_2	T_3	T_4
T_1	–	5	5	5
T_2	–	–	2	5
T_3	–	–	–	1
T_4	–	–	5	–

β_{ij}	T_1	T_2	T_3	T_4
T_1	0	–	–	–
T_2	5	0	–	–
T_3	5	5	0	5
T_4	5	5	1	0

Existing Approaches

Data

- Huge data library for ALBPs
  - `http://assembly-line-balancing.mansci.de/`
- 1076 instances of the SUALBSP

Data

- Huge data library for ALBPs
  - `http://assembly-line-balancing.mansci.de/`
- 1076 instances of the SUALBSP
- We chose 3 subsets with small, medium and large numbers of tasks
  - Total of 396 instances

Data

- Huge data library for ALBPs
  - `http://assembly-line-balancing.mansci.de/`
- 1076 instances of the SUALBSP
- We chose 3 subsets with small, medium and large numbers of tasks
  - Total of 396 instances
  - 288 unsolved

Data

- Huge data library for ALBPs
  - `http://assembly-line-balancing.mansci.de/`
- 1076 instances of the SUALBSP
- We chose 3 subsets with small, medium and large numbers of tasks
  - Total of 396 instances
  - 288 unsolved

Data set	# Instances	# Tasks	# Stations	# Precedences
1	108	7-21 (small)	2-8	6-27
2	112	25-30 (medium)	3-14	32-40
3	176	32-58 (large)	3-31	38-62

Exact Solution Procedure

Best Known MIPs for SUALBSP-2

Best Known MIPs for SUALBSP-2

$$\text{SCBF-2: } \min c \qquad (33)$$

$$\sum_{j \in F_i^F} y_{ij} + \sum_{j \in F_i^B} w_{ij} = 1 \qquad \forall i \in V \qquad (4)$$

$$\sum_{i \in P_j^F} y_{ij} + \sum_{i \in P_j^B} w_{ij} = 1 \qquad \forall j \in V \qquad (5)$$

$$s_i + t_i + \sum_{j \in F_i^B} \mu_{ij} \cdot w_{ij} \leq c \qquad \forall i \in V \qquad (30)$$

$$r_i + 1 \leq r_j \qquad \forall (i, j) \in E \qquad (44)$$

$$q_{ij} + q_{ji} = 1 \qquad \forall i \in V \text{ and } j \in V \backslash (P_i^* \cup F_i^*) \text{ with } i < j \qquad (54)$$

$$r_i + 1 + (n - |F_i^*| - |P_j^*|) \times (q_{ij} - 1) \leq r_j \qquad \forall i \in V \text{ and } j \in V \backslash (P_i^* \cup F_i^*) \text{ with } i \neq j \qquad (55)$$

$$r_j - 1 + (n - |F_j^*| - |P_i^*| - 2) \times (y_{ij} - 1) \leq r_i \qquad \forall i \in V \text{ and } j \in F_i^F \qquad (56)$$

$$y_{ij} \leq q_{ij} \qquad \forall i \in V \text{ and } j \in F_i^F \backslash F_i^* \qquad (57)$$

$$w_{ji} \leq q_{ij} \qquad \forall i \in V \text{ and } j \in P_i^B \backslash F_i^* \text{ with } i \neq j \qquad (58)$$

$$q_{ij} \in \{0, 1\} \qquad \forall i \in V \text{ and } j \in V \backslash (P_i^* \cup F_i^*) \text{ with } i \neq j \qquad (59)$$

$$s_i + (t_i + \tau_{ij}) + (\bar{c} + \tau_{ij}) \cdot (y_{ij} - 1) \leq s_j \qquad \forall i \in V \text{ and } j \in F_i^F \qquad (60)$$

$$\sum_{i \in V} \sum_{j \in F_i^B} w_{ij} = m \qquad (61)$$

$$|P_i^*| + 1 \leq r_i \leq n - |F_i^*| \qquad \forall i \in V \qquad (51)$$

$$\underline{c} \leq c \leq \bar{c} \qquad (37)$$

$$y_{ij} \in \{0, 1\} \qquad \forall i \in V \text{ and } j \in F_i^F \qquad (13)$$

