An Energy Cost Aware Cumulative

Helmut Simonis and Tarik Hadzic

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

Modref 2010, St. Andrews

science foundation ireland
fondúireacht eolaíochta éireann

Cork Constraint Computation Centre

What You Need to Remember

- Propose extension of `Cumulative`
- `CumulativeCost` to handle time and volume dependent resource cost
- Use Case 1: Electricity costs based on new tariffs
- Use Case 2: Manpower scheduling
- Compare several lower bounds
- Best results obtained with LP model based on Hooker

Outline

1. Motivation
2. Lower Bounds
3. Results

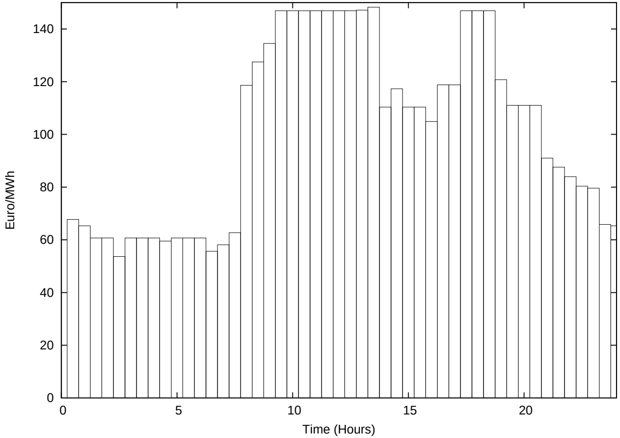
Background

- Funded by Science Foundation Ireland (SFI)
- TIDA project: Application oriented research
- Cooperation with IBM, United Technologies
- Aim: Develop scheduling tools for energy cost efficient scheduling

Use Case 1: Electricity Tariffs

- Wholesale electricity price changes with demand
- Peak demand requires inefficient(=expensive) generation plant
- Off-peak price low due to baseload (always on) plant
- End users' tariffs can follow the wholesale price
- Question: How to react to changing electricity cost?

Irish Electricity Price (Source: `http://allislandmarket.com/`)



Extension of Use Case 1: Volume Dependent Cost

- Many plants use co-generation
- Or limited renewable source like windpower
- This is cheaper than grid electricity, but limited in volume
- Cost changes with volume and time
- Assumption: Extra energy is always more expensive

Use Case 2: Manpower Cost

- Scheduling problems with manpower costs
- Man/hour cost varies over time
- Office Hours/Nights/Weekends/Holidays
- Extra staff costs more: Temps/Freelance
- Natural, otherwise change hire rules

Reminder: `Cumulative`

- Aggoun, Beldiceanu 1993
- Core global constraint for constraint-based scheduling
- Large number of algorithmic developments, few changes of basic constraint
- Time/volume dependent resource cost not considered so far

$$\texttt{Cumulative}([s_1, s_2, ...s_n], [d_1, d_2, ...d_n], [r_1, r_2, ...r_n], l, p),$$

New Constraint Variant: `CumulativeCost`

- Add cost element
- Per unit cost expressed with areas
- Intersection of resource use profile with areas defines cost
- Global reasoning required

Formally: `CumulativeCost`

$$\forall\, 0 \le t < p : \quad pr_t := \sum_{\{i | s_i \le t < s_i + d_i\}} r_i \le l$$

$$\forall\, 1 \le i \le n : \quad 0 \le s_i < s_i + d_i \le p$$

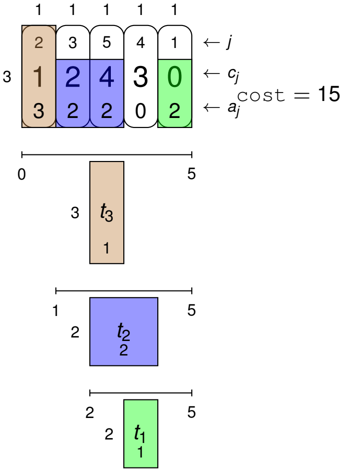
$$\mathtt{ov}(t, pr_t, A_j) := \begin{cases} \max(0, \min(y_j + h_j, pr_t) - y_j) & x_j \le t < x_j + w_j \\ 0 & \textit{otherwise} \end{cases}$$

$$\forall\, 1 \le j \le m : \quad a_j = \sum_{0 \le t < p} \mathtt{ov}(t, pr_t, A_j)$$

$$\mathtt{cost} = \sum_{j=1}^{m} a_j c_j$$

Cork Constraint Computation Centre

Running Example



Outline

Fundamental Question: How to Estimate Cost?

