Is Scheduling Still AI?
Part 1: Scheduling Basics

J. Christopher Beck
Dept. of Mechanical & Industrial Engineering
University of Toronto
Canada

ACAI Summer School
Freiburg, Germany
June 7 – 10, 2011

University of Toronto
Mechanical & Industrial Engineering

Outline

- Part 1: Core Scheduling Technologies
  - CP, MIP, & Metaheuristics
  - 90 minutes
- Part 2: State of the Art
  - CP + Metaheuristics, CP + MIP
  - 60 minutes
- Part 3: Polemics & Perspectives
  - The Past and the Future?
  - 30 minutes

Outline: Part 1

- What is Scheduling?
  - The fundamental bits
  - “The” classical problem
- Constraint Programming (CP)
  - Complete search and inference
- Mixed Integer Programming (MIP)
  - Complete search and relaxation
- Metaheuristics
  - Incomplete search

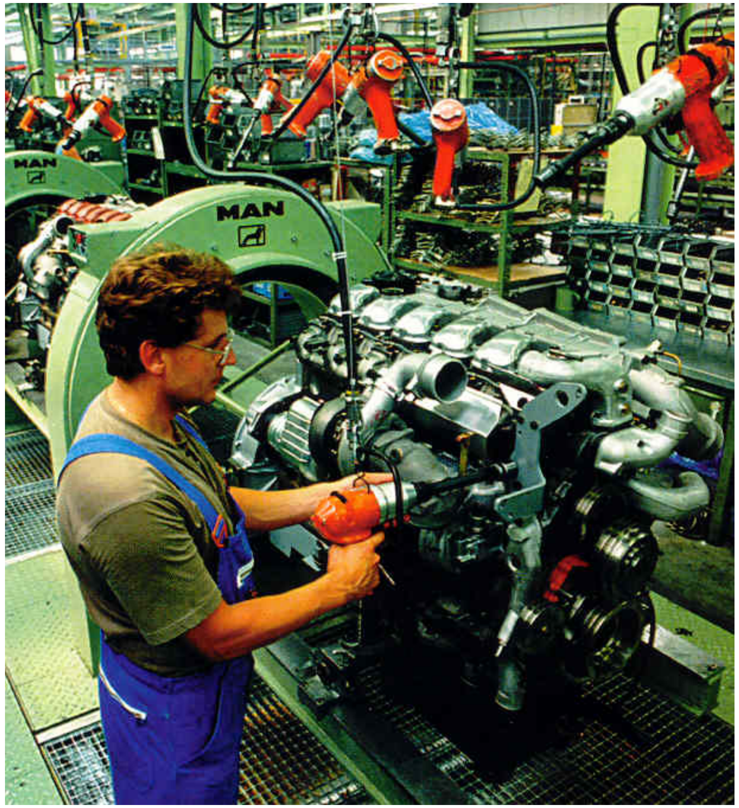
Scheduling is …

- The allocation of resources to activities over time
  - Mixing machines in food manufacturing
  - Classrooms at a university
  - Trucks & planes for FedEx
- Mathematically hard
- Industrially, economically, & environmentally important

Project Scheduling

- There are a series of operations required to complete a project
  - (e.g., build a bridge)
- Each operation requires resources
- Example: schedule the operations on the resources to meet all due dates

Manufacturing Scheduling



- Specialization of project scheduling
  - Series of operations
  - Resources required
  - Need to assign operations to resources over time in order to find shortest schedule, meet due dates, etc.

Airport Facility Scheduling

- Allocate resources required to “service” a plane
  - Runway, gate, baggage carousel, security personnel, re-fueling, re-stock food, …
  - Planes close to connecting flights?
  - Turn-around the plane quickly
- A new plane lands every minute

Workforce Scheduling

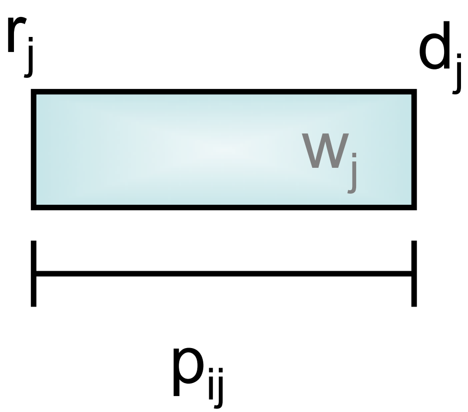
- You need a particular number of people with specific skills on each shift
- You need to schedule breaks, days-off, etc. taking into account regulations about #days/#hours worked without a break
- Nurse scheduling, call-centre staffing, …

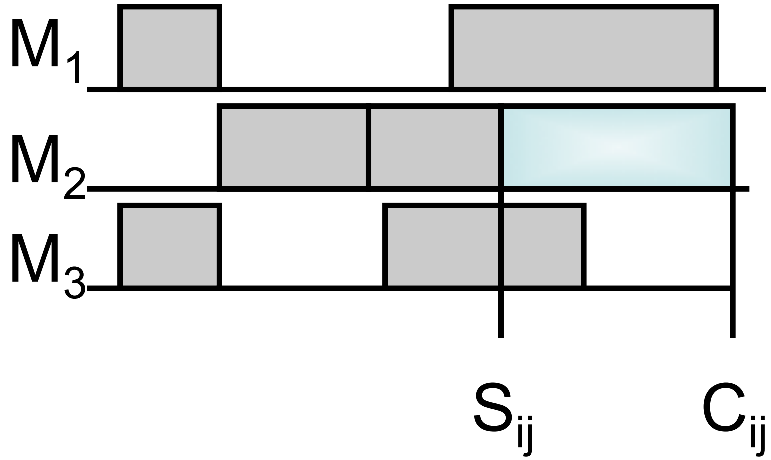
The Key Difference with Planning

- In classical scheduling we know all the operations (e.g., flights, production jobs) at the beginning of the solving process
- In some formulations, we may choose not to schedule all operations but typically (and for this lecture) assume that we never add to the set of operations during search

And now for some details …

Jobs





p_{ij} – processing time of job j on machine i

r_j – release date of job j

d_j – due date of job j

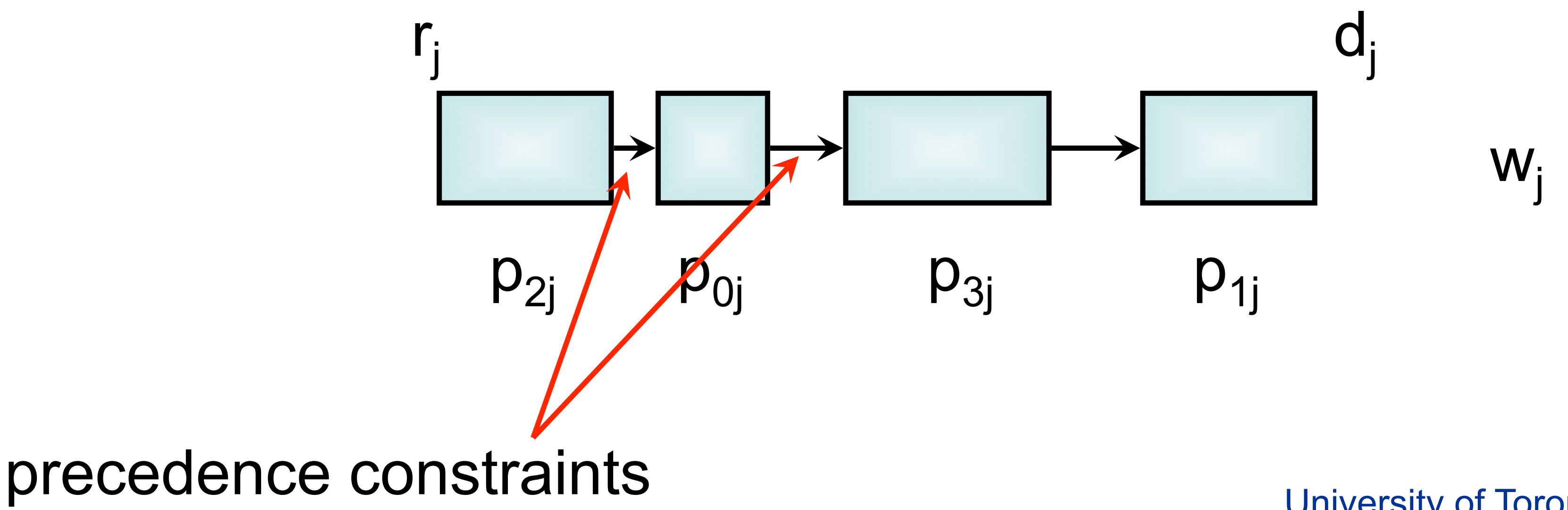
w_j – weight of job j

S_{ij} – starting time of job j on machine i

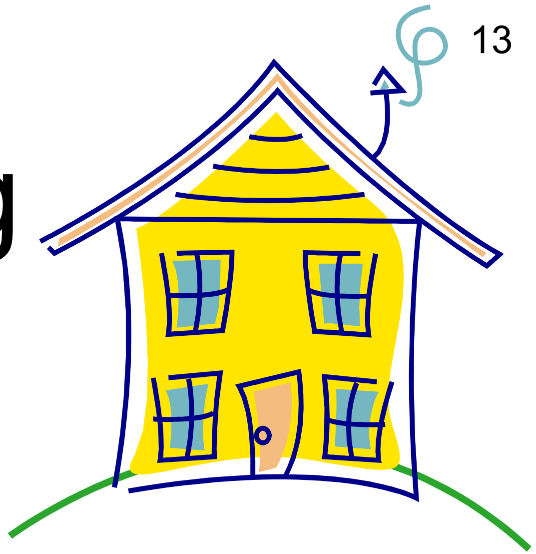
C_{ij} – completion time of job j

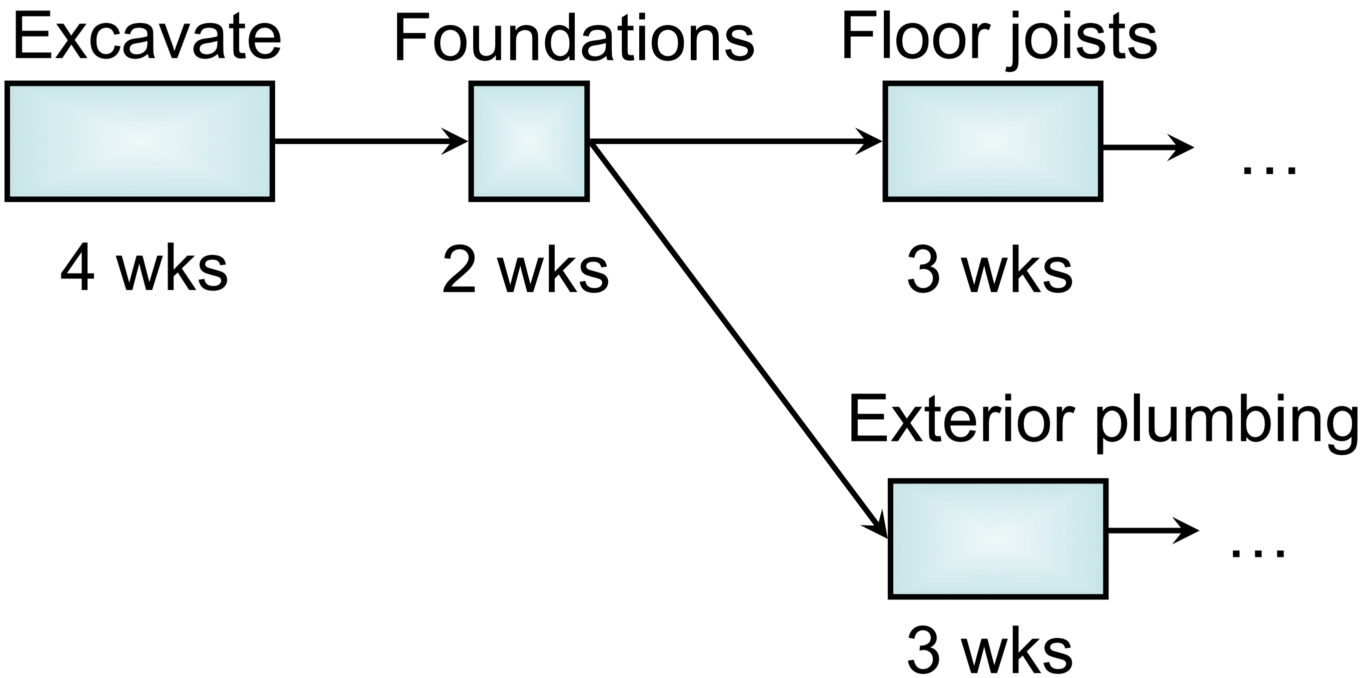
Jobs & Operations

- Often jobs are made up of a set of operations
  - usually once you start an operation, you can't interrupt it → "no pre-emption"



