

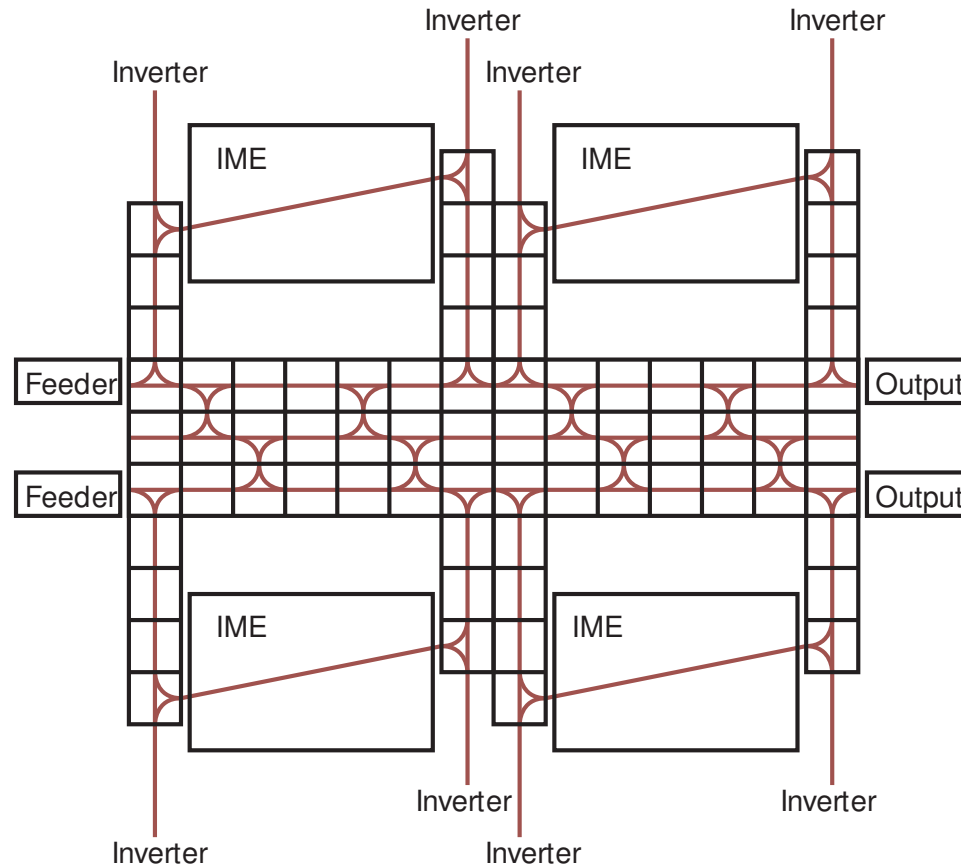
Optimal Routing and Scheduling in Flexible Manufacturing Systems using Integer Programming

Randy Cogill & Haitham Hindi

RC: Dept of Sys. Eng., Univ of Virginia

HH: Palo Alto Research Center, PARC

Overview



Parallel printing system:

- printers, inverters, feeders, finishers
- connected by transports

Overview (ctd)

Inspired by Maglaras, Meyn, Kumar, Harrison, Weiss, Dai, Kelly, Xiao, Johansson, Boyd, Bertsimas, Sethuraman, Gamarnik, etc

Problem: schedule and route sheets to maximize throughput

- maximum throughput schedules are usually not obvious, as we will see
- CDC2006: we presented *multi-stage multi-commodity* (MSMC); solved using *convex network flows*; obtained fractional flows, best case bounds
- this paper: we present dynamic *time-indexed* MSMC; solved using *integer network flows*; obtain *exact* integer flows, *exact* optimum