- We already have core cumulative constraints (15 years of algorithm development)
- Need good lower bound estimate to prune search
- Also want to use cost information to restrict task start times
- Need to answer cost estimate choice before writing pruning methods

Strategy: Explore Possible Lower Bounds

- Decomposition with `Cumulative`
  - Based on `Element`
  - Flow Model and Extensions
  - Greedy Methods
- Direct LP Formulation Based on Hooker

`Element` Model

- For each task, consider all possible start times
- For each start time, estimate cost for this task
- Ignore interaction of tasks, capacity limits
- Total cost estimate: take cheapest estimate for each task

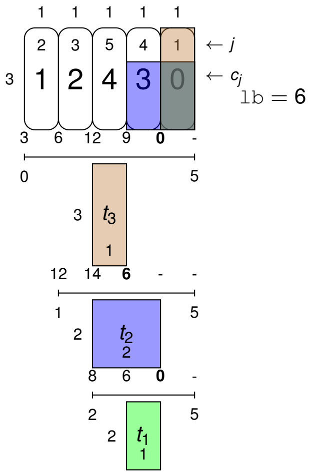
`Element` Model

$$\texttt{cumulative}([s_1, s_2, ...s_n], [d_1, d_2, ...d_n], [r_1, r_2, ...r_n], l, p),$$

$$\texttt{lb} = \min \sum_{i=1}^{n} u_i$$

$$\forall\ 1 \leq i \leq n : \quad \texttt{element}(s_i, [v_{i1}, v_{i2}, ..., v_{ip}], u_i)$$

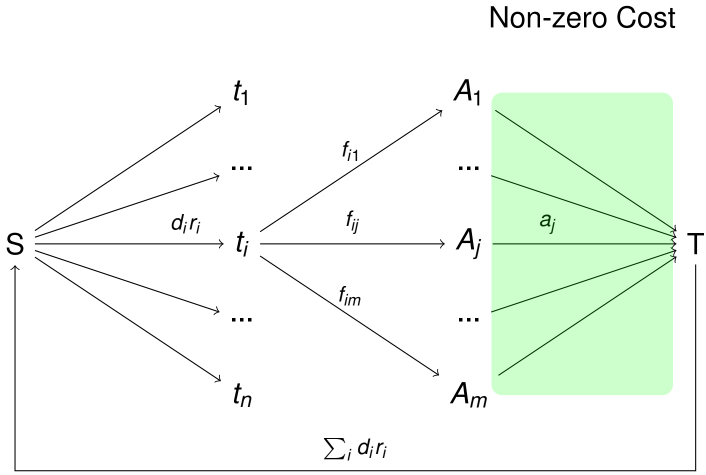
Running Example: `Element` Model



Flow Model

- Many global constraints are based on flow models
- Consider flow from tasks to areas
- Good view of capacity of areas
- Allows to split tasks into cheapest areas

Min Cost Flow Model



Flow Equations

$$\texttt{lb} = \min \sum_{j=1}^{m} a_j c_j$$

$$\forall\, 1 \le j \le m: \quad a_j = \sum_{i=1}^{n} f_{ij}$$

$$\forall\, 1 \le i \le n, \forall\, 1 \le j \le m: \quad \underline{f_{ij}} \le f_{ij} \le \overline{f_{ij}}$$

$$\forall\, 1 \le j \le m: \quad 0 \le \underline{a_j} \le a_j \le \overline{a_j} \le w_j h_j$$

$$\forall\, 1 \le i \le n: \quad \sum_{j=1}^{m} f_{ij} = d_i r_i$$

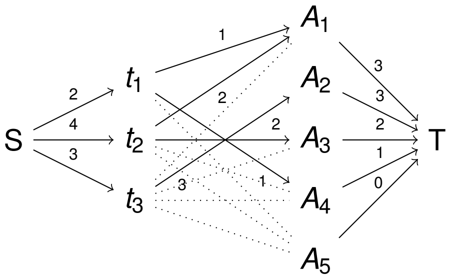
$$\forall\, 1 \le i \le n: \quad \sum_{i=1}^{n} d_i r_i = \sum_{j=1}^{m} a_j$$