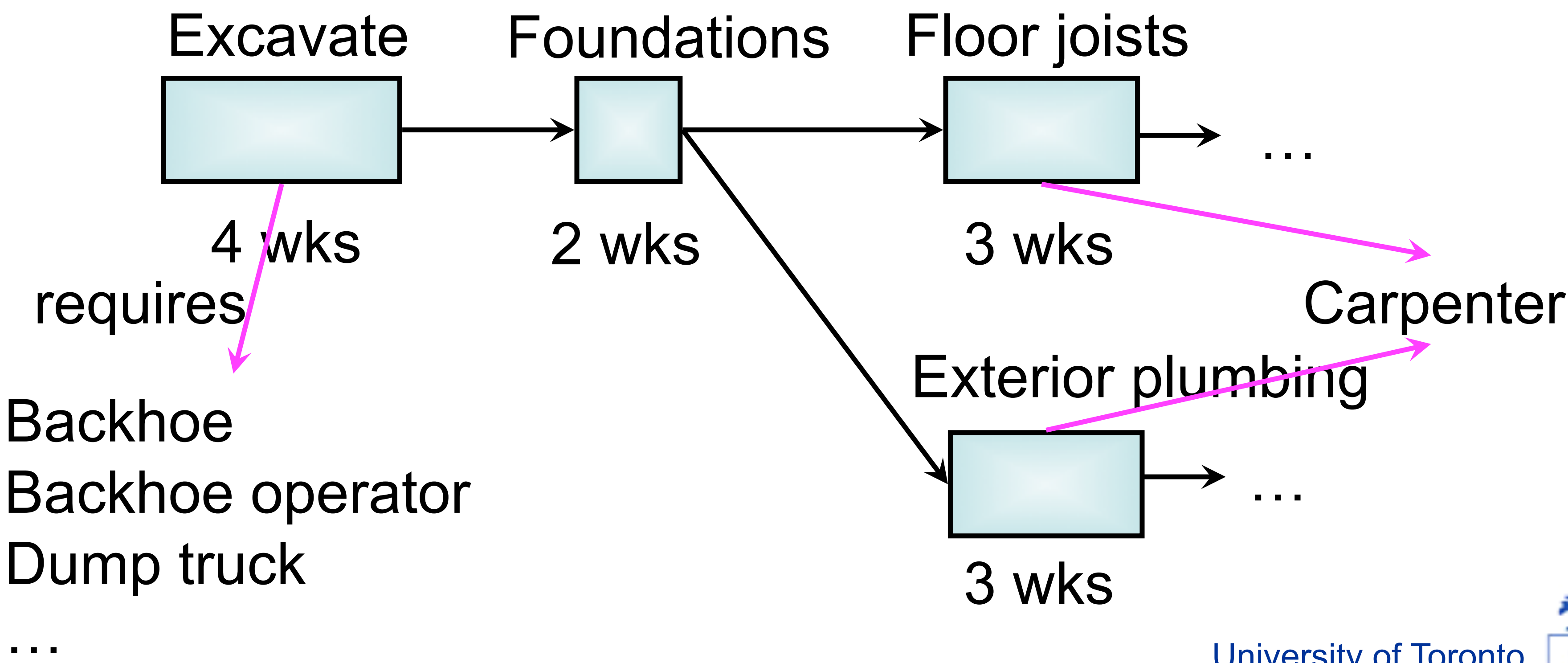
Example: House Building





House Building Resources

Resource Alternatives



Scheduling is …

- Assigning a start time and set of resources to each activity so that the temporal and resource constraints are satisfied
  - Temporal constraints: precedence, min/max
  - Resource constraints: capacity, type
- Often also have an objective function to optimize

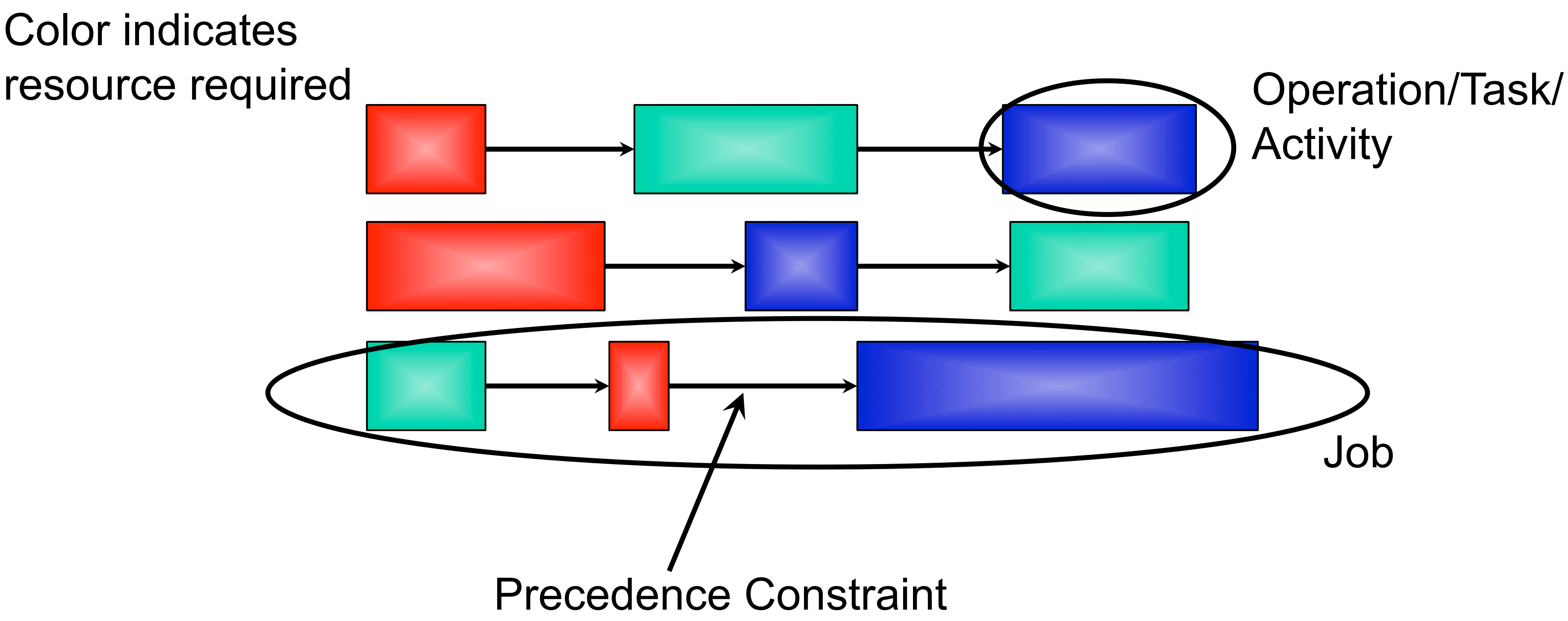
Classical Objective Functions

- Minimize maximum completion time (aka "makespan")
  - Min C_{max} [$C_{max} = \max(C_1, \ldots C_n)$]
- Minimize maximum lateness
  - Min L_{max} [$L_{max} = \max(C_1 - d_1, \ldots C_n - d_n)$]
- Minimize total weighted tardiness
  - Min $\Sigma w_j T_j$ [$T_j = \max(C_j - d_j, 0)$]

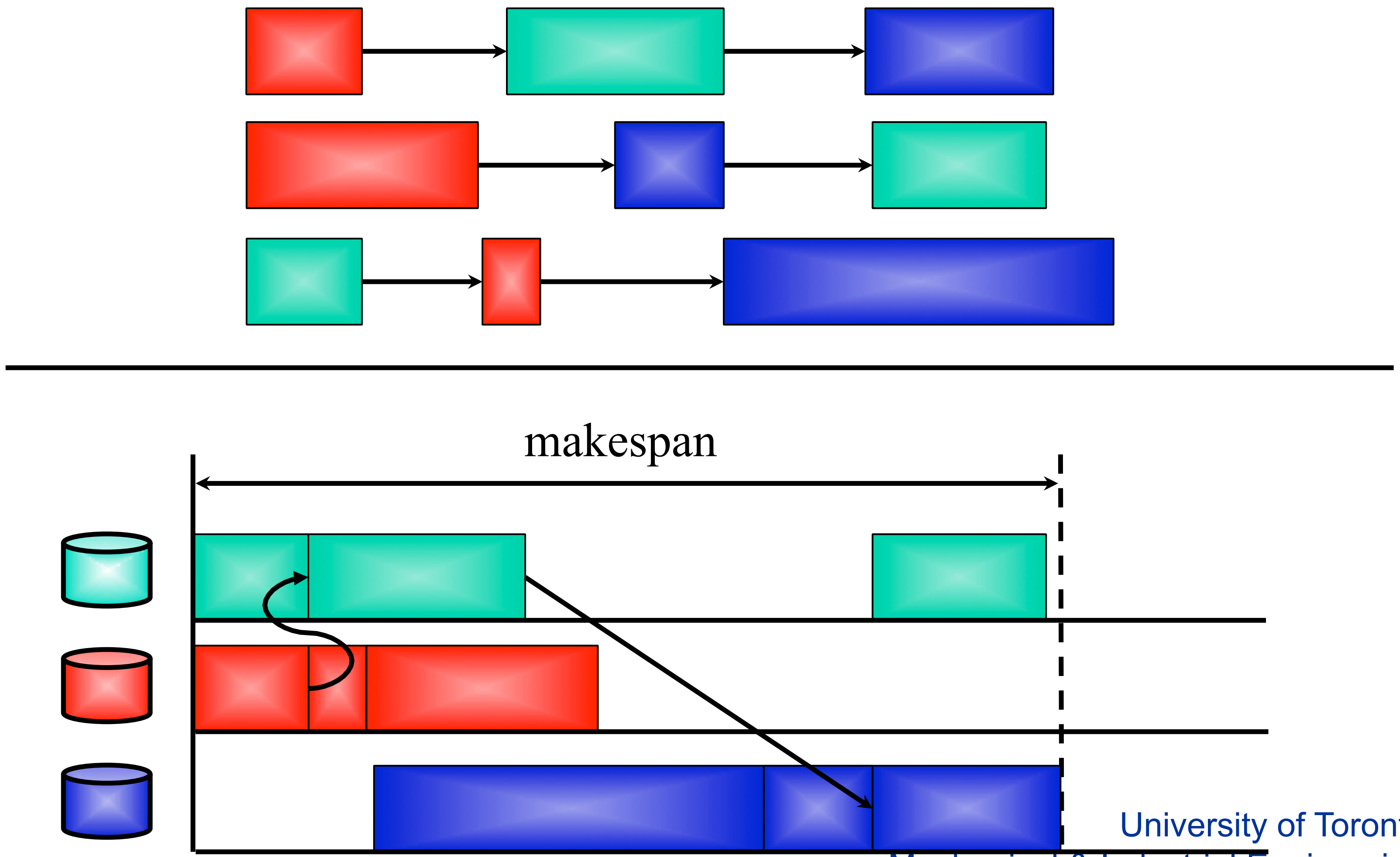
Hard vs. Easy

- Most interesting scheduling problems are at least NP-hard
  - some easy special cases
    - one-machine or two-machine (with restrictions)
  - some approaches use the special case algorithms as heuristics

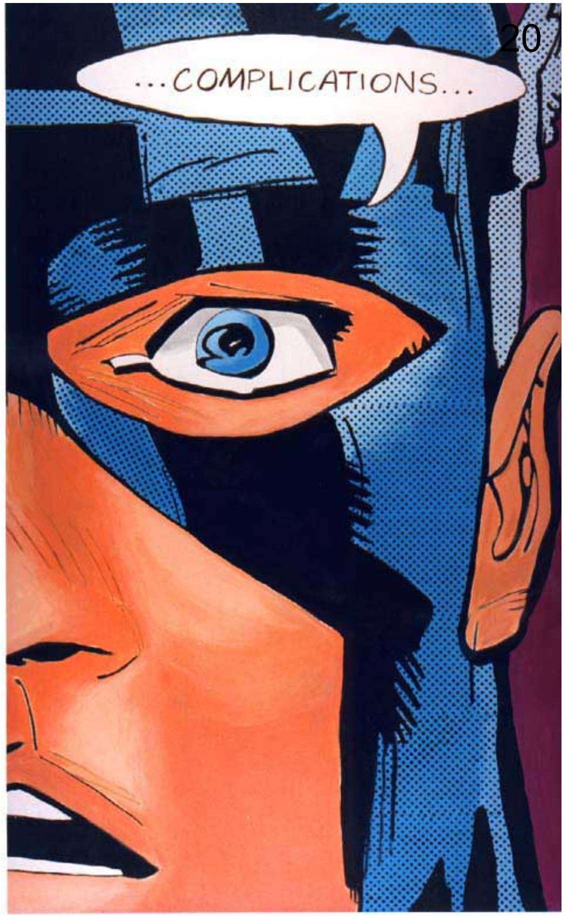
Job Shop Scheduling



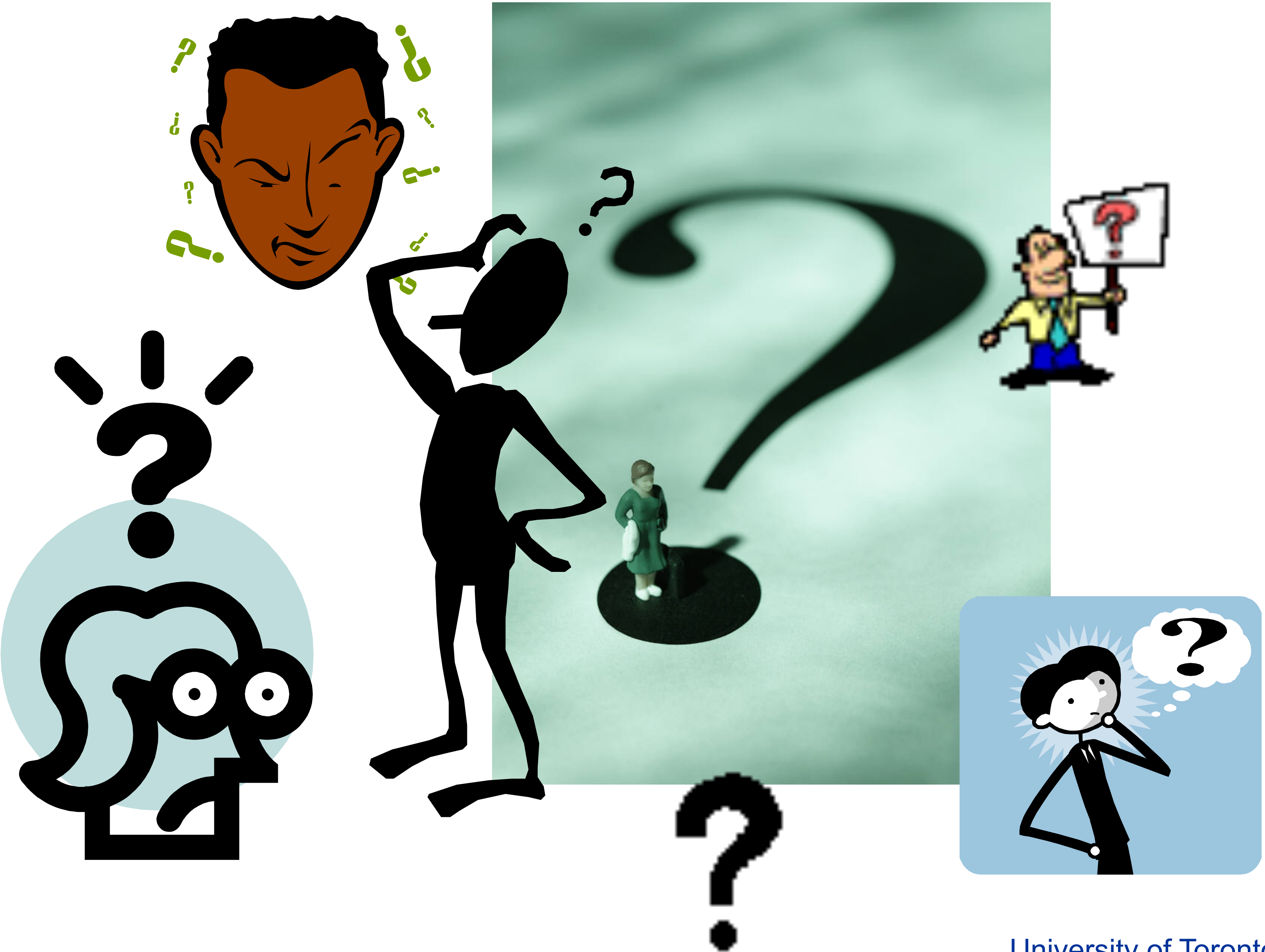
Job Shop Scheduling



Complications



- Resources can be continuously produced and consumed: tanks
- Batch resources: ovens
- Setups & sequence dependent changeovers
- Multi-criteria optimization
- Different processing times on different machines
- …



Outline: Part 1

- What is Scheduling?
  - The fundamental bits
  - "The" classical problem
- Constraint Programming (CP)
  - Complete search and inference
- Mixed Integer Programming (MIP)
  - Complete search and relaxation
- Metaheuristics
  - Incomplete search

What is Constraint Programming (CP)?

