Extended Formulations for The Scheduling Problem: Column Generation

Artur Pessoa and Eduardo Uchoa
Engenharia de Produção, Universidade Federal Fluminense, Brazil.
{artur,uchoa}@producao.uff.br.

Marcus Poggi de Aragão
Departamento de Informática, PUC Rio de Janeiro, Brazil.
poggi@inf.puc-rio.br.

Rosiane Rodrigues
COPPE - Sistemas, Universidade Federal do Rio de Janeiro.
rosiane@cos.ufrj.br.

The Scheduling Problem:

- $J = \{1, \dots, n\}$ - set of jobs to be processed
- $M = \{1, \dots, m\}$ - set of parallel identical machines
- p_j - job j positive integral processing time
- $f_j(C_j)$ - cost function over completion time of job j
  C_j - completion time of job j

Find the machines and instants in time for all jobs to start such that:

i No preemption

ii Each machine can process at most one job at a time

iii Machines can stay idle

iv Minimizes $\sum_{j=1}^{n} f_j(C_j)$

Special cases of this cost function:

- $1||\sum w_j T_j$ single machine weighted tardiness
- $P||\sum w_j T_j$ multiple machine weighted tardiness

    d_j - due date of job j
    $T_j = \max\{0, C_j - d_j\}$ - tardiness of job j
    w_j - weight of tardiness of job j

Strongly NP-Hard

Models any cost function based on penalties for

- job earliness or tardiness
- time window for start and/or completion of jobs

    penalties can be infinity

‘

Exact approaches use two distinct kinds of formulations (Queyranne,Schulz 97):

- MIP formulations where job sequence is represented by binary variables and completion times by continuous variables;
- IP time indexed formulations, where the completion time of each job is represented by binary variables indexed over a discretized time horizon

Latter formulations are known to yield better bounds

Pseudo-polynomially large number of variables $\Longrightarrow$ difficulty

- Avella, Boccia and D'Auria (2005):
  - near-optimal solutions (gaps bellow 3%)
  - instances with n up to 400, using Lagrangean relaxation to approximate the time indexed formulation bound
- Pan and Shi (2007):
  - showed that the classical time indexed bound can be exactly computed by solving a cleverly crafted transportation problem.
  - branch-and-bound for the $1||\sum w_j T_j$ that consistently solved all the OR-Library instances with up to 100 jobs
- Bigras, Gamache and Savard (2008):
  - proposed obtaining the same bound by column generation
  - branch-and-price somehow less efficient
  - could not solve some of those instances

The last two algorithms may need to explore large enumeration trees

Time indexed formulation for the single machine scheduling problem.

Dyer, Wolsey 1990

Sousa, Wolsey 1992

Van der Akker et al. 1999,2000

...

- all jobs must be processed in a given time horizon ranging from 0 to T
- binary variables y_j^t indicate that job j starts at time t on some machine

Time-Indexed Formulation

$$\text{Minimize} \sum_{j \in J} \sum_{t=0}^{T-p_j} f_j(t+p_j) y_j^t \qquad (1a)$$

S.t.

$$\sum_{t=0}^{T-p_j} y_j^t = 1 \qquad j \in J \qquad (1b)$$

$$\sum_{\substack{j \in J, \\ t+p_j \leq T}} \sum_{s=\max\{0, t-p_j+1\}}^{t} y_j^s \leq 1 \qquad (t = 0, \ldots, T-1), \qquad (1c)$$

$$y_j^t \in \{0,1\} \quad j \in J; \ t = 0, \ldots, T-p_j \qquad (1d)$$

Parallel machines: right-hand side 1 $\longrightarrow$ m.