Computing $\overline{f_{ij}}$

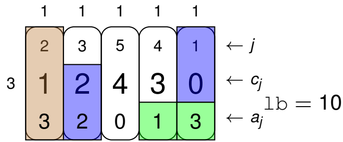
$$\overline{f_{ij}} = \max_{t \in d(s_i)} \max(0, (\min(x_j + w_j, t + d_i) - \max(x_j, t))) * \min(h_j, r_i)$$

Flow Example

Flow Graph



Assignment



Extensions to Flow Model

- Often: If task uses one cheap area, it can not also use other cheap areas
- Consider how much of a task can be placed in cheap areas
- b_{ij} how much of task i can be placed into all areas $1..j$
- Add this as a constraint to flow model
- Results in general LP model

Extending Flow Model: LP 1 Model

- Add one constraint for each increasing group of areas
- Aggregate information from multiple tasks, less precise

$$\forall\ 1 \leq j \leq m: \quad \sum_{i=1}^{n} \sum_{k=1}^{j} f_{ik} = \sum_{k=1}^{j} a_k \leq \overline{B_j} = \sum_{i=1}^{n} \overline{b_{ij}}$$

Cork Constraint Computation Centre

f_{ij} and b_{ij}

f_{ij}	1	2	3	4	5
1	2	0	0	2	2
2	2	0	2	2	2
3	3	3	3	3	3

b_{ij}	1	2	3	4	5
1	2	2	2	2	2
2	2	2	2	4	4
3	3	3	3	3	3
B_j	7	7	7	9	9

Extending Flow Model: LP 2 Model

- State constraint for each task, and increasing groups of areas
- Improved accuracy
- Larger number of constraints

$$\forall\ 1 \leq i \leq n, \forall\ 1 \leq j \leq m : \quad \sum_{k=1}^{j} f_{ik} \leq \overline{b_{ij}}$$

This is Getting Expensive!

- Solving LP at each step is quite expensive
- Can we obtain similar bounds more cheaply?
- Greedy methods
- Fill areas with tasks, starting with cheapest area
- Compute how much can go into each area

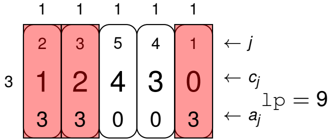
Algorithm A

$$\texttt{lb} = \sum_{j=1}^{m} u_j c_j$$

$$\forall\ 1 \leq j \leq m: \quad u_j = \min(\sum_{i=1}^{n} d_i r_i - \sum_{k=1}^{j-1} u_k, \sum_{i=1}^{n} \overline{f_{ij}}, w_j h_j)$$

Example Run: Algorithm A

u_j	rem	$\sum_{i=1}^{n} \overline{f_{ij}}$	$w_j h_j$	lb
3	9	7	3	0
3	6	7	3	3
3	3	5	3	9
0	0	-	-	9
0	0	-	-	9



Extension

- Also consider $\overline{b_{ij}}$ limits in greedy method
- Easy to add
- Requires computation of $\overline{b_{ij}}$ at each step

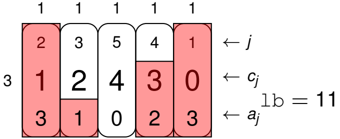
Algorithm B

$$\texttt{lb} = \sum_{j=1}^{m} u_j c_j$$

$$\forall\, 1 \leq j \leq m: \quad u_j = \min\left(\sum_{i=1}^{n} d_i r_i - \sum_{k=1}^{j-1} u_k, \sum_{i=1}^{n} \overline{f_{ij}}, w_j h_j, \sum_{i=1}^{n} \overline{b_{ij}} - \sum_{k=1}^{j-1} u_k\right)$$

Example Run: Algorithm B

u_j	rem	$\sum_{i=1}^{n} \overline{f_{ij}}$	$w_j h_j$	$\sum_{i=1}^{n} \overline{b_{ij}} - \sum_{k=1}^{j-1} u_k$	lb
3	9	7	3	7	0
3	6	7	3	7-3	3
1	3	5	3	7-6	5
2	2	7	3	9-7	11
0	0	-	-	-	11



Direct Model

- Express cumulative and cost consideration as one linear model
- Based on `Cumulative` relaxation by Hooker
- y_{it} 0/1 variable, task i starts at time t
- Enforcing integrality leads to MIP
- Possibly very expensive due to number of time points
- Integrate with finite domain `Cumulative` for best results