- An approach to combinatorial optimization arising from Artificial Intelligence and Computer Science
  - in contrast to Operations Research
- Core technology
  - tree search + inference
- Successes: scheduling, planning, network provisioning, graph theory, …

Constraint Satisfaction Problem (CSP)

- Given:
  - V, a set of variables $\{v_0, v_1, \ldots, v_n\}$
  - D, a set of domains $\{D_0, D_1, \ldots, D_n\}$
  - C, a set of constraints $\{c_0, c_1, \ldots, c_m\}$
- Each constraint, c_i, has a scope $c_i(v_0, v_2, v_4, v_{117}, \ldots)$, the variables that it constrains

Constraint Satisfaction Problem (CSP)

- A constraint, c_i, is a mapping from the elements of the Cartesian product of the domains of the variables in its scope to {T,F}
  - $c_i(v_0, v_2, v_4, v_{117}, \ldots)$ maps: $(D_0 \times D_2 \times D_4 \times D_{117} \times \ldots) \rightarrow \{T,F\}$
- A constraint is satisfied iff the assignment of the variables in its scope map to T

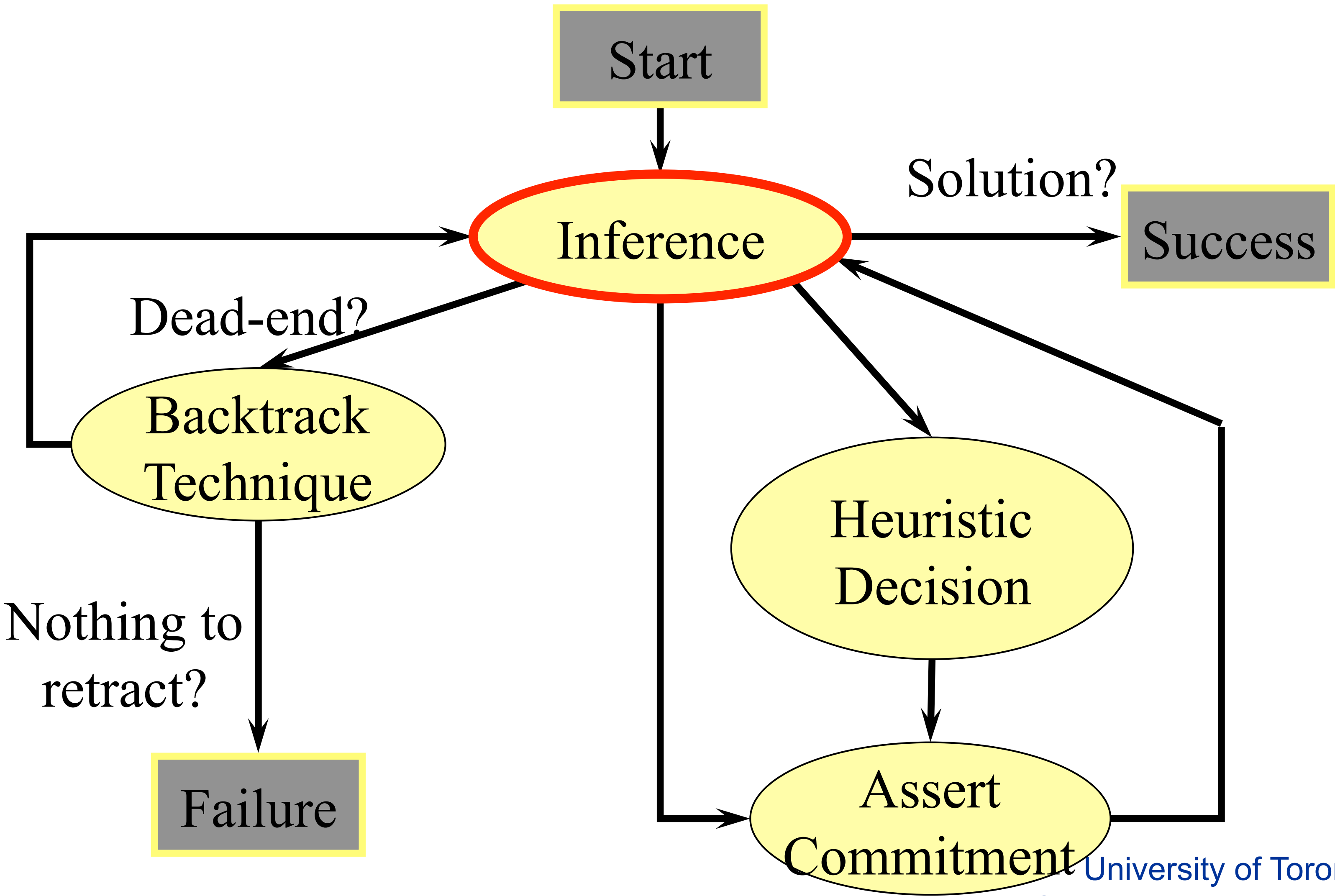
Constraint Satisfaction Problem (CSP)

- In a solution to a CSP:
  - each variable is assigned a value from its domain: $v_i = d_i$, $d_i \in D_i$
  - each constraint is satisfied

Constraint Optimization Problem (COP)

- A CSP plus a cost function f(V)
  - f is a mapping from the Cartesian product of a subset of the domains to integers or reals
- A solution is a solution to the CSP where f is (wolog) minimized

Generic CP Algorithm

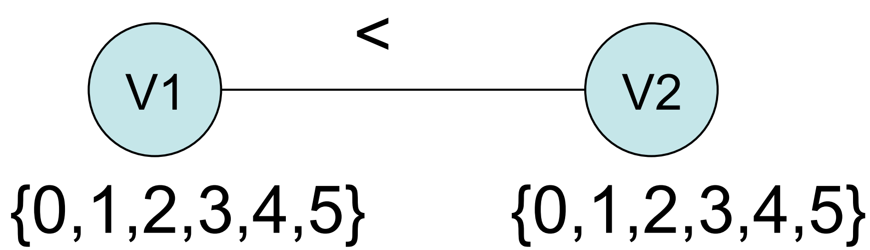


Arc Consistency

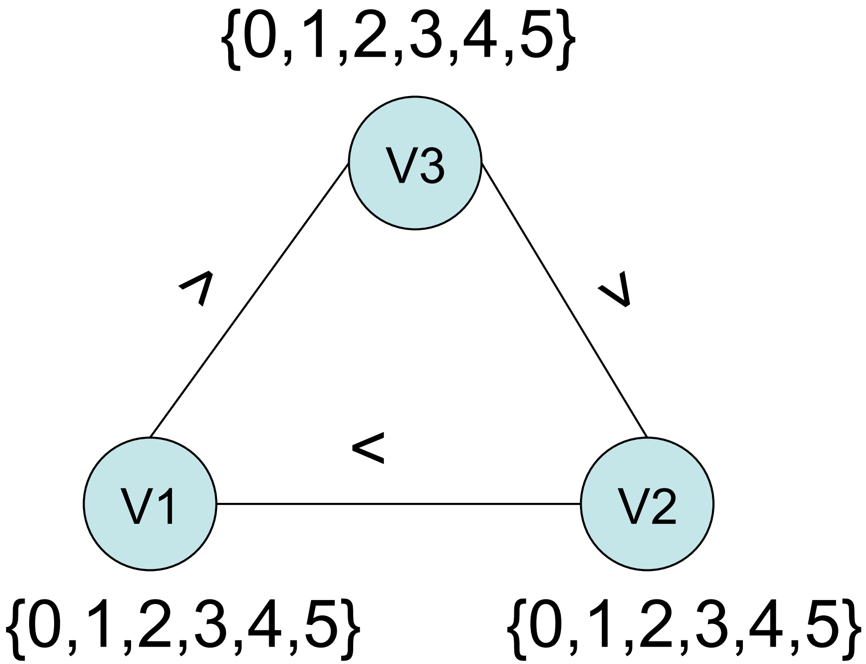
- Fundamental notion in CP!
- Given: $c_1(v_1,v_2)$
  - a binary constraint
  - e.g., $v_1 < v_2$
- Given: $D_1 = D_2 = \{0, 1, \ldots, 5\}$

Arc Consistency

- c_1 is arc consistent iff
  - for all values $d_1 \in D_1$ there exists a value $d_2 \in D_2$ such that $c_1(v_1=d_1, v_2=d_2) \rightarrow T$
  - And similarly for all values $d_2 \in D_2$
  - We say d_1 "supports" d_2 (and vice versa)



What Now?

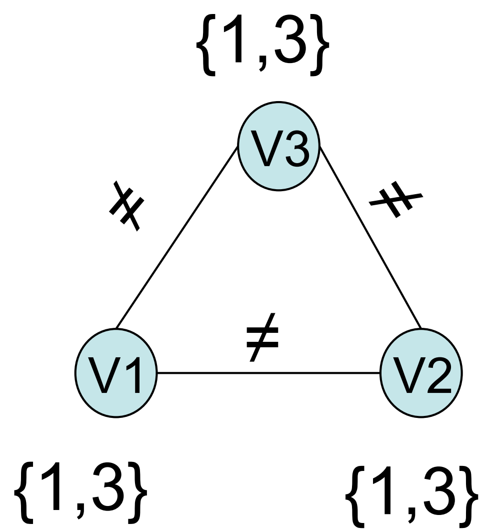


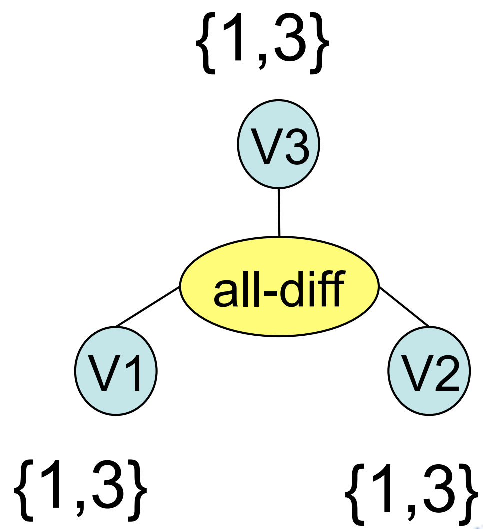
Generalized Arc Consistency (GAC)

- Given: $c_1(v_1,\ldots, v_m)$
- C_1 is GAC iff
  - for all variables d_i, for all values $d_i \in D_i$ there exists a tuple of values $[d_j \in D_j]$, $j \neq i$ such that $C_1 (v_i=d_i,[v_j=d_j]) \rightarrow T$
- E.g., $c_1(v_1,v_2,v_3,v_4)$
  - for every value in $d_1 \in D_1$ there must be some triple $[d_2 \in D_2, d_3 \in D_3, d_4 \in D_4]$ s.t. $c_1(v_1=d_1, v_2=d_2, v_3=d_3, v_4=d_4) \rightarrow T$

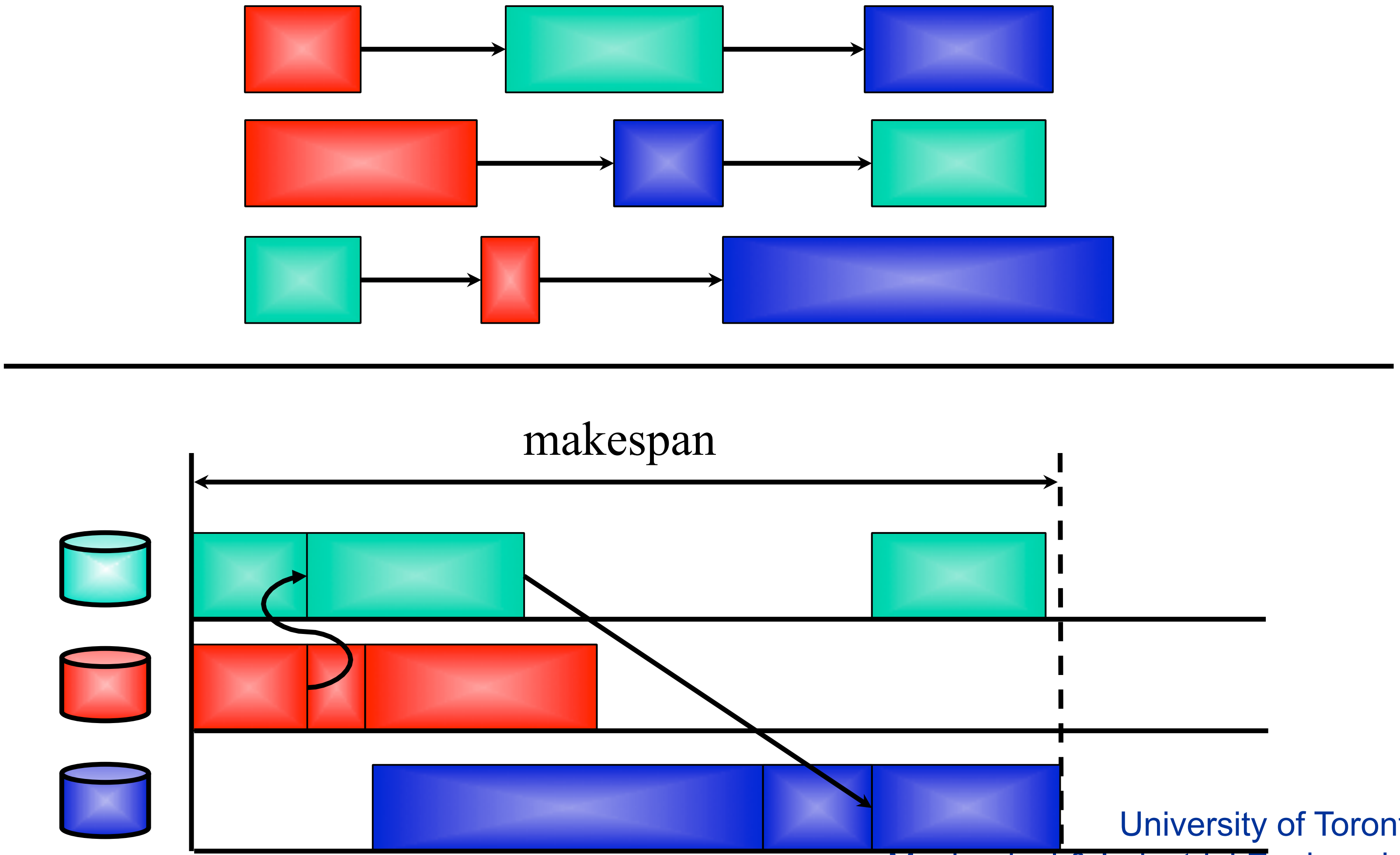
All-Diff vs. Clique of ≠

- all-diff(v_1, v_2, …, v_n) $=_{def}$
  $v_i \neq v_j$ for $1 \leq i < j \leq n$
- $D_1=D_2=D_3=\{1,3\}$
- Establish AC (or GAC) for
  - $v_1 \neq v_2$, $v_1 \neq v_3$, $v_2 \neq v_3$
  - all-diff(v_1,v_2,v_3)