Proposed Formulation: Arc-Time-Indexed

- Uses an even larger number of variables:
  - one for each pair of jobs and
  - each possible completion time.
- Also assumes an execution time horizon from 0 to T
- Machines are idle at time 0 and after time T

- Binary variables x_{ij}^t, $i \neq j$,

  indicate that job i completes and job j
  starts at time t on the same machine.
- x_{0j}^t indicate that job j starts at time t in a machine that was idle from time $t-1$ to t
  - in particular, x_{0j}^0 indicate that j starts on some machine at time 0
- x_{i0}^t indicate that job i finishes at time t at a machine that will stay idle from time t to $t+1$
  - in particular, variables x_{i0}^T indicate that i is the last job at a machine
- integral variables x_{00}^t indicate the number of machines that were idle from time $t-1$ to t that will remain idle from time t to $t+1$

  $J_+ = \{0, 1, \ldots, n\}$

Arc-Time-Indexed Formulation

$$\text{Min} \sum_{i \in J_+} \sum_{j \in J \setminus \{i\}} \sum_{t=p_i}^{T-p_j} f_j(t+p_j) x_{ij}^t \tag{2a}$$

$$\sum_{i \in J_+ \setminus \{j\}} \sum_{t=p_i}^{T-p_j} x_{ij}^t = 1 \qquad \forall j \in J \tag{2b}$$

$$\sum_{\substack{j \in J_+ \setminus \{i\} \\ t-p_j \geq 0}} x_{ji}^t - \sum_{\substack{j \in J_+ \setminus \{i\} \\ t+p_i+p_j \leq T}} x_{ij}^{t+p_i} = 0 \qquad \forall i \in J;\ t = 0, \ldots, T-p_i \tag{2c}$$

$$\sum_{\substack{j \in J_+ \\ t-p_j \geq 0}} x_{j0}^t - \sum_{\substack{j \in J_+, \\ t+p_j+1 \leq T}} x_{0j}^{t+1} = 0 \qquad t = 0, \ldots, T-1 \tag{2d}$$

(2e)

$$\sum_{j \in J_+} x_{0j}^0 = m \tag{3a}$$

$$x_{ij}^t \in Z_+ \quad \forall\, i \in J_+;\ \forall j \in J_+ \setminus \{i\} \tag{3b}$$

$$t = p_i, \ldots, T - p_j), \tag{3c}$$

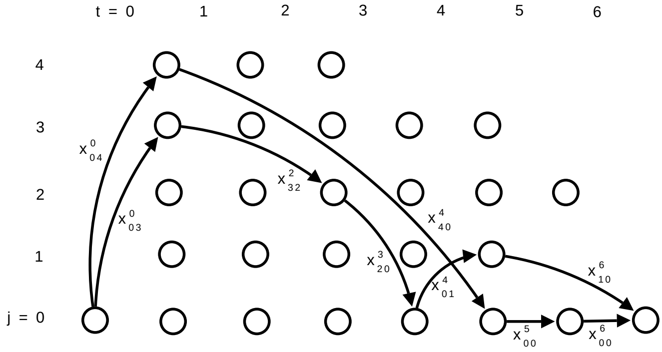
$$x_{00}^t \in Z_+ \qquad t = 0, \ldots, T-1 \tag{3d}$$

. . . and the redundant equation:

$$\sum_{i \in J_+} x_{i0}^T = m \tag{4}$$

It defines a network flow of m units over an acyclic layered graph $G = (V, A)$.

Network m units sent from source to sink
Example: $m = 2,\ n = 4\ p_1 = 2,\ p_2 = 1,\ p_3 = 2,\ p_4 = 4$
$T = 6$



An integral solution of the arc-time indexed formulation:

paths in the layered network.

Proposition

The Arc-Time-Indexed formulation dominates the Time-Indexed formulation.

Proof.

- Let $\bar{x}$ be a linear relaxation solution of Arc-Time-Indexed with cost z. $\bar{x}$ can be converted into $\bar{y}$ a linear relaxation solution of the Time-Indexed formulation with same cost:

  $$\bar{y}_j^t = \sum_{i \in J_+ \setminus \{j\}} \bar{x}_{ij}^t,\ j \in J,\ t = 0, \ldots, T - p_j.$$

- Arc-Time-Indexed formulation can be strictly better than the Time-Indexed formulation: Example

  $1||\sum w_j T_j$ problem where $n = 3$;
  $p_1 = 100, p_2 = 300, p_3 = 200$; $d_1 = 200, d_2 = 300, d_3 = 400$;
  $w_1 = 6, w_2 = 3, w_3 = 2$; and $T = 600$.