Direct LP

$$\texttt{lb} = \min \sum_{j=1}^{m} a_j c_j$$

$$pr_t \in [0, l],\ y_{it} \in \{0, 1\},\ z_{jt} \in [0, h_j]$$

$$\forall\, 1 \le j \le m : \quad 0 \le \underline{a_j} \le a_j \le \overline{a_j} \le w_j h_j$$

$$\forall\, 1 \le i \le n : \quad s_i = \sum_{t=0}^{p-1} t y_{it}$$

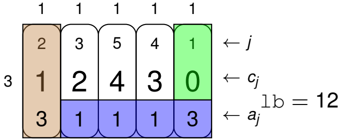
$$\forall\, 1 \le i \le n : \quad \sum_{t=0}^{p-1} y_{it} = 1$$

$$\forall\, 0 \le t < p : \quad pr_t = \sum_{t' \le t < t' + d_i} y_{it'} r_i = \sum_{j=1}^{m} z_{jt}$$

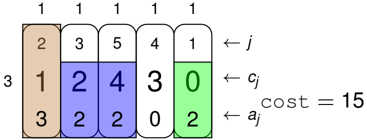
$$\forall\, 1 \le j \le m : \quad a_j = \sum_{t=x_j}^{x_j + w_j - 1} z_{jt}$$

Example for Direct Models

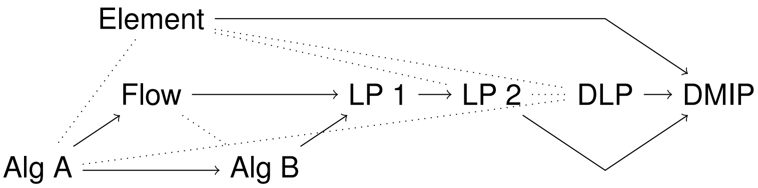
Direct LP



Direct MIP



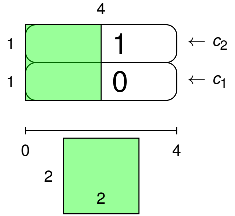
Theorem: Comparative Power of Algorithms



Example: Element and Flow Models stronger than DLP

lb(Element)=2 $>$ lb(DLP)=0

lb(Alg A)=2 $>$ lb(DLP)=0



Model Comparison

Method	Single Area Capacity	Multi Area Capacity	Earliest Start Latest End	Task Profile
Element	no	no	yes	yes
Flow	yes	no	yes	$\leq$ height
LP 1	yes	$\overline{B_j}$	yes	$\leq$ height
LP 2	yes	$\overline{b_{ij}}$	yes	$\leq$ height
Alg A	yes	no	no	no
Alg B	yes	$\overline{B_j}$	no	no
DLP	yes	yes	yes	width
DMIP	yes	yes	yes	yes

Outline

Experiments

- Done with Choco, CPLEX 12.1
- First set, 100 tasks each
- Evaluation of algorithms for $d_{max} = r_{max} = 8$ and $\Delta = 10$
- Vary utilization between 30% and 80%
- Try with cost function shown above and random cost
- Test each lower bound and compare to DMIP solution
- DMIP can be expensive for high utilization (max. 8,121 sec)

Experiments: 100 Tasks, fixed and random cost profiles

Scenario	Element		A		B		Flow		LP1		LP2		DLP	
util=30	92	92	0	0	0	0	0	0	0	0	0	0	93	100
fixed	99.998	99.876	57.268	33.055	99.77	99.337	97.429	94.665	99.77	99.337	99.77	99.337	99.999	99.994
	185	509	8	73	12	126	34	138	150	277	211	617	111	380
util=50	2	2	0	0	0	0	0	0	0	0	0	0	7	100
fixed	99.038	94.641	68.789	54.22	99.131	95.963	97.816	95.89	99.407	97.773	99.435	97.913	99.948	99.358
	176	243	6	12	7	94	33	130	139	274	194	275	96	181
util=70	0	0	0	0	0	0	0	1	0	5	0	6	1	100
fixed	93.541	81.603	84.572	69.953	96.495	87.884	99.1	96.994	99.24	97.838	99.346	98.071	99.764	98.992
	177	242	7	103	8	72	34	97	136	239	213	1,798	110	1,551
util=80	0	0	0	0	0	1	0	4	0	20	0	21	0	100
fixed	88.561	70.901	92.633	81.302	96.163	89.437	99.34	96.728	99.354	96.737	99.392	96.737	99.649	98.528
	206	450	10	67	15	96	38	124	156	235	220	426	125	299
util=30	94	94	0	0	0	0	0	0	0	0	0	0	97	100
random	99.996	99.872	58.094	41.953	96.965	93.237	73.759	54.641	96.965	93.237	96.966	93.254	99.999	99.977
	192	427	7	24	8	42	32	94	145	224	203	361	99	274
util=50	0	0	2	8	2	8	2	8	2	8	2	8	5	100
random	88.277	30.379	76.457	57.049	96.585	92.563	83.314	69.178	96.619	92.604	96.861	93.242	99.93	99.724
	202	814	10	99	13	131	43	177	165	380	238	903	108	327
util=70	0	0	0	0	0	0	0	0	0	0	0	0	0	100
random	91.045	72.06	89.784	75.822	95.242	90.496	92.953	84.277	95.953	92.24	96.947	94.012	99.697	99.374
	226	436	13	74	26	98	70	178	223	543	280	566	152	428
util=80	0	0	0	0	0	0	0	0	0	0	0	0	0	100
random	86.377	72.566	94.813	88.039	96.092	89.233	97.231	93.919	97.658	94.917	98.426	96.342	99.626	99.161
	320	2,148	16	100	31	180	63	370	223	1,586	363	23,91	286	7,242