Job Shop Scheduling



A CP Model for JSP

$$
\begin{array}{lll}
\min & C_{max} & \\
\text{s.t.} & S_j \geq 0 & \forall j \in \mathcal{J} \\
 & S_j + p_j \leq C_{max} & \forall j \in \mathcal{J} \\
 & S_j + p_j \leq S_i & \forall (j,i) \in \mathcal{E} \\
 & \texttt{disjunctive}(\boldsymbol{S}., \boldsymbol{p}.) & \forall k \in \mathcal{K} \\
 & S_j \in \mathbb{Z} & \forall j \in \mathcal{J}.
\end{array}
$$

Minimize the makespan

All activities end before the makespan

Precedence constraints

Where:

- $\mathcal{J}$ is the set of all activities
- $\mathcal{K}$ is the set of all resources
- $\mathcal{E}$ is set of all precedence constraints
- S_j is the start-time variable of job j
- p_j is the processing time of job j

`disjunctive` is a global constraint enforcing the resource capacity

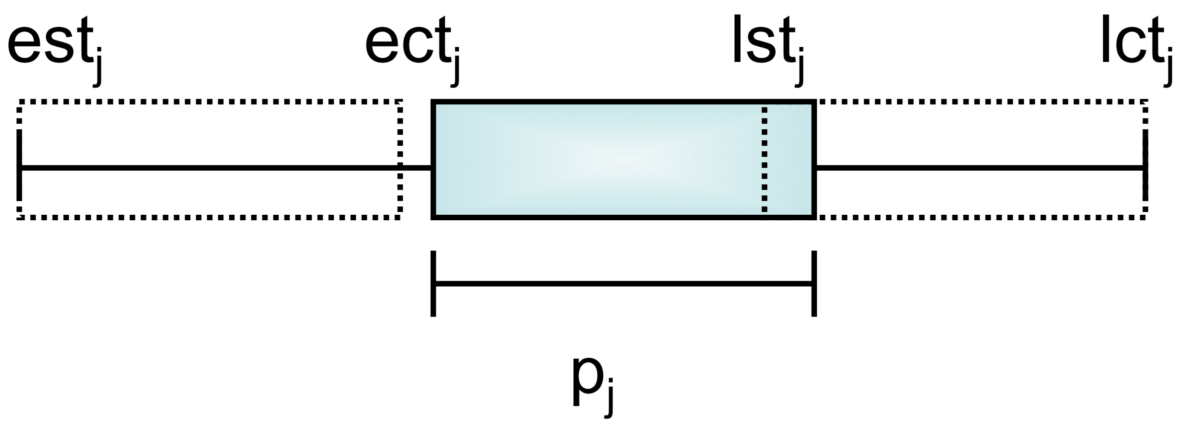
The `disjunctive` Global Constraint

- Called disjunctive because it enforces:

$$S_j + p_j \leq S_i \vee S_i + p_i \leq S_j$$

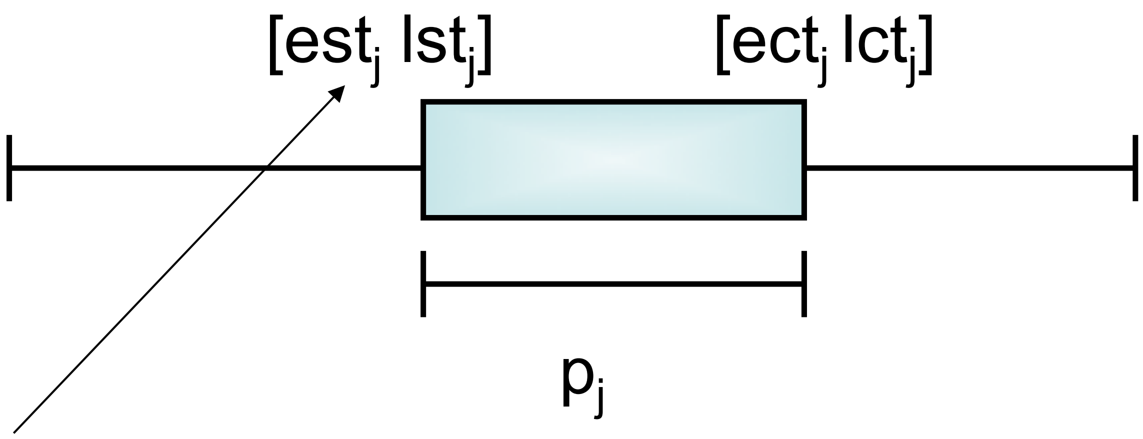
  for all activities *i,j* on the same resource.
- There are a number of inference algorithms that have been invented – we'll look at only one.

Notation



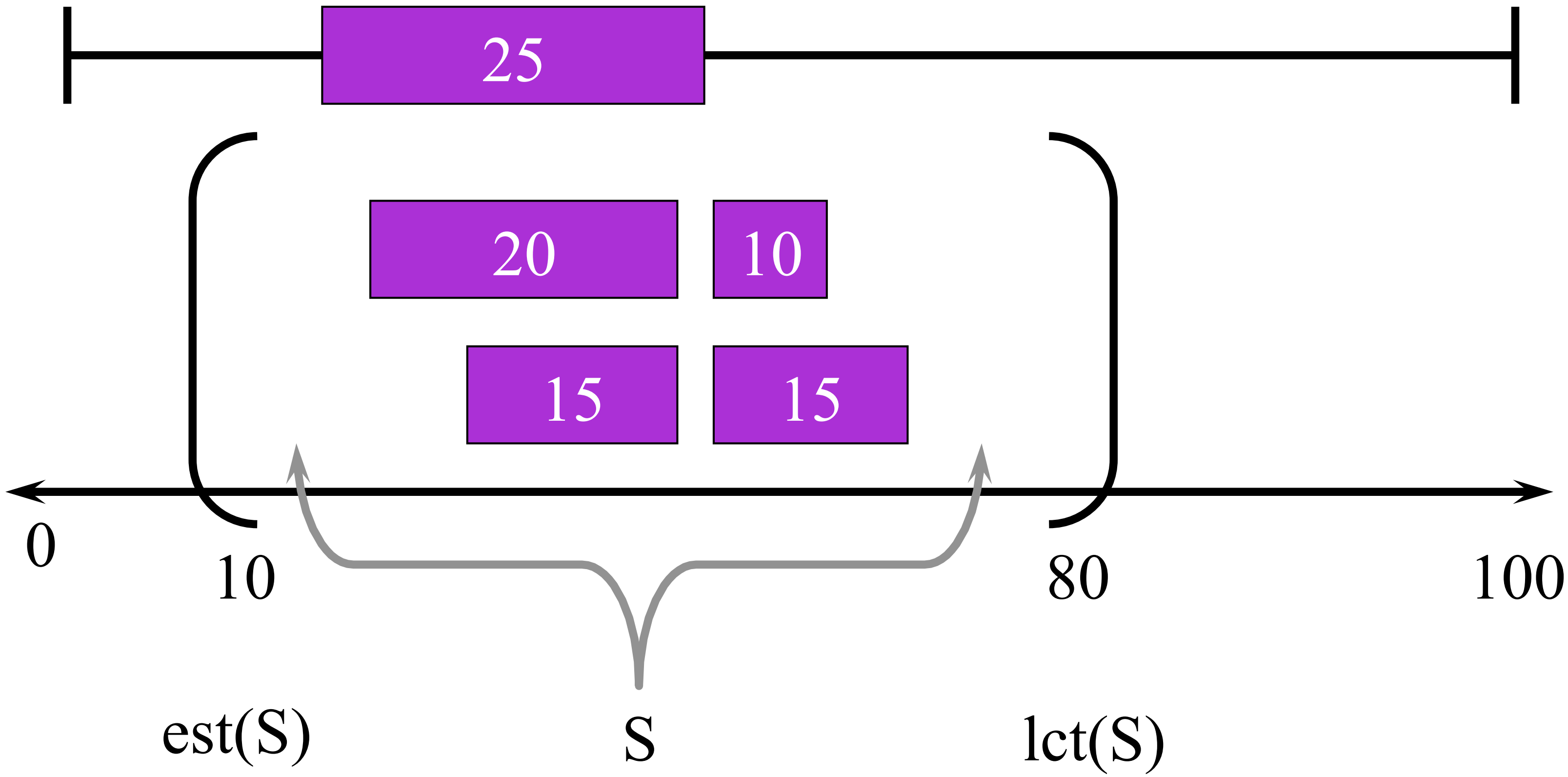
p_j – processing time of activity j (aka duration)
est_j – earliest start time of activity j
lst_j – latest start time of activity j
ect_j – earliest completion time of activity j
lct_j – latest completion time of activity j

Notation

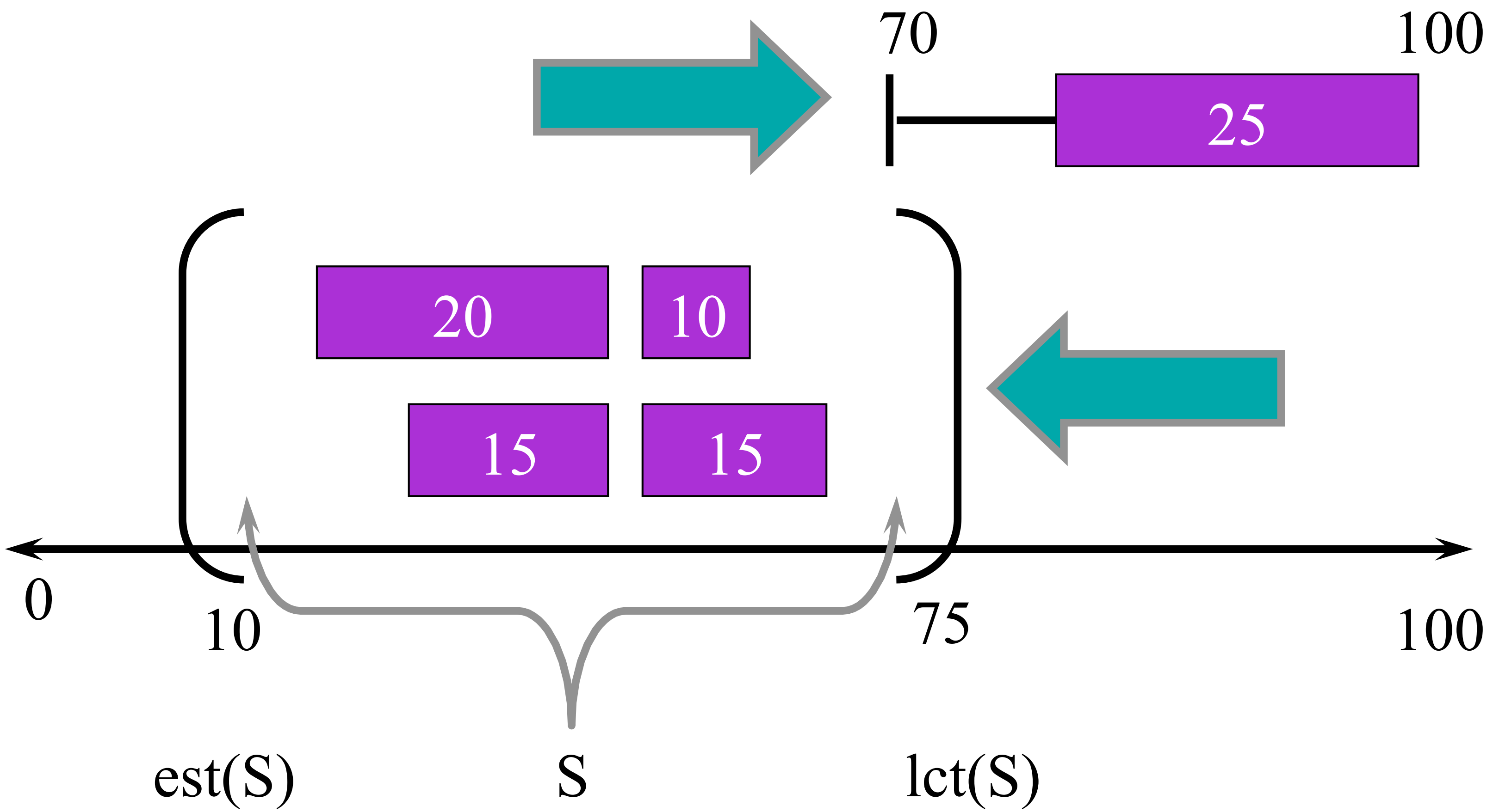


Domain of start times
(represented by an interval)