$$w_{ij} \in \{0, 1\} \qquad \forall i \in V \text{ and } j \in F_i^B \qquad (14)$$

$$s_i \geq 0 \qquad \forall i \in V \qquad (32)$$

Best Known MIPs for SUALBSP-2

SCBF-2: $\min c$ (33)

$$\sum_{j \in F_i^F} y_{ij} + \sum_{j \in F_i^B} w_{ij} = 1 \qquad \forall i$$

$$\sum_{i \in P_j^F} y_{ij} + \sum_{i \in P_j^B} w_{ij} = 1 \qquad \forall j$$

$$s_i + t_i + \sum_{j \in F_i^B} \mu_{ij} \cdot w_{ij} \le c$$

$$r_i + 1 \le r_j \qquad \forall (i,j) \in E$$

$$q_{ij} + q_{ji} = 1 \qquad \forall i \in V \text{ and } j$$

$$r_i + 1 + (n - |F_i^*| - |P_j^*|) \times (q_{ij} - 1) \le r_j \qquad \forall i \in V \text{ and } j$$

$$r_j - 1 + (n - |F_j^*| - |P_i^*| - 2) \times (y_{ij} - 1) \le r_i \qquad \forall i \in V \text{ and } j$$

$$y_{ij} \le q_{ij} \qquad \forall i \in V \text{ and } j$$

$$w_{ji} \le q_{ij} \qquad \forall i \in V \text{ and } j$$

$$q_{ij} \in \{0,1\} \qquad \forall i \in V \text{ and } j$$

$$s_i + (t_i + \tau_{ij}) + (\bar{c} + \tau_{ij}) \cdot (y_{ij} - 1) \le s_j$$

$$\sum_{i \in V} \sum_{j \in F_i^B} w_{ij} = m$$

$$|P_i^*| + 1 \le r_i \le n - |F_i^*| \qquad \forall i \in V$$

$$\underline{c} \le c \le \bar{c}$$

$$y_{ij} \in \{0,1\}$$

$$w_{ij} \in \{0,1\}$$

$$s_i \ge 0$$

FSBF-2: $\text{Min } c$ (3.1)

$$\text{s.t.} \quad \sum_{k \in FS_i} x_{ik} = 1 \qquad \forall i \in V \qquad (3.2)$$

$$\sum_{k \in FS_i} k \cdot x_{ik} = w_i \qquad \forall i \in V \qquad (3.3)$$

$$\sum_{j \in F_i^\phi} y_{ij} + \sum_{j \in F_i^\beta} z_{ij} = 1 \qquad \forall i \in V \qquad (3.4)$$

$$\sum_{i \in P_j^\phi} y_{ij} + \sum_{i \in P_j^\beta} z_{ij} = 1 \qquad \forall j \in V \qquad (3.5)$$

$$o_{ik} \le x_{ik} \qquad \forall i \in V,\ k \in FS_i \qquad (3.6)$$

$$\sum_{j \in F_i^\beta} z_{ij} \le \sum_{k \in FS_i} o_{ik} \qquad \forall i \in V \qquad (3.7)$$

$$\sum_{i \in FT_k} o_{ik} \le 1 \qquad \forall k \in K \qquad (3.8)$$

$$w_j - w_i \le M \cdot (1 - y_{ij}) \text{ and } w_i - w_j \le M \cdot (1 - y_{ij}) \qquad \forall i \in V,\ j \in F_i^\phi \qquad (3.9)$$

$$w_j - w_i \le M \cdot (1 - z_{ij}) \text{ and } w_i - w_j \le M \cdot (1 - z_{ij}) \qquad \forall i \in V,\ j \in F_i^\beta \qquad (3.10)$$

$$\sum_{i \in FT_k} t_i \cdot x_{ik} \le c \qquad \forall k \in K \qquad (3.11)$$

$$s_i + t_i + \sum_{j \in F_i^\beta} \beta_{ij} \cdot z_{ij} \le c \qquad \forall i \in V \qquad (3.12)$$

