Submodular Interval Scheduling

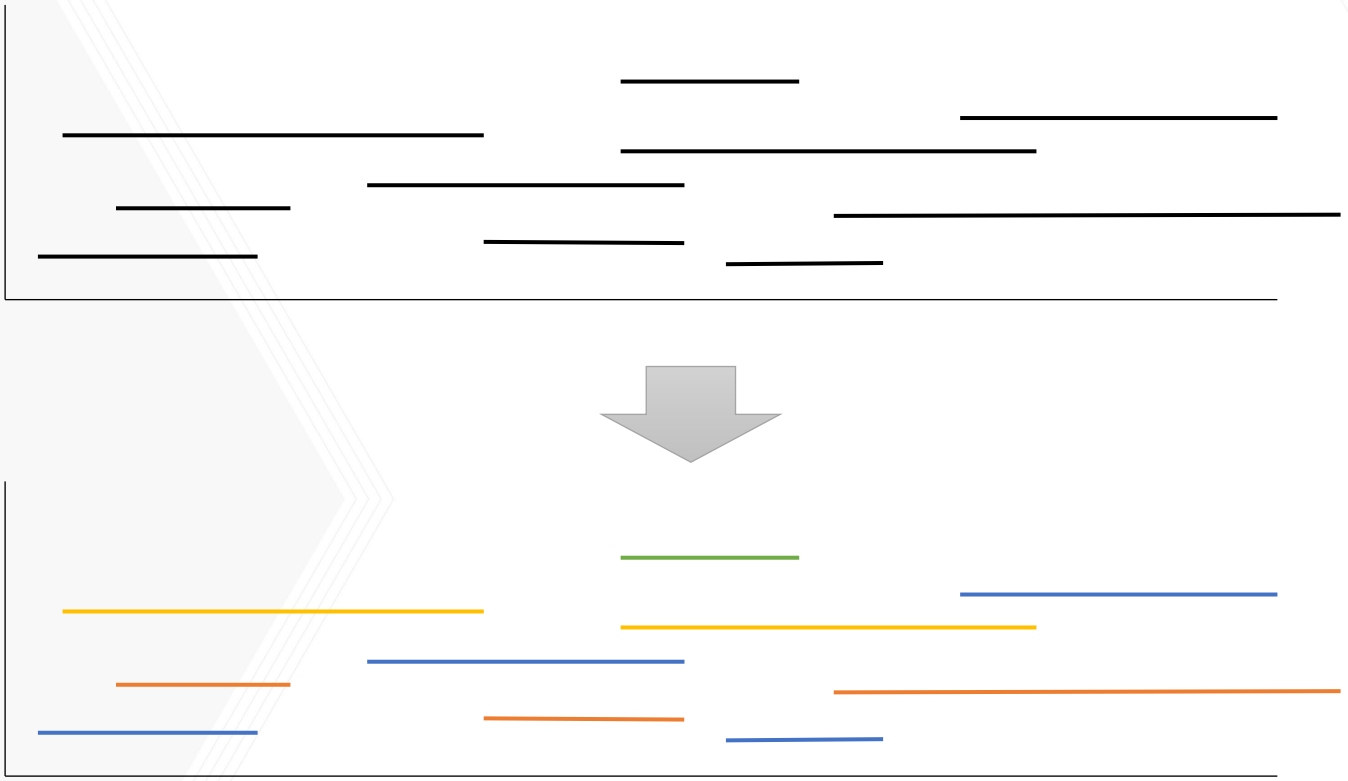
INFORMS Computing Society Conference

Tampa, FL January 2022

Alejandro Toriello

Joint with Chris Muir

Submodular Interval Scheduling



$$cost = f(Y) + f(B) + f(O) + f(G)$$

Submodular Interval Scheduling

- **Input:** Set J of n jobs and a function $f: 2^J \to \mathbb{R}_+$
  - Job j: start time s_j, end time $e_j > s_j$, and weight $w_j > 0$
  - Monotone: $f(S) \leq f(T), S \subseteq T$
  - Submodular: $f(S \cap T) + f(S \cup T) \leq f(S) + f(T)$
  - Weight-defined: $f(S) = g(w_j, j \in S)$

- **Output:** Partition $S_1, S_2, \ldots, S_k$ of J minimizing $\sum f(S_i)$
  - S_i feasible if $[s_j, e_j] \cap [s_k, e_k] = \emptyset \ \forall j, k \in S_i$

- **Applications:** crew scheduling, channel assignment, airline gate assignment, cloud scheduling