$\square$

- If the Arc-Time-Indexed formulation is weakened by adding x_{jj}^t variables:

  it becomes equivalent to the Time-Indexed formulation
  i.e., it is only slightly better

- On the other hand, the Arc-Time-Indexed formulation can be strengthened

Proposition

- *For jobs i and j in J, $i < j$, let x_{ij}^t and $x_{ji}^{t-p_i+p_j}$ be a pair of variables defined in Arc-Time-Indexed and let*
  $\Delta = (f_i(t) + f_j(t + p_j)) - (f_j(t - p_i + p_j) + f_i(t + p_j))$
  - *If $\Delta \geq 0$ variable: x_{ij}^t can be removed*
  - *(Else) If $\Delta < 0$, $x_{ji}^{t-p_i+p_j}$ can be removed.*

- Job i and j are processed consecutively on some machine
- Swap jobs
- Completion times are known for each job in both cases
- Just compare the resulting $f(C_i) + f(C_j)$

A similar reasoning shows that:

Proposition

For job j in J, let x_{j0}^{t} and $x_{0j}^{t-p_j+1}$ be a pair of variables defined in the Arc-Time-Indexed formulation. Let $\Delta = f_j(t) - f_j(t+1)$.

- *If $\Delta > 0$ variable x_{j0}^{t} can be removed*
- *If $\Delta \leq 0$, $x_{0j}^{t-p_j+1}$ can be removed.*

Result of this Preprocessing

Exact 50% of the arcs are removed

- The Arc-Time-Indexed Formulation has none of the eliminated arcs
- The acyclic network $G = (V, A)$ also has none of these arcs in A
- The LP relaxation accepts pseudo-schedules: jobs may repeat (although this preprocessing eliminates many)

New Formulation

To the best of our knowledge, this formulation is NEW.

- Picard and Queyranne (1978): three-index formulation for the $1||\sum w_j T_j$
  - variables x_{ij}^k, meaning that job j follows job i and is the k-th job to be scheduled
  - it has $O(n^3)$ variables and $O(n^2)$ constraints
- Arc-Time-Indexed formulation has $O(n^2 T)$ variables and $O(nT)$ constraints

- Pseudo-polynomially large number of variables and constraints makes the direct use of this formulation prohibitive
- We can rewrite it in terms of variables associated to the pseudo-schedules
- The pseudo-schedules are source-destination paths in $G = (V, A)$

Let P be the set of all source-destination paths in $G = (V, A)$.

- λ_p: 0-1 variable associated to pseudo-schedule p
- q_a^{tp}: 0-1 coefficient indicating if arc a^t is in the path of pseudo-schedule p
- q_a^{tp} is associated to variable x_a^t in the Arc-Time-Indexed formulation
- Define $f_0(t)$ as zero for all t.

- We may write the Explicit Master:

$$\text{Minimize} \quad \sum_{(i,j)^t \in A} f_j(t + p_j) x_{ij}^t \tag{5a}$$

S.t.

$$\sum_{p \in P} q_a^{tp} \lambda_p - x_a^t = 0 \quad (\forall\, a^t \in A), \tag{5b}$$

$$\sum_{(j,i)^t \in A} x_{ji}^t = 1 \quad (\forall\, i \in J), \tag{5c}$$

$$\sum_{(0,j)^0 \in A} x_{0j}^0 = m \tag{5d}$$

$$\lambda_p \geq 0 \quad (\forall\, p \in P), \tag{5e}$$

$$x_a^t \in Z_+ \quad (\forall\, a^t \in A). \tag{5f}$$

We eliminate the x variables and relax integrality to obtain the Dantzig-Wolfe Master (DWM) LP:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{p \in P} \Big(\sum_{(i,j)^t \in A} q_{ij}^{tp} f_j(t+p_j) \Big) \lambda_p && \text{(6a)} \\
\text{S.t.} \quad & && \\
& \sum_{p \in P} \Big(\sum_{(j,i)^t \in A} q_{ji}^{tp} \Big) \lambda_p = 1 \qquad (\forall\, i \in J), && \text{(6b)} \\
& \sum_{p \in P} \Big(\textstyle\sum_{(0,j)^0 \in A} q_{0j}^{0p} \Big) \lambda_p = m && \text{(6c)} \\
& \lambda_p \geq 0 \qquad (\forall\, p \in P). && \text{(6d)}
\end{aligned}
$$