Edge-Finding Exclusion



Edge-Finding Exclusion



Exclusion Rules

On the same, unary capacity resource

For all non-empty subsets, S, and activities A∉S:

$$\left[\begin{array}{l}(lct(S) - est(S) < p_A + p(S)) \\ \wedge\ (lct(S) - est_A < p_A + p(S))\end{array}\right] \longrightarrow est_A \geq est(S) + p(S)$$

$$\left[\begin{array}{l}(lct(S) - est(S) < p_A + p(S)) \\ \wedge\ (lct_A - est(S) < p_A + p(S))\end{array}\right] \longrightarrow lct_A \leq lct(S) - p(S)$$

Global Constraints

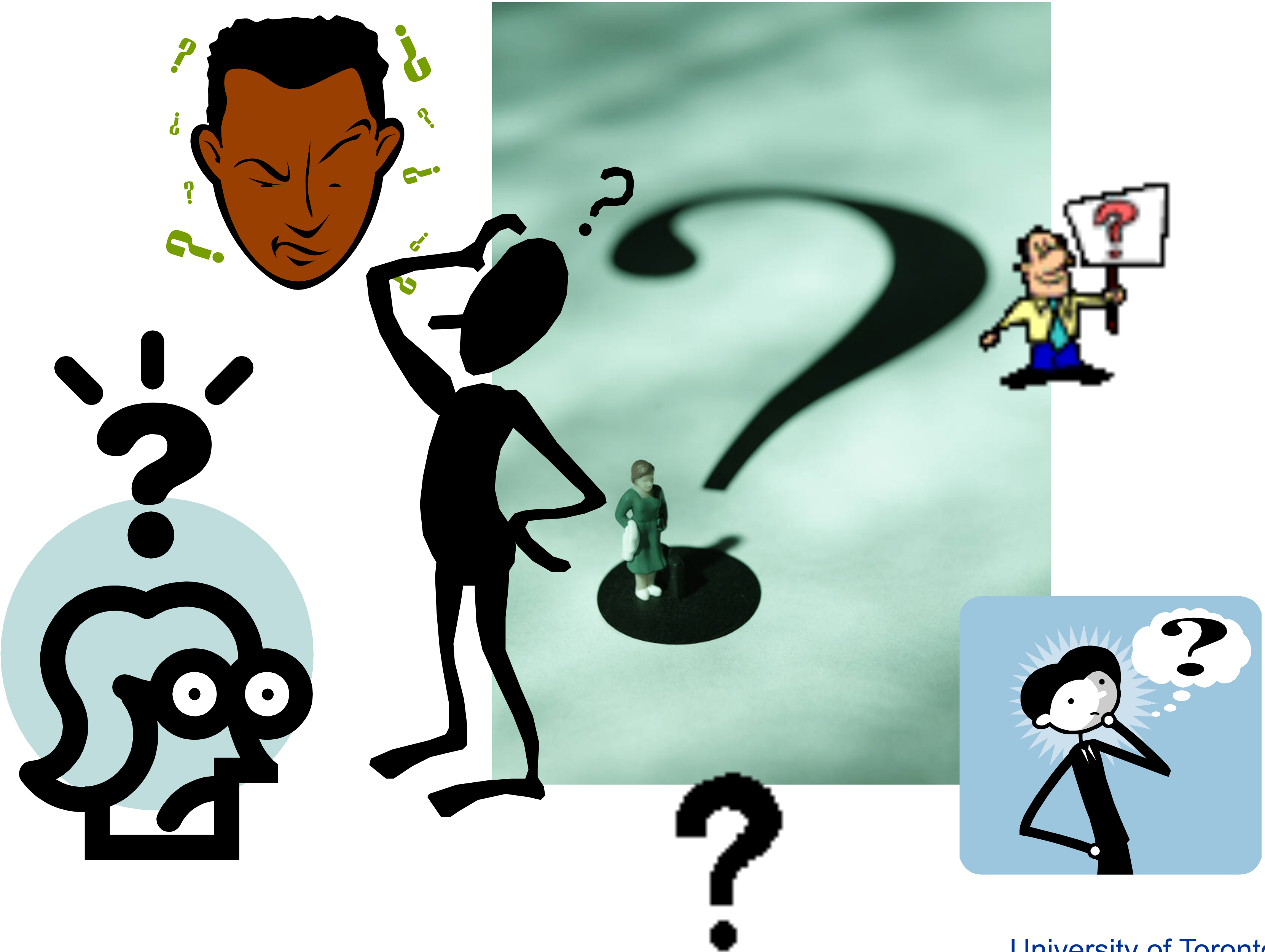
- A lot of interesting global constraints for scheduling
  - Balance constraint [Laborie, 2003]
  - Setups (TSP and AP) [Focacci et al, 2000]
  - Inter-distance [Artiouchine & Baptiste, 2005]
  - Timetable Edge-finding [Vilim, 2011]

Other Critical Solver Components

- Search (branching heuristics)
  - Min-slack [Cheng & Smith, 1993]
  - Texture measurements [B., 1999]
  - Solution-guided search (stay tuned …)
- Backtracking
  - Usual standard CP approaches
    - Chronological, LDS, restart, …

What Makes CP Different?

- Rich, expressive language
  - you can define anything you want as a constraint (not always a good thing)
- Focus on inference as the key technique to reduce search tree



Outline: Part 1

- What is Scheduling?
  – The fundamental bits
  – "The" classical problem
- Constraint Programming (CP)
  – Complete search and inference
- Mixed Integer Programming (MIP)
  – Complete search and relaxation
- Metaheuristics
  – Incomplete search

Mixed Integer Programming (MIP)

- Very successful complete optimization approach
- From the Operations Research/Applied Math community
- (Much) longer history than CP
  - 1940s and 1950s

MIP Basics

- Variables: integer or continuous
- Constraints: linear
- Objective function: linear
- (more accurately called Mixed Integer Linear Programming (MILP))

Comment: Much more restricted language than CP

MIP Basics

$$\min \sum_{i \in V} c_i x_i$$

$$\text{s.t.} \sum_{i \in V} \sum_{j \in C} a_{ij} x_i \leq b_j$$

$$V = V_I \cup V_R$$

$$x_i \in \mathbb{Z}, \forall i \in V_I$$

$$x_i \in \mathbb{R}, \forall i \in V_R$$

Objective function

Could be ≤, =, or ≥

Constraints

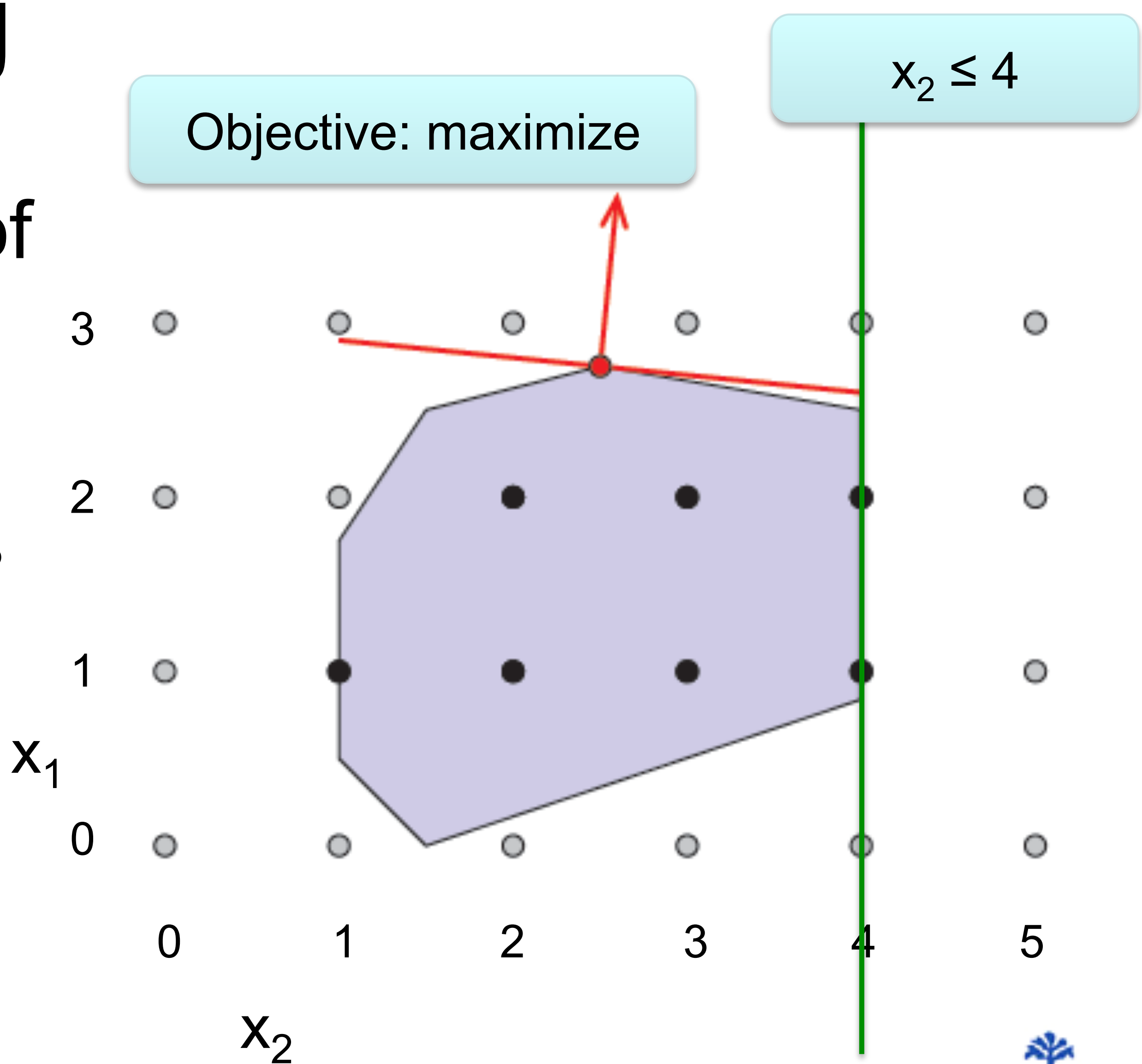
Integer variables

Continuous variables

Continuous (linear) relaxation: poly-time soluble!

MIP Solving

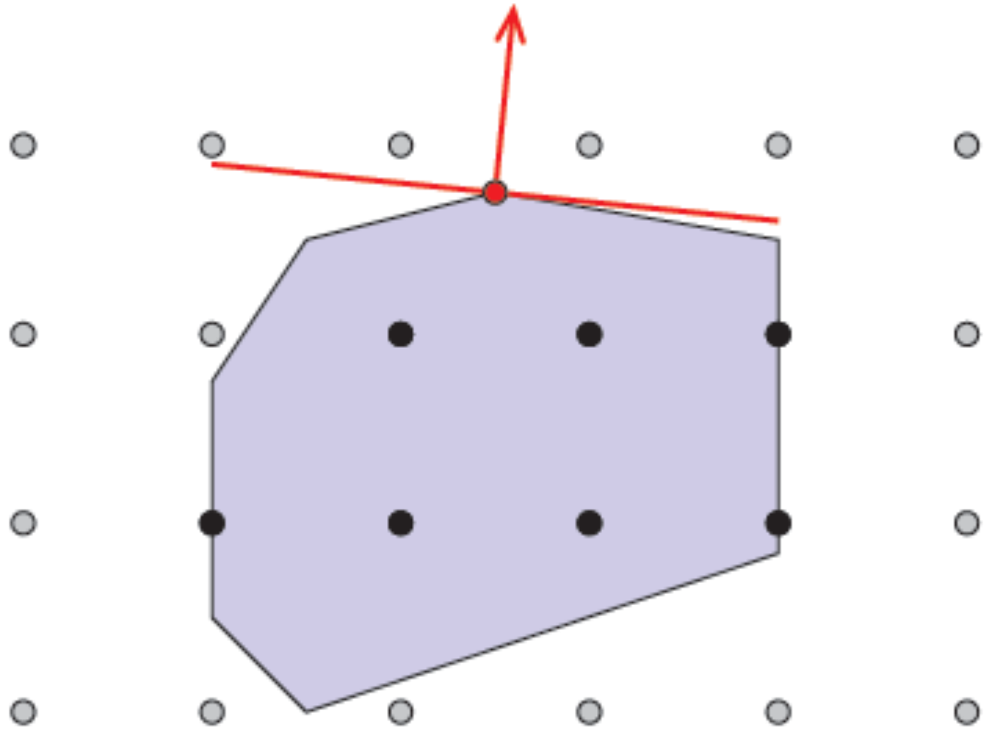
- Combination of
  - tree search
  - relaxation
  - cutting planes



Thanks to Stefan Heinz & Timo Berthold, Zuse Institute Berlin, for the pictures.