Other Application Domains

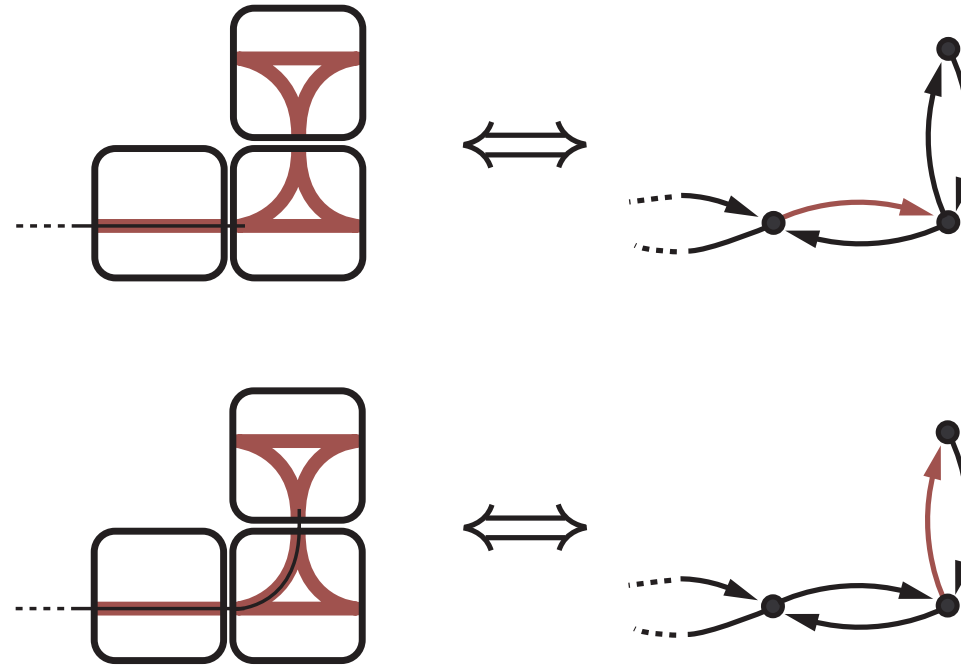
- general manufacturing flow line optimization
- transportation networks
- network system design, packet scheduling/routing, ...

Summary of Results

For a discrete model of a general printing network:

- We consider the problem of finding *periodic schedules* with max throughput
- This can be posed as a network flow-like 0-1 LP
- For small networks, 0-1 LP can be solved by existing software
- For larger (TIPP-scale) networks, we can:
 - Solve relaxation, obtaining a performance bound
 - Apply a rounding heuristic to obtain a suboptimal schedule
 - Resulting schedule has guaranteed performance bound
- Current work is a custom 0-1 solver which exploits problem-specific structure

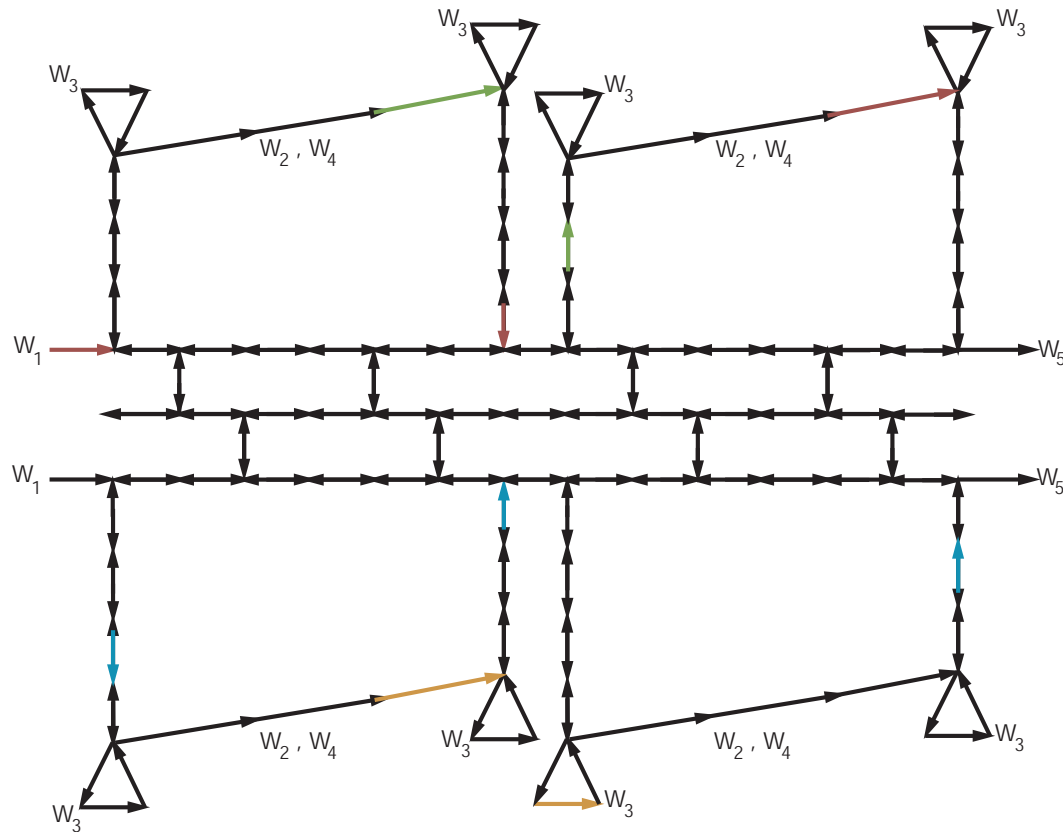
Discrete Model of TIPP



We consider a discrete-time model with discrete job locations:

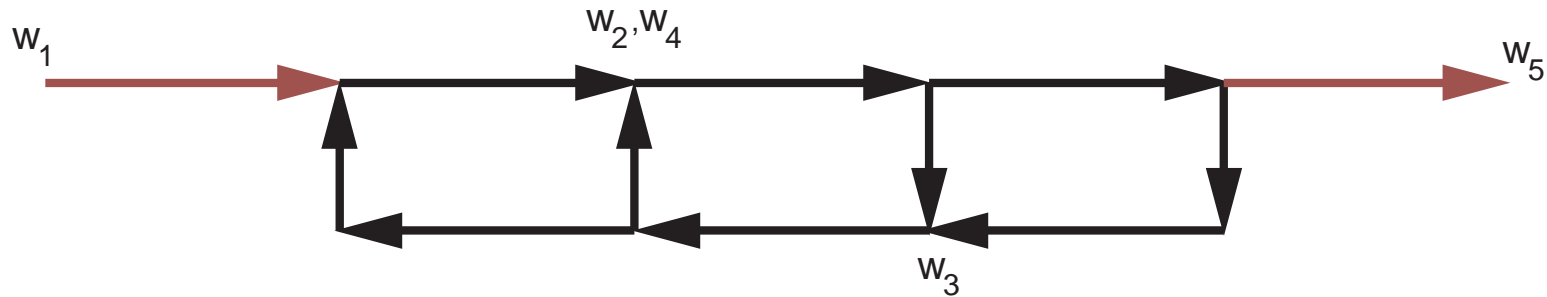
- Interconnection of components \iff interconnection of vertices
- Time for leading edge to cross a module \iff discrete time unit
- Page incoming to a module \iff active edge incoming to a vertex
- Processing time of any component is integer multiple of discrete time unit

Discrete Network Flow (cont.)



- Vertex $v \in \mathcal{V}$ can only process one job per $p(v)$ time steps
- We consider periodic schedules with a given cycle length T
- **Goal:** Schedule and route jobs to maximize the number of jobs entering per cycle

Periodic Schedules

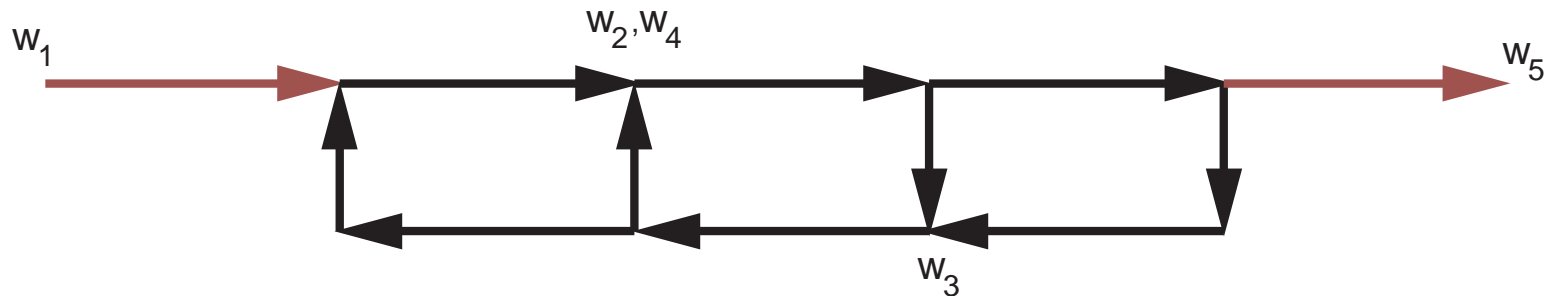


- **Periodic schedule:** The system repeats its state every T time steps
- Throughput maximizing schedules are periodic
- We start with a given cycle length T
- Steady-state throughput is

$$\frac{\# \text{ of utilized slots per cycle}}{T}$$

- Animation of a single sheet periodic schedule...