$$\sum_{i \in V} \sum_{j \in F_i^\beta} z_{ij} \ge m \qquad (3.13)$$

$$s_i + c \cdot (w_i - w_j) + t_i + \phi_{ij} \cdot y_{ij} \le s_j \qquad \forall (i,j) \in E \qquad (3.14)$$

$$s_i + (t_i + \phi_{ij}) + (c + \phi_{ij}) \cdot (y_{ij} - 1) \le s_j \qquad \forall i \in V,\ j \in F_i^\phi \setminus F_i \qquad (3.15)$$

$$\underline{c} \le c \le \bar{c} \qquad (3.16)$$

$$s_i \ge 0 \qquad \forall i \in V \qquad (3.17)$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in V,\ k \in FS_i \qquad (3.18)$$

$$y_{ij} \in \{0,1\} \qquad \forall i \in V,\ j \in F_i^\phi \qquad (3.19)$$

Results: Mixed Integer Programs

- Time limit: 30 minutes for each instance

Results: Mixed Integer Programs

- Time limit: 30 minutes for each instance

# Tasks	Model	Gap (%)	No Sol. (%)	Optimal (%)	Runtime (s)
7–21	MIP-1	0.16	0.0	93.5	166.57
	MIP-2	1.62	0.0	86.1	374.88
25–30	MIP-1	13.76	0.0	**0.0**	1800.66
	MIP-2	18.01	**25.0**	**0.0**	1800.33
32–58	MIP-1	19.56	5.1	**3.9**	1759.67
	MIP-2	10.09	**61.9**	**0.0**	1800.88

Results: Mixed Integer Programs

- Time limit: 30 minutes for each instance

# Tasks	Model	Gap (%)	No Sol. (%)	Optimal (%)	Runtime (s)
7–21	MIP-1	0.16	0.0	93.5	166.57
	MIP-2	1.62	0.0	86.1	374.88
25–30	MIP-1	13.76	0.0	**0.0**	1800.66
	MIP-2	18.01	**25.0**	**0.0**	1800.33
32–58	MIP-1	19.56	5.1	**3.9**	1759.67
	MIP-2	10.09	**61.9**	**0.0**	1800.88

"There is a need for more efficient exact methods."

— Esmaeilbeigi *et al.* (2016)

Our Method

Benders Decomposition: History

- Initially proposed by J. F. Benders in 1962
  - Created to efficiently solve MIPs

Benders Decomposition: History

- Initially proposed by J. F. Benders in 1962
  - Created to efficiently solve MIPs
- Largely neglected by the literature for ~ 40 years

Benders Decomposition: History

- Initially proposed by J. F. Benders in 1962
  - Created to efficiently solve MIPs
- Largely neglected by the literature for ~ 40 years
- Recently re-emerged showing it's applicability
  - See surveys by Costa (2005) and Rahmaniani *et al.* (2016)

Benders Decomposition: History

- Initially proposed by J. F. Benders in 1962
  - Created to efficiently solve MIPs
- Largely neglected by the literature for ~ 40 years
- Recently re-emerged showing it's applicability
  - See surveys by Costa (2005) and Rahmaniani *et al.* (2016)
- Generalised to *logic-based* Benders decomposition by Hooker (2003)
  - Can now be applied to a much wider range of problems

Benders Decomposition: Crash Course

- Strategy: "learn from your mistakes"

Benders Decomposition: Crash Course

- Strategy: "learn from your mistakes"
- Two types of decisions: primary and secondary

Benders Decomposition: Crash Course

- Strategy: "learn from your mistakes"
- Two types of decisions: primary and secondary
- Divide the problem into a master problem and a sub-problem
  - Iterate between them until optimality

Benders Decomposition: Crash Course

- Strategy: "learn from your mistakes"
- Two types of decisions: primary and secondary
- Divide the problem into a master problem and a sub-problem
  - Iterate between them until optimality

- Well-suited to problems with a combinatorial number of constraints
  - But only a select few are relevant for finding optimality

Benders Decomposition: Crash Course

- Strategy: "learn from your mistakes"
- Two types of decisions: primary and secondary
- Divide the problem into a master problem and a sub-problem
  - Iterate between them until optimality

- Well-suited to problems with a combinatorial number of constraints
  - But only a select few are relevant for finding optimality
- Sub-problem tells Master where the most important infeasibilities are