- $\sum_{(0,j)^0 \in A} q_{0j}^{0p} = 1$ for any $p \in P$

- Cuts on the x variables, $\sum_{a^t \in A} \alpha_{al}^t x_a^t \geq b_l$
  take the form $\sum_{p=1}^{P} (\sum_{a^t \in A} \alpha_{al}^t q_a^{tp}) \lambda_p \geq b_l$ in the DWM.
- Suppose we have r constraints where α is the coefficients for it in the x format.
- π are the dual variables associated to these constraints

The reduced cost of an arc $a^t = (i,j)^t$ is then:

$$\bar{c}_a^t = f_j(t + p_j) - \sum_{l=0}^{r} \alpha_{al}^t \pi_l. \tag{7}$$

We separated the Extended Capacity Cuts:

- Generic family of cuts (Uchoa 2005)
- Effective on the capacitated minimum spanning tree (Uchoa et al. 2008)
- Also effective on many vehicle routing problem variants (Pessoa et al. 2008)

- Cuts over the time-indexed variables are derived.
- For each vertex $i \in V_+$ the following balance equation is valid:

$$\sum_{a^t \in \delta^-(i)} t x_a^t - \sum_{a^t \in \delta^+(i)} t x_a^t = p_i \quad . \tag{8}$$

- Let $S \subseteq V_+$ be a set of vertices. Summing the equalities corresponding to each $i \in S$, we get the *time-balance equation over S*:

Definition

An *Extended Capacity Cut* (ECC) over S is any inequality valid for $P(S)$, the polyhedron given by the convex hull of the 0-1 solutions of

$$\sum_{a^t \in \delta^-(S)} t x_a^t - \sum_{a^t \in \delta^+(S)} t x_a^t = p(S)$$

- It can be noted that those equations are always satisfied by the solutions (DWM) (translated to the x^t space by $\sum_{j=1}^{p} q_a^{tj} \lambda_j - x_a^t$)

- HECCs: aggregated variables v^t and z^t

$$v^t = \sum_{a^t \in \delta^+(S)} x_a^t \qquad (t = 1, \ldots, T), \tag{9}$$

$$z^t = \sum_{a^t \in \delta^-(S)} x_a^t \qquad (t = 1, \ldots, T). \tag{10}$$

The balance equation over those variables is:

$$\sum_{t=1}^{T} t v^t - \sum_{t=1}^{T} t z^t = p(S) \quad . \tag{11}$$

For each possible pair of values of T and $D = p(S)$, a polyhedron $P(T, D)$

- HECCs are facets of $P(T, D)$

Pricing subproblem:

- Shortest path in the acyclic network $G = (V, A)$
  - Takes $\Theta(|A|)$
  - $|A| = \Theta(n^2 T)$
  - $T = \Omega(np_{avg}/m)$, where p_{avg} is the average job processing time

Time consuming:

For $m = 1$, $n = 100$ and $p_{avg} = 50$, $|A|$ is more than 25 million.

i extreme degeneracy, in fact, when $m = 1$ it can happen that any optimal basis has just one variable with a positive value

ii extreme variable symmetry, in the sense that there is usually many alternative solutions with the same cost

iii an expensive pricing with complexity $\Omega(n^3 p_{avg}/m)$, where p_{avg} is the average job processing time.

Lagrangean subproblem:

$$L(\pi) = \text{Min} \quad \sum_{a^t \in A} \bar{c}_a^t x_a^t + \sum_{l=0}^{r} b_l \pi_l \qquad (12a)$$

$$\text{S.t.}$$

$$\sum_{p \in P} q_a^{tp} \lambda_p - x_a^t = 0 \quad (\forall\, a^t \in A), \qquad (12b)$$

$$\sum_{(0,j)^0 \in A} x_{0j}^0 = m \qquad (12c)$$

$$\lambda_p \geq 0 \quad (\forall\, p \in P), \qquad (12d)$$

$$x_a^t \in Z_+ \quad (\forall\, a^t \in A). \qquad (12e)$$

- For each possible π, an optimal solution can be constructed by setting $\lambda_{p^*} = m$ - p^*: path of minimum reduced cost,
- all other λ variables are set to zero,
- the x variables are set in order to satisfy (12b).