0-1 Linear Program: The Variables



The system state at time t is modeled by a set of 0-1 variables:

- x_{et}^j indicates the presence of a job in stage j on edge e at time t
- s_{kt}^1 indicates that job in stage 1 is introduced to waypoint w_k^1 at time t
- s_{kt}^{F+1} indicates that job in stage F will leave waypoint w_k^{F+1} at time $t + 1$

0-1 LP: Conservation Constraints



If a job enters vertex v at time t , a job must leave v at time $t + 1$:

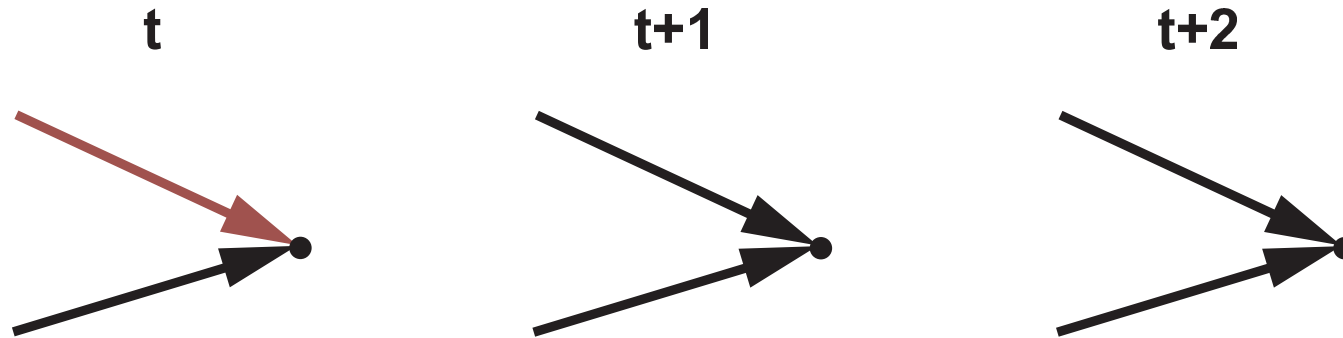
- If v is not a waypoint, this is expressed as

$$\sum_{e \in \mathcal{I}(v)} x_{et}^j = \sum_{e \in \mathcal{O}(v)} x_{e(t+1)T}^j$$

for each stage j and time slot t

- Similar constraints for waypoints, except jobs move to next stage

0-1 LP: Processing Time Constraints



At most one job may enter vertex v in $p(v)$ consecutive time steps:

- This is expressed as

$$\sum_{\tau=0}^{p_v-1} \sum_{j=1}^F \sum_{e \in \mathcal{I}(v)} x_{e(t+\tau)_T}^j \leq 1$$

for each time slot t

0-1 Linear Program

Overall 0-1 LP has:

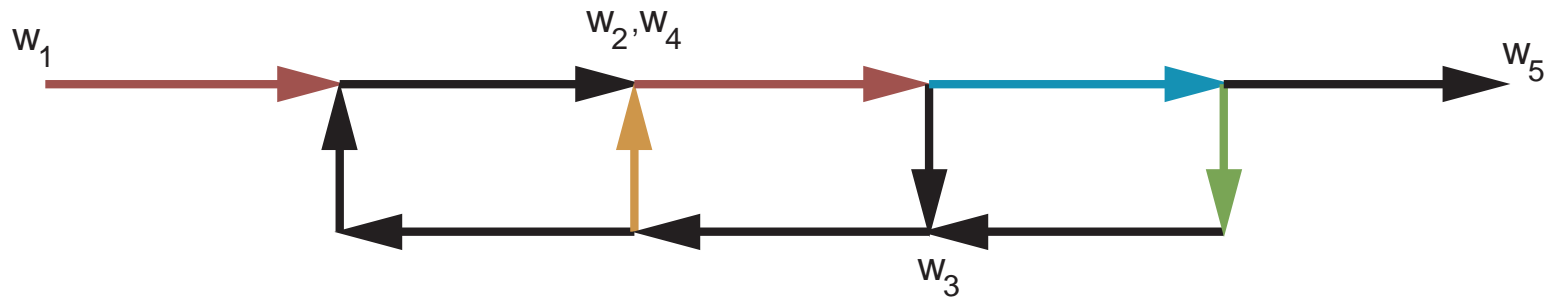
- Objective of maximizing $\sum_{k=1}^{K_1} \sum_{t=1}^T s_{kt}^1$
- $F|\mathcal{E}|T$ job location variables x_{et}^j
- $|\mathcal{W}_1|T$ source variables and $|\mathcal{W}_{F+1}|T$ sink variables
- $|\mathcal{V}|T$ processing time constraints
- $F|\mathcal{V}|T$ conservation constraints