Decomposing the Assembly Line

Separate decisions of the Assembly Line Problem:

Decomposing the Assembly Line

Separate decisions of the Assembly Line Problem:

Primary

Assignment of tasks to stations

Secondary

Sequencing of tasks within each station

RMP:
Assign tasks
to stations

Decomposing the Assembly Line

Separate decisions of the Assembly Line Problem:

Primary

Assignment of tasks to stations

Secondary

Sequencing of tasks within each station

Decomposing the Assembly Line

Separate decisions of the Assembly Line Problem:

Primary

Assignment of tasks to stations

Secondary

Sequencing of tasks within each station

Framework of our Decomposition

Solution Methodology: Overview

Master Problem Formulation

Assignment problem $\Rightarrow$ Formulate a MIP, solve with Gurobi

Master Problem Formulation

Assignment problem $\Rightarrow$ Formulate a MIP, solve with Gurobi

$$
\begin{aligned}
&\text{RMP}(\mu)\text{: Min } c^{\mu} \\
\text{s.t.} \quad & \sum_{k \in FS_i} x_{ik}^{\mu} = 1 && \forall i \in V \\
& \sum_{k \in FS_i} k \cdot x_{ik}^{\mu} \leq \sum_{k \in FS_j} k \cdot x_{jk}^{\mu} && \forall (i,j) \in E \\
& \sum_{i \in V} t_i \cdot x_{ik}^{\mu} \leq c^{\mu} && \forall k \in K \\
& \mathcal{C}^{\nu} && \nu \in \{1, 2, \ldots, \mu - 1\} \\
& \underline{c}^{\mu} \leq c^{\mu} \leq \bar{c}^{\mu} \\
& x_{ik}^{\mu} \in \{0, 1\} && \forall i \in V,\ k \in FS_i
\end{aligned}
$$

Sub-Problem Formulation

- 'Kind of' similar to the asymmetric TSP $\Rightarrow$ highly combinatorial

Sub-Problem Formulation

- 'Kind of' similar to the asymmetric TSP $\Rightarrow$ highly combinatorial

Sub-Problem Formulation

- 'Kind of' similar to the asymmetric TSP ⇒ highly combinatorial

Sub-Problem Formulation

- 'Kind of' similar to the asymmetric TSP ⇒ highly combinatorial

- Possible approaches
  - Mixed-Integer Problem
  - Constraint Problem
  - Travelling Salesperson Problem (requires some adaptation)

Sub-Problem Formulation

- 'Kind of' similar to the asymmetric TSP ⇒ highly combinatorial

- Possible approaches
  - Mixed-Integer Problem
  - Constraint Problem
  - Travelling Salesperson Problem (requires some adaptation)
    - ★ Pro: Use dedicated TSP solvers (Concorde, `tsp_solve` etc.)
    - ★ Con: Leads to multiplying number of tasks by $2n^2$ ⇒ Did not explore

Sub-Problem: Mixed Integer Program

SP-MIP(μ): Min l_k^μ

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{j \in F_i^\phi} y_{ij} + \sum_{j \in F_i^\beta} z_{ij} = 1 && \forall i \in V_k^\mu \\
& \sum_{i \in P_j^\phi} y_{ij} + \sum_{i \in P_j^\beta} z_{ij} = 1 && \forall j \in V_k^\mu \\
& \sum_{i \in V_k^\mu} \sum_{j \in F_i^\beta} z_{ij} = 1 \\
& s_i + t_i + \phi_{ij} \cdot y_{ij} \leq s_j && \forall (i,j) \in E_k^\mu \\
& s_i + t_i + \phi_{ij} \leq s_j + M(1 - y_{ij}) && \forall i \in V_k^\mu,\ j \in F_i^\phi \\
& s_i + t_i + \sum_{j \in F_i^\beta} \beta_{ij} \cdot z_{ij} \leq l_k^\mu && \forall i \in V_k^\mu \\
& l_k^\mu \geq 0,\ \ s_i \geq 0 && \forall i \in V_k^\mu \\
& y_{ij}^\mu\ (z_{ij}^\mu) \in \{0,1\} && \forall i \in V_k^\mu,\ j \in F_i^\phi (F_i^\beta)
\end{aligned}
$$