- Let $L(\pi, a, t)$ be the solution of the Lagrangean problem with the addtional constraint $x_a^t \geq 1$
  - a^t can be eliminated if $L(\pi, a, t) \geq Z_{INC}$ - Z_{INC} best known solution
- $L(\pi, a, t)$ can be computed by obtaining the shortest path labels forward and backward, and finally adding a^t reduced cost to its extremeties' labels
- Amounts to have a pricing 3 times slower

This fixing procedure performed extremely well.

Stabilization

- Presented by Eduardo Uchoa yesterday
- Convex combination of Lagrangean dual and DWM's current simplex multipliers
- One parameter
- Misprice
- Either dual or primal improvement: exponential convergence
- Hot start with the Volume algorithm

Branch-cut-and-price (robust)

- Primal heuristics
- Root bounds are strong
- Branching on original variables x_a^t
- Switch to Branch-and-cut when the number x_a^t is small

Instances for $1||\sum w_j T_j$

- Experiments taken on the set of 375 instances of the OR Library
- Generated by Potts and Wassenhove (1985) and contains 125 instances for each $n \in \{40, 50, 100\}$.
- Same set used by Pan and Shi (2007) and Bigras, Gamache and Savard (2008)

Instances for $P||\sum w_j T_j$

- We derived 100 new instances from those in the OR-Library
- For $m \in \{2, 4\}, n\{40, 50\}$
  - We pick the first $1||\sum w_j T_j$ instance in each group (the one ending in 1 or 6)
  - Divided each due date d_j by m (and rounded down)
  - Processing times p_j and weights w_j were kept unchanged

All our experiments were performed in a notebook with processor Intel Core Duo (but using a single core) with a clock of 1.66GHz and 2GB of RAM. The linear program solver was CPLEX 11

Table: Comparison of the complete BCP algorithm with the best algorithm by Pan and Shi

n	*Alg.*	*Avg T(s)*	*Max T(s)*	*Avg. Nd*	*Max Nd*	*Root Gap %*
40	PS	69.0	235	141	293	0.68
	BCP	12.1	43.6	1	1	0
50	PS	142.8	232	416	5623	0.74
	BCP	28.1	123.8	1	1	0
100	PS	1811	32400	18877	> 909844	0.52
	BCP	648.5	8508	2.03	42	0.0013

Table: Detailed results of the complete BCP algorithm over a sample of 25 OR-Library instances with $n = 100$.

	Volume				1st.LP				Remain Root		
Inst	*LB*	*Iter*	*Time*	*R.Arcs*	*LB*	*Iter*	*Time*	*R.Arcs*	*LB*	*CutR*	*Time*
16	407703	160	279.9	0	—	—	—	—	—	—	—
21	898925	125	212.7	0	—	—	—	—	—	—	—
26	8	1	69.4	0	—	—	—	—	—	—	—
31	24202	20	90.2	0	—	—	—	—	—	—	—
36	108293	93	209.3	0	—	—	—	—	—	—	—
41	462117	401	922.7	13093	462123	227	120.1	10596	462324	1	24.3
46	829771	340	498.0	1583	829773	115	67.5	1375	829828	1	14.6
51	—	—	—	—	—	—	—	—	—	—	—
56	9046	20	76.6	0	—	—	—	—	—	—	—
61	86793	227	480.4	0	—	—	—	—	—	—	—
66	243637	555	1164.7	33525	243644	382	119.3	30002	243822	9	1088.1
71	640799	351	735.1	1178	640802	113	103.2	775	640816	1	16.0
76	—	—	—	—	—	—	—	—	—	—	—
81	1400	30	80.2	0	—	—	—	—	—	—	—
86	66850	186	463.5	0	—	—	—	—	—	—	—
91	248284	401	1027.7	77260	248293	428	152.6	64425	248699	4	3089.8
96	495358	376	918.7	18853	495362	275	104.7	15994	495516	2	50.0
101	—	—	—	—	—	—	—	—	—	—	—
106	—	—	—	—	—	—	—	—	—	—	—
111	158962	454	1554.2	29457	158968	337	143.6	25652	159123	2	1369.2
116	370435	445	1288.2	40265	370451	354	176.4	34754	370614	2	1250.3
121	471166	392	957.4	4024	471175	235	144.2	2626	471214	1	15.6

Table: Comparison of different bounding methods for multi-machine instances.