For example, an instance with:

- $F = 4, T = 8, |\mathcal{V}| = 10, |\mathcal{E}| = 12, |\mathcal{W}_1| = 1, |\mathcal{W}_{F+1}| = 1$
- ...is specified as a 0-1 LP with 400 variables and 400 constraints

Example: Simple Re-entrant Line

Consider this simple reentrant system with $T = 8$:

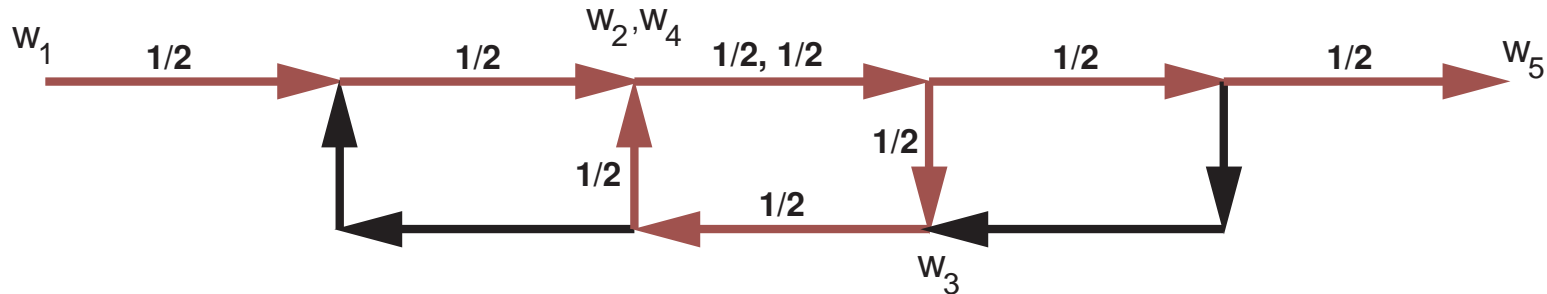


- Here we have a 0-1 LP with 400 variables and 400 constraints
- The CPLEX 0-1 solver handles this easily
- An optimal solution is actually not obvious!
- Optimal schedule shown in animation...

Relaxation and Rounding Heuristic

- Problems quickly become too large for a generic 0-1 LP solver
- We can still obtain performance bounds and suboptimal schedules by:
- **Relaxation:** Let 0-1 variables lie *between* 0 and 1
 - Resulting relaxation efficiently solved as a linear program
 - Max relaxation value an upper bound on max throughput
- **Rounding:** LP relaxation produces a “fractional packing” of single sheet schedules
 - Decompose fractional solution into constituent single sheet schedules
 - Repack the resulting schedules
 - More detail on this...

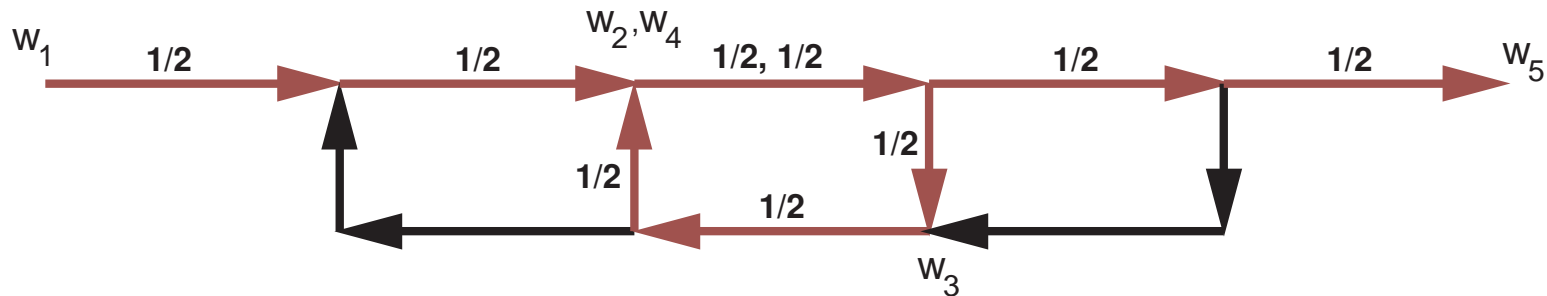
Rounding via Path Decomposition



Consider the optimal LP solution depicted above:

- At every time step, we have a weight of $1/2$ along the highlighted path
- This solution is a sum of T single sheet paths, each weighed by $1/2$

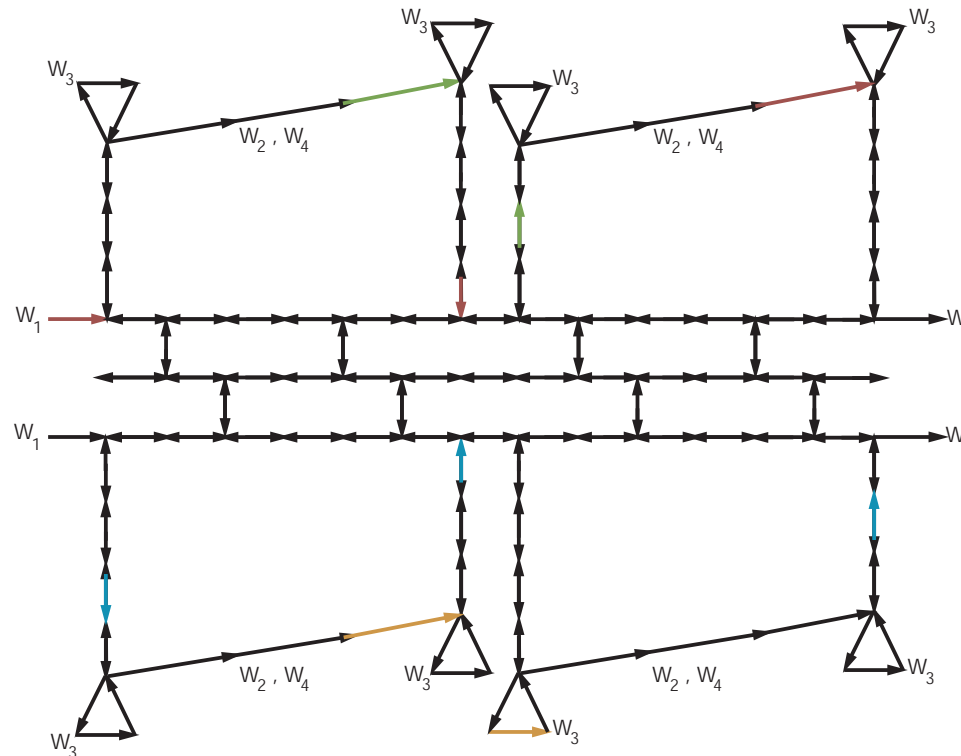
Rounding via Path Decomposition (cont.)



- There is an algorithm which decomposes any LP solution into a set of paths
- Given these paths, optimal fractional packing is given by LP solution
- Can think of LP relaxation as a relaxed packing problem
- **Rounding heuristic:**
 1. Given LP solution, decompose into paths
 2. Given paths, greedily construct a feasible *integral* packing

Example: TIPP System

Now consider a TIPP-scale example with $T = 10$:



- Waypoints are set for duplex jobs
- Processing time $p(v) = 4$ at printers, $p(v) = 3$ elsewhere
- 7640 variables and 4900 constraints (too big for CPLEX)

Current Work: Custom 0-1 Solver

- TIPP-scale examples are too large for a generic 0-1 solver
- Several key features can be exploited by a custom solver:
 - Have a rounding heuristic which appears to produce good suboptimal solutions
 - Cost function always takes integer values

In the previous example:

- At first iteration we have:
 - A feasible schedule obtaining objective value of 4
 - An upper bound of 5 on the optimal objective value
- A search would terminate as soon as either:
 - We can schedule one more job
 - Our upper bound decreases (even to, say, 4.9)

Conclusions

- We have considered the problem of finding max throughput periodic schedules
- This can be posed as a network flow-like 0-1 LP
- For small networks, 0-1 LP can be solved by existing software
- For larger networks, we can:
 - Solve relaxation, obtaining a performance bound
 - Apply a rounding heuristic to obtain a suboptimal schedule
- Current work is a custom 0-1 solver which exploits problem-specific structure
- Extensions to other manufacturing systems problems, eg, discrete *flexible job shop* problem
- Extensions to other domains: transportation, network design, packet switching, etc