Sub-Problem: Constraint Program

- Contraint Programming handles scheduling problems well

Sub-Problem: Constraint Program

- Contraint Programming handles scheduling problems well
- Expressive nature of CP removes need for big-M constraints

Sub-Problem: Constraint Program

- Contraint Programming handles scheduling problems well
- Expressive nature of CP removes need for big-M constraints
- `Chuffed` is a state-of-the-art solver for many scheduling problems

Sub-Problem: Constraint Program

- Contraint Programming handles scheduling problems well
- Expressive nature of CP removes need for big-M constraints
- `Chuffed` is a state-of-the-art solver for many scheduling problems
- 'Learning' solver which combines
  - Lazy clause generation
  - Boolean satisfiability solving

CP Solver Chuffed: Global Constraints

- First main advantage of CP
- Capture common combinatorial structures across problems
- Provide effective propagation for CP solvers

CP Solver Chuffed: Global Constraints

- First main advantage of CP
- Capture common combinatorial structures across problems
- Provide effective propagation for CP solvers
- Some include:
  - `knapsack()`
  - `disjunctive()`
  - `cumulative()`
  - `bin_packing()`
  - `int_set_channel()`
  - `diffn()`
  - `geost()`
  - `all_different()`
  - ⋮

CP Solver Chuffed: Search Strategy

- Second main advantage of CP
- Combinatorial problem $\Rightarrow$ Constant battle against huge solution space

CP Solver Chuffed: Search Strategy

- Second main advantage of CP
- Combinatorial problem $\Rightarrow$ Constant battle against huge solution space
- Search strategies are a powerful weapon here

CP Solver Chuffed: Search Strategy

- Second main advantage of CP
- Combinatorial problem $\Rightarrow$ Constant battle against huge solution space
- Search strategies are a powerful weapon here

`Chuffed`'s search language:

CP Solver Chuffed: Search Strategy

- Second main advantage of CP
- Combinatorial problem $\Rightarrow$ Constant battle against huge solution space
- Search strategies are a powerful weapon here

`Chuffed`'s search language:

- Basic searches
  - Integers: `int_search`
  - Booleans: `bool_search`

CP Solver Chuffed: Search Strategy

- Second main advantage of CP
- Combinatorial problem $\Rightarrow$ Constant battle against huge solution space
- Search strategies are a powerful weapon here

`Chuffed`'s search language:

- Basic searches
  - Integers: `int_search`
  - Booleans: `bool_search`
- Compositional searches
  - Sequential: `seq_search`
  - Priority: `priority_search`

Benders Cuts

- Communicate information from sub-problems to master problem
- Richer information leads to faster convergence

Benders Cuts

- Communicate information from sub-problems to master problem
- Richer information leads to faster convergence
- We proposed a range of cutting procedures:

Feasibility Cuts	Optimality Cuts
Nogood cut	Logic cut
	Inference cut #1 (simple)
	Inference cut #2 (smarter)
	Inference cut #3 (smartest)
	Global cut/bound

Benders Cuts: Inference Cut Example

Inference cut #1 (simple):

$$l_k^\nu - M \sum_{i \in V_k^\nu} (1 - x_{ik}^\mu) \leq c^\mu$$

- If same assignment as before $\Rightarrow$ Lower bound on cycle time

Benders Cuts: Inference Cut Example

Inference cut #1 (simple):

$$l_k^\nu - M \sum_{i \in V_k^\nu} (1 - x_{ik}^\mu) \leq c^\mu$$

- If same assignment as before $\Rightarrow$ Lower bound on cycle time

Inference cut #3 (smartest):

$$l_k^\nu - \sum_{i \in V_k^\nu} B_{ik}^\nu (1 - x_{ik}^\mu) + \sum_{i \in V \setminus V_k^\nu} b_{ik}^\nu \cdot x_{ik}^\mu \leq c^\mu$$