Linear Relaxation

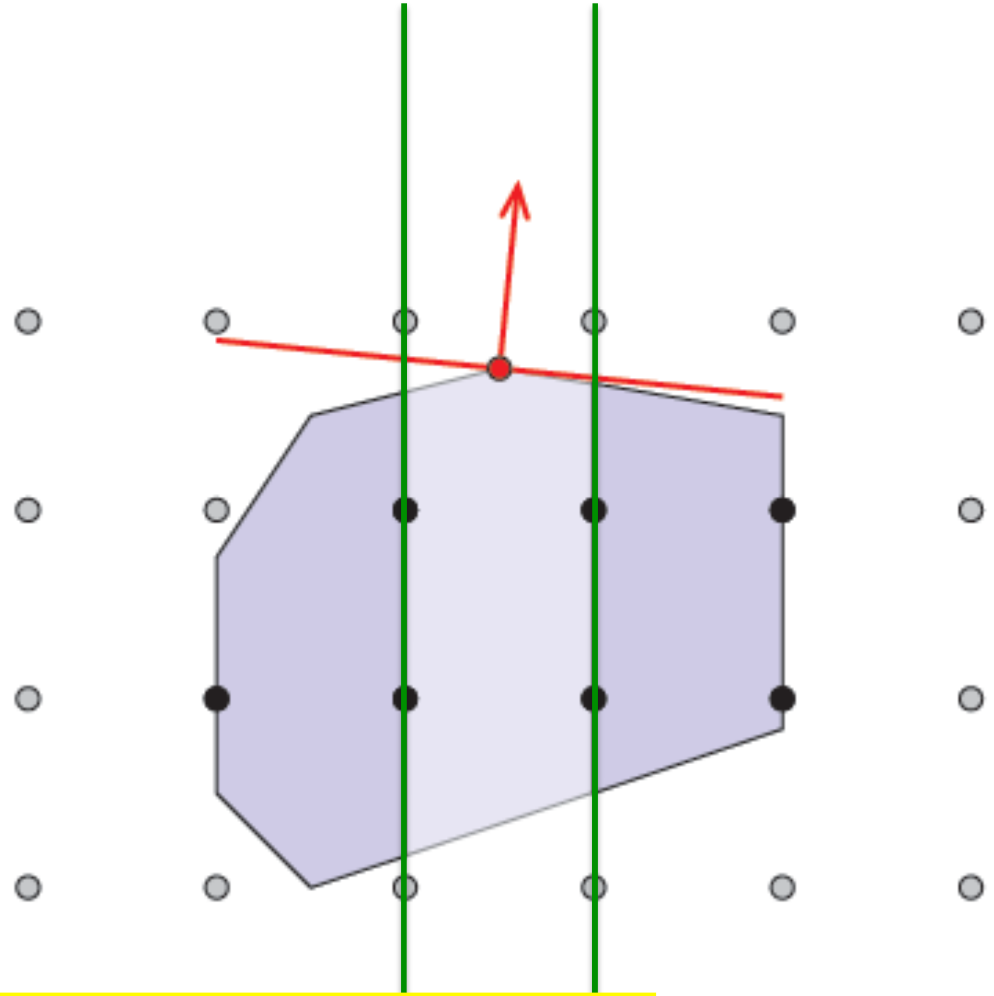
- Finds a "corner" that maximizes the cost function
  - Algorithms: simplex, dual simplex, interior point, …



So are we done?

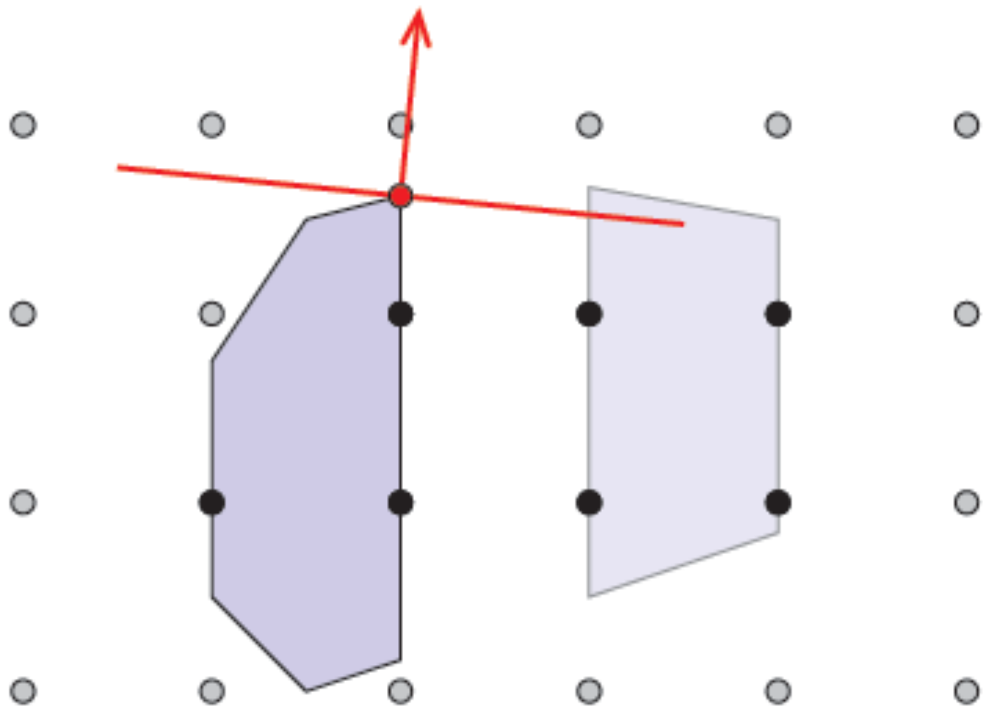
Branching

- Add a constraint
- Focus on one sub-problem
- Return (backtrack) later



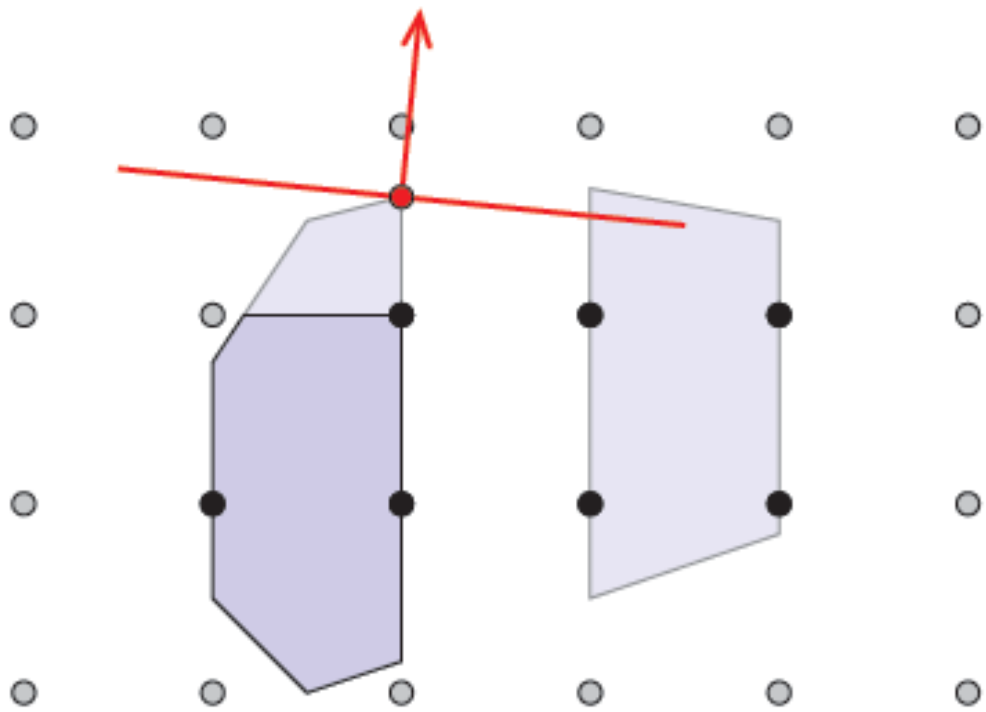
What constraints have been added here?

Branching



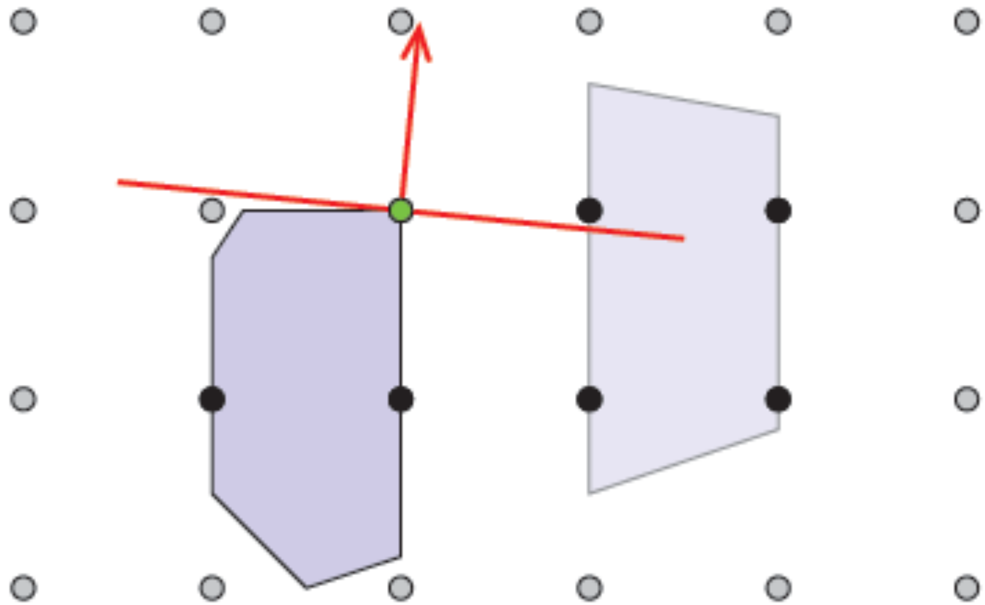
Bounds Strengthening

- A form of inference



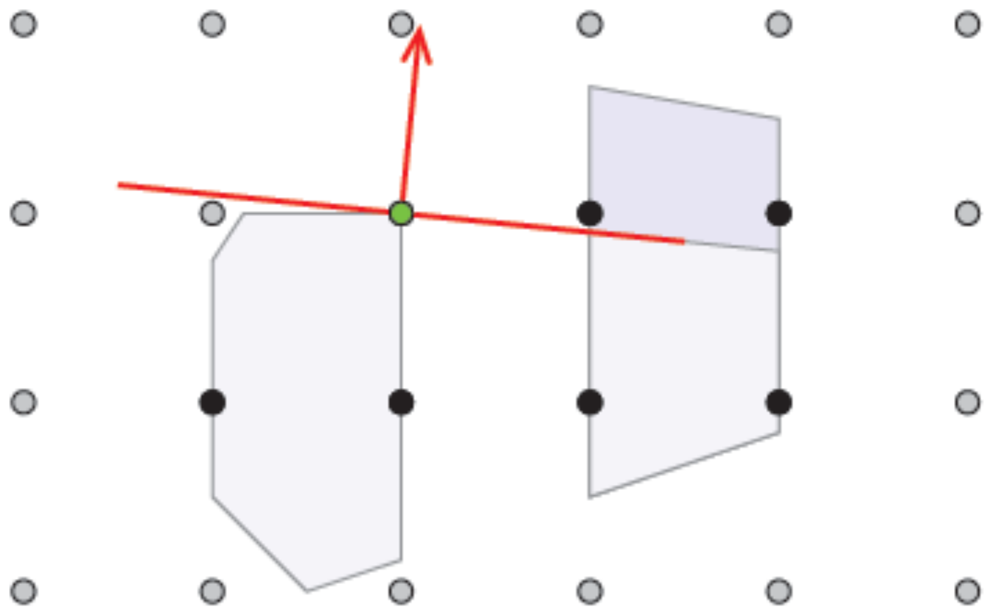
Solve LP Again …

- Integer solution!
  - recall that the LP solution is guaranteed to be a corner

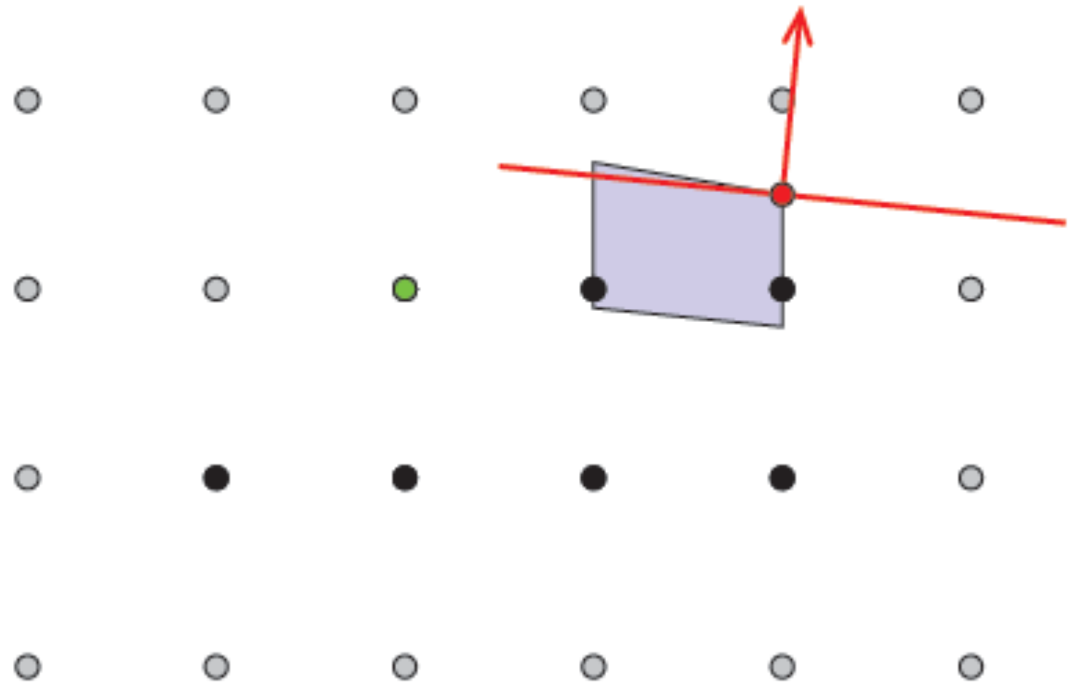


Bounding

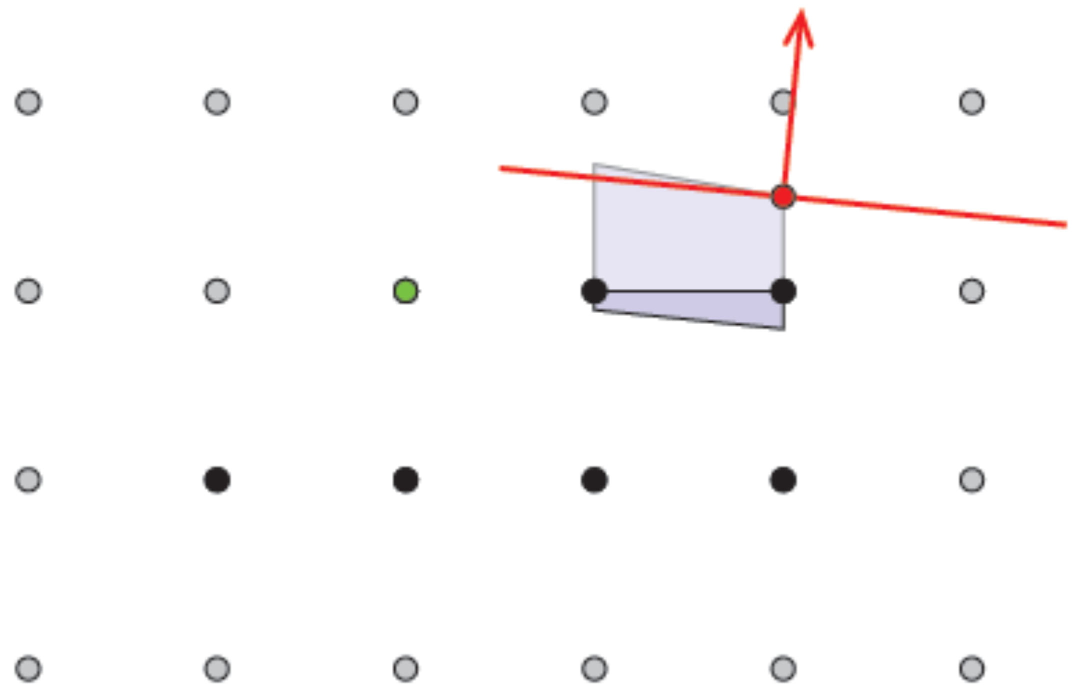
- Found an "incumbent" solution so add a constraint to require a better solution