Algorithm Comparison

- Around 2000 instances
- For each pair of algorithms, count number of winners
- Also count average and max improvement
- Columns marked (-), winning not possible due to ordering
- Empty columns, no winning scenario found (but possible)

Pairwise Comparison

	Element			A			B			Flow			LP1			LP2			DLP		
DMIP	1802	26.39	88.62	1944	17.47	68.55	1944	3.99	30.71	1944	9.85	68.55	1944	3.49	30.71	1944	3.24	30.71	1387	0.18	1.47
EL	-	-	-	1034	25.62	66.95	656	3.0	22.07	856	15.65	65.82	650	3.01	22.07	621	3.04	22.07			
A	1052	38.1	88.61	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			
B	1429	29.22	88.61	1439	18.22	66.65	-	-	-	1107	11.02	51.86	-	-	-	-	-	-			
FLW	1230	33.97	88.61	1184	12.51	64.35	333	2.39	10.49	-	-	-	-	-	-	-	-	-			
LP1	1436	29.74	88.61	1441	18.86	66.65	726	1.33	10.51	1413	8.75	51.86	-	-	-	-	-	-			
LP2	1465	29.44	88.61	1441	19.19	66.65	846	1.71	10.64	1425	9.02	51.86	690	0.7	5.09	-	-	-			
DLP	1802	26.24	88.61	1752	19.24	68.55	1751	4.28	30.71	1747	10.82	68.55	1727	3.78	30.71	1725	3.51	30.71	-	-	-

Impact of Number of Tasks

- Tested between 50 and 400 tasks
- Does not seem to be limiting factor

Increasing Number of Tasks

Scenario	Element		A		B		Flow		LP1		LP2		DLP	
n=50	0	0	0	0	0	0	0	0	0	0	0	0	0	100
	89.331	71.865	88.833	70.372	93.991	86.143	92.384	84.589	94.863	87.417	95.899	89.937	98.664	95.523
	102	314	8	83	6	36	23	78	93	254	132	298	75	233
n=100	0	0	0	0	0	0	0	0	0	0	0	0	0	100
	91.045	72.06	89.784	75.822	95.242	90.496	92.953	84.277	95.953	92.24	96.947	94.012	99.697	99.374
	226	436	13	74	26	98	70	178	223	543	280	566	152	428
n=200	0	0	0	0	0	0	0	0	0	0	0	0	0	100
	92.537	84.06	89.819	79.862	95.885	93.01	92.566	83.516	96.48	93.328	97.158	93.885	99.936	99.833
	395	700	19	148	22	113	83	208	341	533	468	638	226	456
n=400	0	0	0	0	0	0	0	0	0	0	0	0	0	100
	93.239	86.419	90.417	84.205	96.3	92.721	92.939	86.658	96.716	93.275	97.23	95.013	99.985	99.961
	831	1,222	31	164	36	189	181	305	923	3,053	1,214	3,434	484	871

Evaluation

- DLP is clear winner
  - Much more consistent than other models
  - But quite expensive
- LP1, LP2 too expensive
- Flow, Alg. A too weak
- `Element` only good for low utilization
- Alg. B: good value for money

Future Work

- Domain Pruning based on DLP
  - Fairy straightforward reduced-cost filtering
  - Removes many values inside domains
- Pruning based on Algorithm B
- Classifier for algorithm selection
- Specific search strategies
- Talk this afternoon in CROCS workshop!

Cork Constraint Computation Centre