- If *similar* assignment as before $\Rightarrow$ Lower bound on cycle time

Results

Benders Results: Sub-Problem Formulations

- Instances with 25-30 tasks
- Time limit: 30 minutes on each instance

#Tasks	Type	#SP Nodes	%Gap	%Opt.	SP Runtime(s)
25–30	SP-MIP	639k	11.7	82.1	226.2
	SP-CP	1,692k	13.0	**83.9**	229.4

Benders Results: Comparison with MIPs

- All 3 data sets
- Time limit: 30 minutes on each instance

	Benders			MIP-1			MIP-2		
#Tasks	%Gap	%NoSol.	%Opt.	%Gap	%NoSol.	%Opt.	%Gap	%NoSol.	%Opt.
7–21	0.0	0.0	100.0	0.1	0.0	93.5	1.6	0.0	77.8
25–30	13.0	0.0	83.9	13.8	0.0	0.0	18.0	25.0	0.0
32–58	35.8	0.0	32.4	19.6	5.1	3.9	10.1	61.9	0.0

Benders Results: Comparison with MIPs

- All 3 data sets
- Time limit: 30 minutes on each instance

	Benders			MIP-1			MIP-2		
#Tasks	%Gap	%NoSol.	%Opt.	%Gap	%NoSol.	%Opt.	%Gap	%NoSol.	%Opt.
7–21	0.0	0.0	100.0	0.1	0.0	93.5	1.6	0.0	77.8
25–30	13.0	0.0	83.9	13.8	0.0	0.0	18.0	25.0	0.0
32–58	35.8	0.0	32.4	19.6	5.1	3.9	10.1	61.9	0.0

- Solved the small instances in 0.1 seconds on average
- "...more efficient exact methods."

Benders Results: Comparison with MIPs

- All 3 data sets
- Time limit: 30 minutes on each instance

	Benders			MIP-1			MIP-2		
#Tasks	%Gap	%NoSol.	%Opt.	%Gap	%NoSol.	%Opt.	%Gap	%NoSol.	%Opt.
7–21	0.0	0.0	100.0	0.1	0.0	93.5	1.6	0.0	77.8
25–30	13.0	0.0	83.9	13.8	0.0	0.0	18.0	25.0	0.0
32–58	35.8	0.0	32.4	19.6	5.1	3.9	10.1	61.9	0.0

- Solved the small instances in 0.1 seconds on average
- "...more efficient exact methods." Done ✓

Benders Results: Runtime Distribution

- All 3 data sets
- Time limit: 30 minutes on each instance

#Tasks	# Iters.	# Cuts	Master Runtime (s)	SP Runtime(s)
7–21	3	9	0.03 (27.3%)	0.08 (72.7%)
25–30	71	262	269.9 (54.1%)	229.4 (45.9%)
32–58	21	113	832.5 (65.4%)	440.9 (34.6%)

Conclusion

- Proposed a logic-based Benders decomposition framework
  - Together with some Benders cuts

Conclusion

- Proposed a logic-based Benders decomposition framework
  - Together with some Benders cuts
- Optimally solved 257 of the 396 instances in 30 minute time limit
  - Closed 149 open instances

Future Work

Problem
Formulation
Solver
SUALBSP-2
Relaxed Master Problem
Sub-Problem
Mixed-Integer Program
Mixed-Integer Program
Constraint Program
Gurobi, CPLEX, etc.
Chuffed, Gecode, etc.

Future Work

Problem
Formulation
Solver
SUALBSP-2
Relaxed Master Problem
Sub-Problem
Heuristic
Other MIP formulations
Mixed-Integer Program
Mixed-Integer Program
Constraint Program
Travelling Salesperson Program
Gurobi, CPLEX, etc.
Chuffed, Gecode, etc.
Concorde, tsp_solve, etc.

Future Work

Problem
Formulation
Solver
SUALBSP-2
Relaxed Master Problem
Sub-Problem
SUALBSP-1, SUALWABSP, SUALWIBSP, etc.
Heuristic
Other MIP formulations
Mixed-Integer Program
Mixed-Integer Program
Constraint Program
Travelling Salesperson Program
Gurobi, CPLEX, etc.
Chuffed, Gecode, etc.
Concorde, tsp_solve, etc.

Thanks for listening!

Any questions?