Backtrack

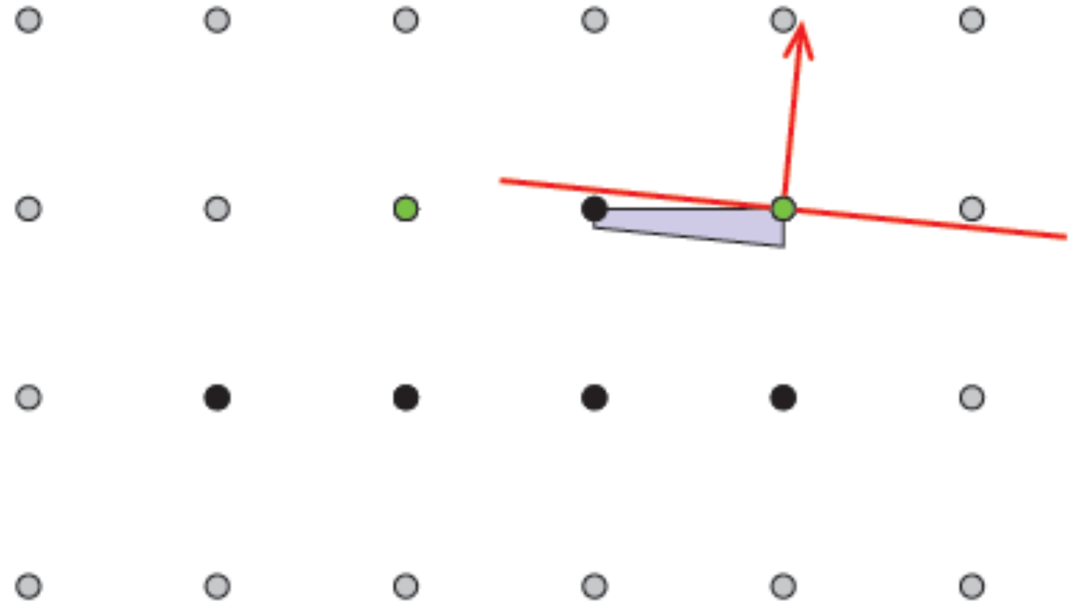


Bounds Strengthen



Solve LP

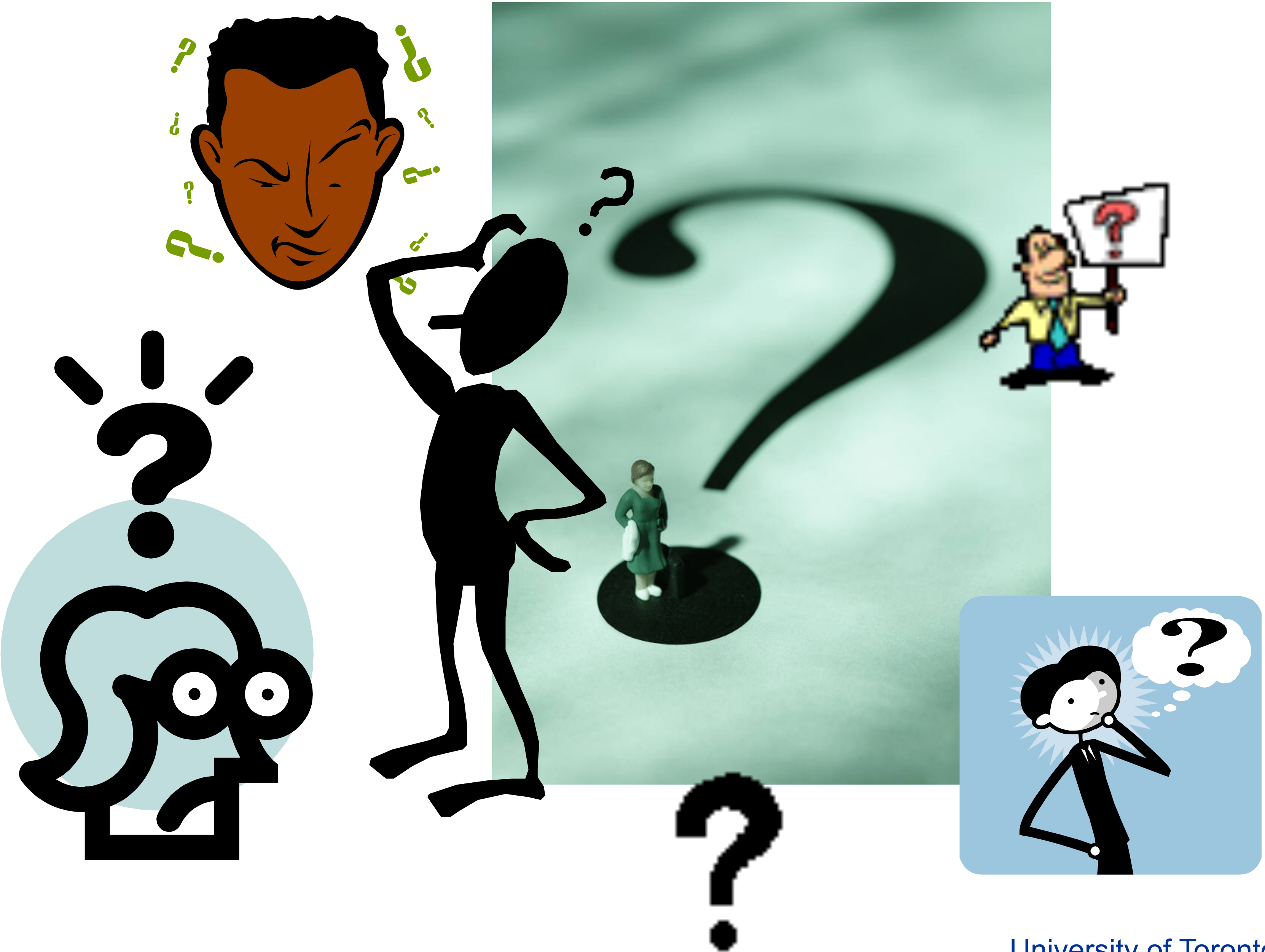
- New solution!



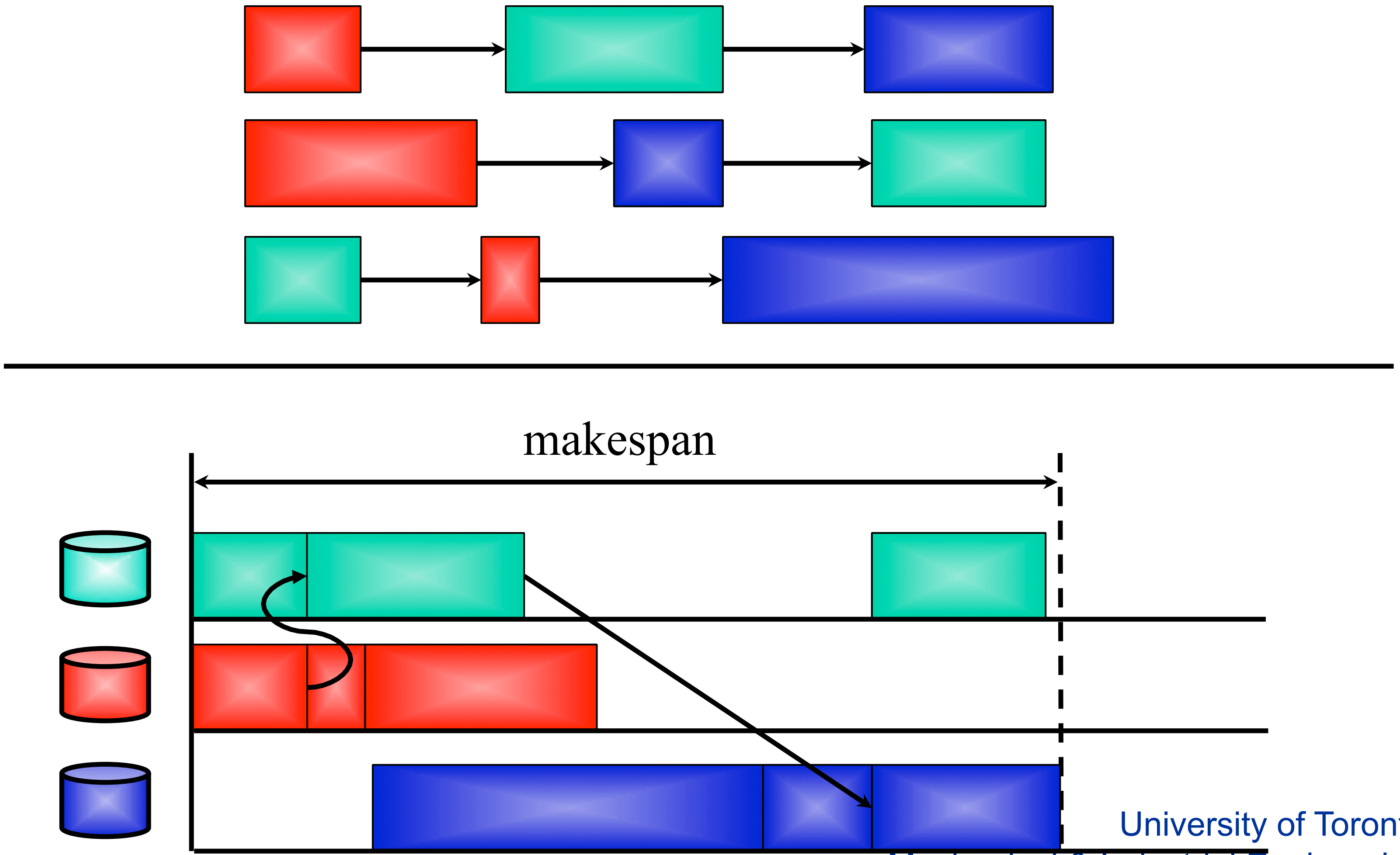
And we're done

Other Critical Solver Components

- Branching heuristics
- Cutting planes
- Primal heuristics
- Backtracking
  - Best-First Search or Depth-First Search



Job Shop Scheduling



MIP for Job Shop Scheduling

$$\min \quad C_{max}$$

$$\text{s. t.} \quad \sum_{k \in \mathcal{K}} \sum_{t \in H} x_{jkt} = 1$$

All activities start only once

$$\sum_{j \in \mathcal{J}} \sum_{t' \in T_{jkt}} x_{jkt'} \leq 1$$

Resource constraints

$$\sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{H}} (t + p_j) x_{jkt} \leq C_{max}$$

C_{max} is the largest end-time

$$\sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{H}} (t + p_j) x_{jkt} \leq \sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{H}} t x_{ikt}$$

Precedence constraints

$$x_{jkt} \in \{0, 1\} \qquad \forall k \in \mathcal{K}, \ \forall j \in \mathcal{J}, \ \forall t \in \mathcal{H}$$

Where:

- $\mathcal{H}$ is the set of all time-points
- $x_{jkt} = 1$ iff job j starts at time t on resource k
- $T_{jkt} = \{t - p_j, \ldots, t\}$.

Weaknesses?

What Makes MIP Different?

- Restricted language
  - cf. SAT
- Focus on the linear relaxation as the key technique to reduce search tree

Outline: Part 1

- What is Scheduling?
  - The fundamental bits
  - "The" classical problem
- Constraint Programming (CP)
  - Complete search and inference
- Mixed Integer Programming (MIP)
  - Complete search and relaxation
- Metaheuristics
  - Incomplete search

Now For Something Completely Different

Metaheuristics (aka Local Search)

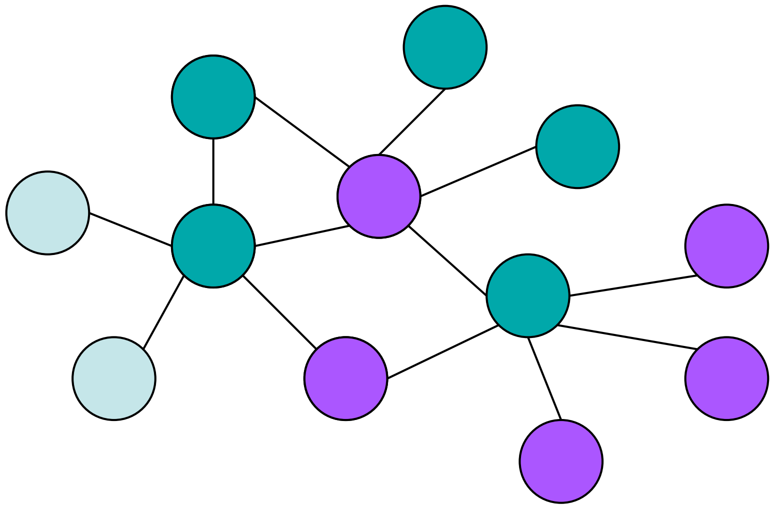
- Quickly & heuristically find a “good” solution
- Perturb the solution slightly, generating neighboring solutions
- Evaluate neighbors and move to the best one
- Repeat

Notation

- V is a set of variables $\{v_1, \ldots, v_n\}$
- s is an assignment of each variable to a value
- Let S be the set of all assignments
- A neighborhood N is a function from s to T where $T \subseteq S$