Related Work

- **Previous work**
  - Submodular coloring
  - Fukunaga et al 2007: 4-approximation for concave functions,
    - Prove NP-Hard for strictly concave functions
  - Correa et al 2015: 5-approximation for value monotone-submodular functions
  - Escoffier et al 2006: Special case max-weight function
    - Prove NP-Hard
  - Mehrotra et al 1996: Set covering model for graph coloring

- **Our contributions**
  - Propose set cover formulation
  - Exact branch-and-price algorithm for max-weight and concave cost functions
    - Max-weight: $f(S) = \max_{i \in S} w_i$
    - Concave: $f(S) = h(\sum_{i \in S} w_i)$, h monotone, non-negative, concave

Set Cover Formulation

$$\min \sum_{S \in \mathcal{S}} f(S) z_S$$

$$s.t. \sum_{S \in \mathcal{S}(i)} z_S \geq 1 \qquad \forall i \in J$$

$$z_S \geq 0 \qquad \forall S \in \mathcal{S}$$

- $\mathcal{S}$: set of feasible schedules
- $\mathcal{S}(i)$: set of schedules containing i
- $O(2^{|J|})$ variables, solve with column generation

Pricing Problem

$$\max_{S \in \mathcal{S}} \left\{ \sum_{i \in S} \pi_i - f(S) \right\}$$

- π: current dual prices
- Interval packing with supermodular maximization objective

Pricing: Max-Weight Function

- Pricing problem: $\max_{S \in \mathcal{S}} \left\{ \sum_{i \in S} \pi_i - \max_{i \in S} w_i \right\}$
- For fixed value of $\max_{i \in S} w_i = \bar{w}$,

  simplifies to an ***interval packing*** problem

  $$\max_{S \in \mathcal{S}} \sum_{i \in S : w_i \leq \bar{w}} \pi_i$$

- Interval Packing DP: $v_i = \max\{\pi_i + v_{\eta_i}, v_{i+1}\} \ \forall i \in J : w_i \leq \bar{w}$
  - $\eta_i = \min\{j : j > i, \ s_j \geq e_i, w_j \leq \bar{w}\}$
  - $\Theta(n)$ running time
- Similar DP with pairs has $O(n^2)$ running time
- **Pricing concave function**: $O(n \sum_{i \in J} w_i)$ or $O(n^2 \sum_{i \in J} w_i)$ running time

Branch-and-Price

- Branching: i, j scheduled consecutively in some schedule
  - $\theta_{i,j} = \sum_{S \in \mathcal{S}} \mathbb{I}_{(i,j) \in S} z_S$
- Modified DP, select what job follows next in schedule
  - Max-weight function: $O(n^3)$ complexity
  - Concave functions: $O(n^2 \sum_{i \in J} w_i)$ complexity
- Implementation Details
  - Branch on most fractional $\theta_{i,j}$
  - Iterative depth-first exploration, explore $\theta_{i,j} = 1$ branches first
  - Initial solution: greedy heuristic + local search

Computational Study

- Random Instances
  - Start times ~ UniformInt$(0, t_{max})$
  - Lengths ~ UniformInt(1, 10)
  - Weights ~ UniformInt(1, 100)
  - Average of 10 replications
- Heuristics: greedy, IP-based, local search
- Max-weight: Compare B&P vs. assignment IP
- Use $f(S) = \sqrt{\sum_{i \in S} w_i}$ for concave function
  - Solve LP + primal heuristics

Max-Weight Function

$\|J\|$	t_{max}	Root Gap	B&P Gap	Sol. Time	% Sol.	Assign Gap	Sol. Time	% Sol.
100	100	0.94%	0.00%	41.56	100%	0.00%	436.45	100%
100	50	3.25%	0.00%	43.04	100%	0.75%	1199.34	60%
100	20	1.66%	0.00%	15.01	100%	0.68%	405.86	30%
250	250	1.54%	0.00%	5892.44	100%	2.01%	2228.56	50%
250	125	2.64%	0.00%	2779.20	100%	3.02%	-	0%
250	50	3.11%	0.00%	1006.74	100%	4.46%	-	0%

- Time limit: 3 hours
- Assignment IP solved with Gurobi
- B&P scales well to ~550 jobs

Square Root Function

$\|J\|$	t_{max}	CG Gap	CG Time	% Conv.	% Sol.
100	20	0.00%	252.55	100%	100%
250	50	0.30%	4959.18	70%	70%
400	80	1.30%	-	0%	0%

- Time limit: 6 hours
- If CG did not converge, used dual bounds
  - In-out column generation see Ben-Ameur et al 2007

Conclusion

- Proposed a set cover model for submodular interval scheduling
- Designed branch-and-price algorithm for two important cases

Future Work

- Extend to other submodular functions
- Extend to branch-price-and-cut

atoriello@isye.gatech.edu
sites.gatech.edu/alejandro-toriello