		Time indexed			Arc-Time			Arc-Time + Cuts		
n	m	*Av Gap %*	*M Gap%*	*T(s)*	*Av Gap %*	*M Gap%*	*T(s)*	*Av Gap %*	*M Gap%*	*T(s)*
40	2	1.533	21.016	85.6	1.243	20.840	32.2	0.053	0.853	295.9
	4	0.544	4.787	32.2	0.406	3.390	14.1	0.105	0.841	63.9
50	2	0.535	4.074	182.2	0.487	4.074	88.3	0.078	1.051	2298.7
	4	0.529	5.614	79.5	0.489	5.614	36.8	0.266	5.088	262.5

Table: Detailed results of the complete BCP algorithm over the instances with $m = 2$ and $n = 40$.

	1st.LP				Remaining Root Node				Total		
Inst	*LB*	*Iter*	*Time*	*R.Arcs*	*LB*	*CutR*	*Time*	*R.Arcs*	*Nd*	*Time*	*Opt*
1	584	89	18.0	156948	606	7	325.4	0	1	343.4	606
6	3875	141	25.5	82838	3886	3	119.6	0	1	145.1	3886
11	9592	189	34.5	66999	9617	2	94.6	0	1	129.1	9617
16	38279	292	45.2	59225	38351	2	515.8	59225	3	561.0	38356
21	41048	384	37.1	0	—	—	—	—	1	37.1	41048
26	87	48	12.7	0	—	—	—	—	1	12.7	87
31	3758	172	34.0	106263	3812	5	452.0	0	1	486.0	3812
36	10662	303	44.4	52812	10700	2	1113.6	52812	5	1193.6	10713
41	30802	387	46.4	0	—	—	—	—	1	46.4	30802
46	34146	430	29.8	0	—	—	—	—	1	29.8	34146
51	—	—	—	—	—	—	—	—	—	0	0
56	1272	80	16.5	107098	1279	2	72.3	0	1	88.8	1279
61	11311	269	45.3	72238	11390	2	1754.2	72238	327	9097.3	11488
66	35130	323	51.9	75499	35196	2	1503.6	75499	196	6451.1	35279
71	47935	423	42.9	42430	47952	2	19.5	0	1	62.4	47952
76	—	—	—	—	—	—	—	—	—	0	0
81	452	150	20.6	71423	571	2	947.2	0	1	967.8	571
86	5996	302	40.4	47829	6041	2	253.5	47829	6	298.0	6048
91	26075	388	56.6	0	—	—	—	—	1	56.6	26075
96	66110	358	50.9	46481	66116	2	2.9	0	1	53.8	66116
101	—	—	—	—	—	—	—	—	—	0	0
106	—	—	—	—	—	—	—	—	—	0	0
111	17898	292	50.0	51884	17936	2	46.5	0	1	96.5	17936
116	25786	317	50.4	54574	25870	2	173.7	0	1	224.1	25870
121	64507	390	50.9	48152	64516	2	3.0	0	1	53.9	64516

Comments

- At the end of this first column generation step, besides having a bound close to the optimal, the number of non-fixed variables is usually quite small. Switching to branch-and-cut was an alternative
- In almost all $1||\sum w_j T_j$ benchmark instances from the OR-Library, with $n \in \{40, 50, 100\}$, we found that the duality gaps were reduced to zero still in the root node.
- The same algorithm was also tested on the $P||\sum w_j T_j$, a harder problem. We do not know any paper claiming optimal solutions on instances of significant size.
- Our branch-cut-and-price could solve instances derived from those in the OR-Library, with $m \in \{2, 4\}$ and $n \in \{40, 50\}$, consistently. However, the solution of several such multi-machine instances did required a significant amount of branching.

Thank you!

Merci Guy, Jacques and Marco !