Notation

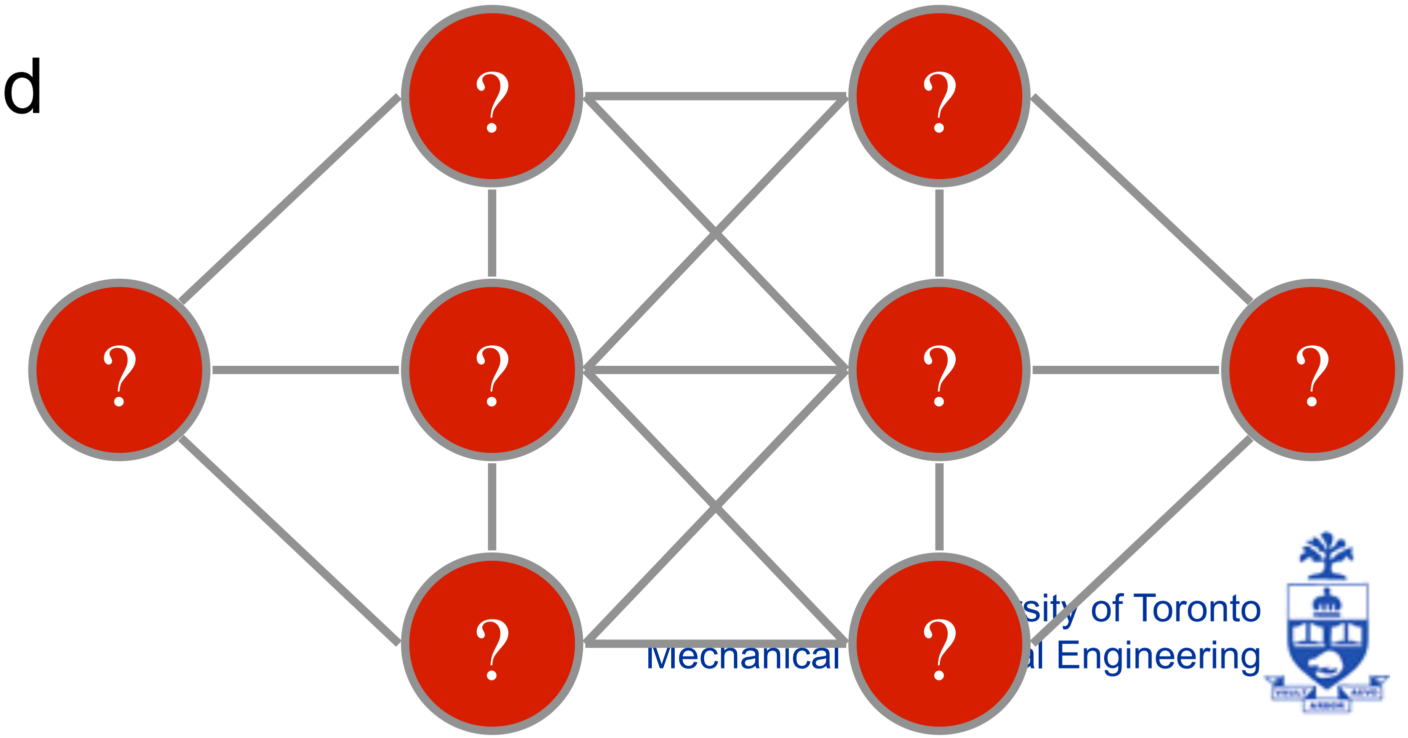
- So $N(s) \subseteq S$
- The assignments in N(s) are the "neighbors" of s



Crystal Maze

- Place the numbers 1 through 8 in the nodes such that:
  - Each number appears exactly once

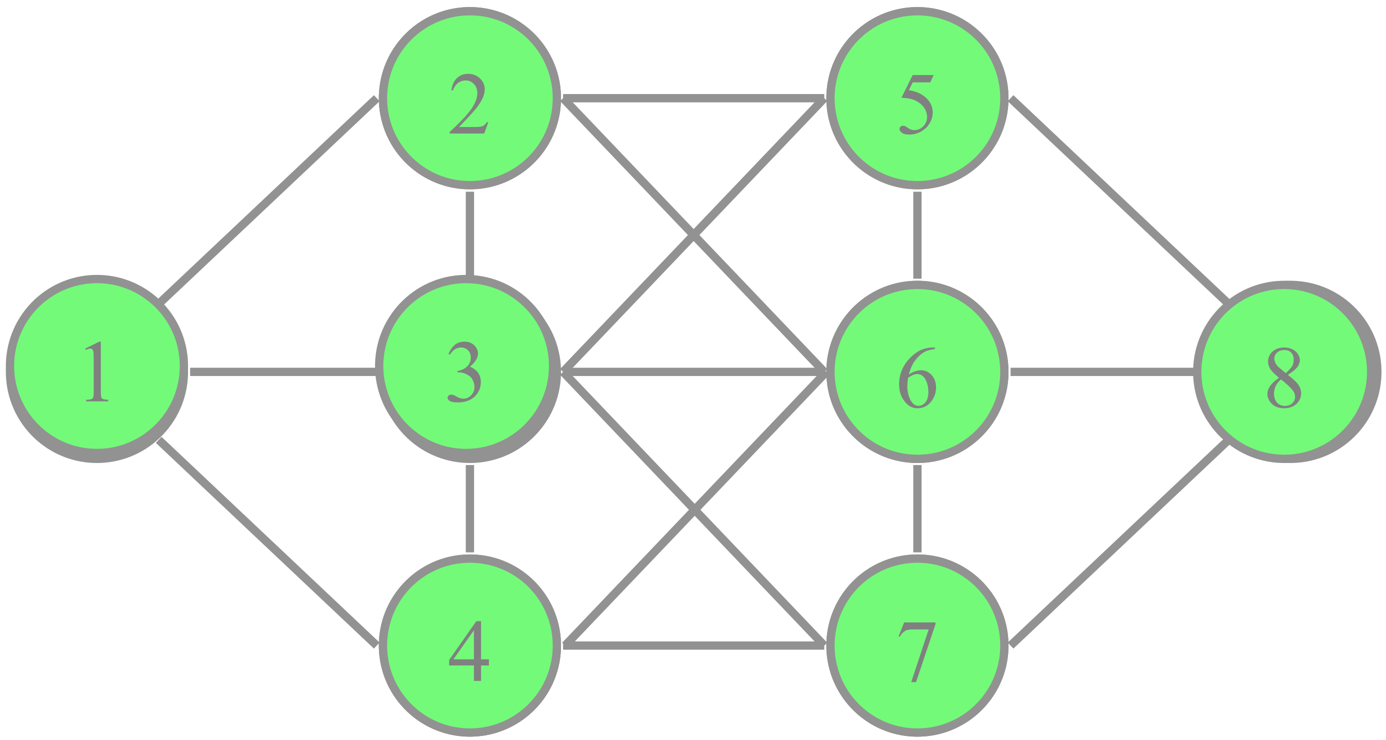
  - No connected nodes have consecutive numbers



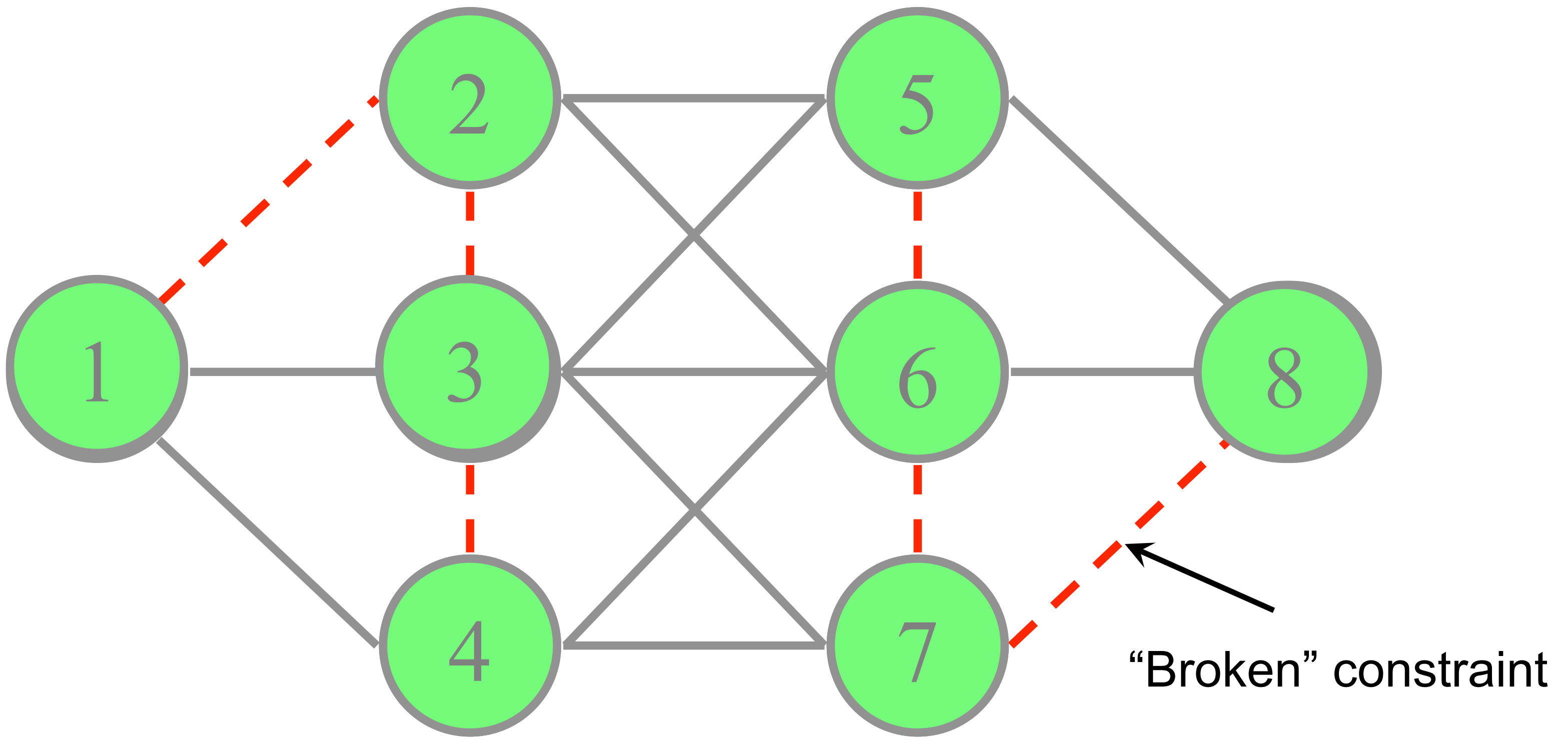
Local Search Idea

- Randomly assign values (even if the constraints are "broken")
  - Initial state will probably be infeasible
- Make "moves" to try to move toward a solution

Random Initial Solution



Random Initial Solution



Cost = # of broken constraints

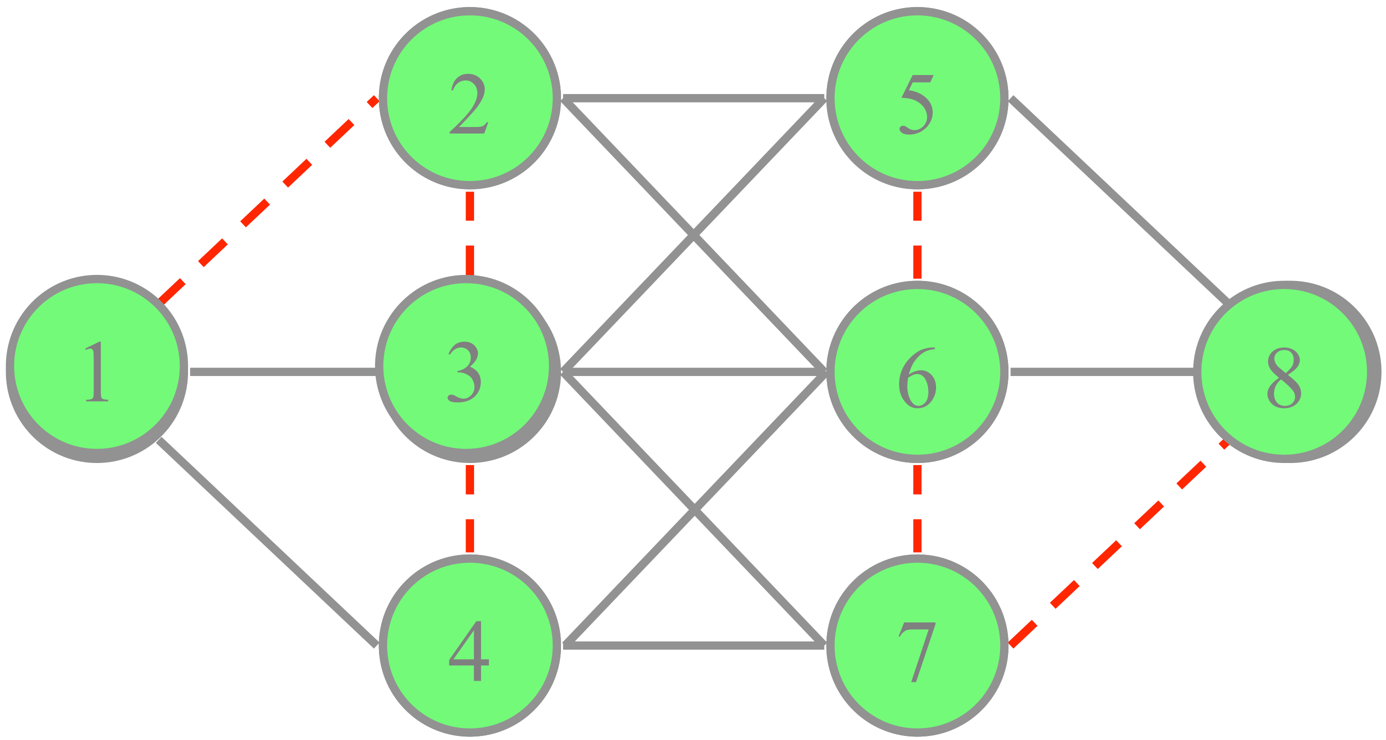
What Should We Do Now?

- Move:
  - Swap two numbers
- Which two numbers?
  - Randomly pick a pair
  - The pair that will lead to the biggest decrease in cost
    - Cost: number of broken constraints

What Should We Do Now?

- Move:
  - Swap two numbers
- Which two numbers?
  - Randomly pick a pair
  - The pair that will lead to the biggest decrease in cost
    - Cost: number of broken constraints

Random Initial Solution



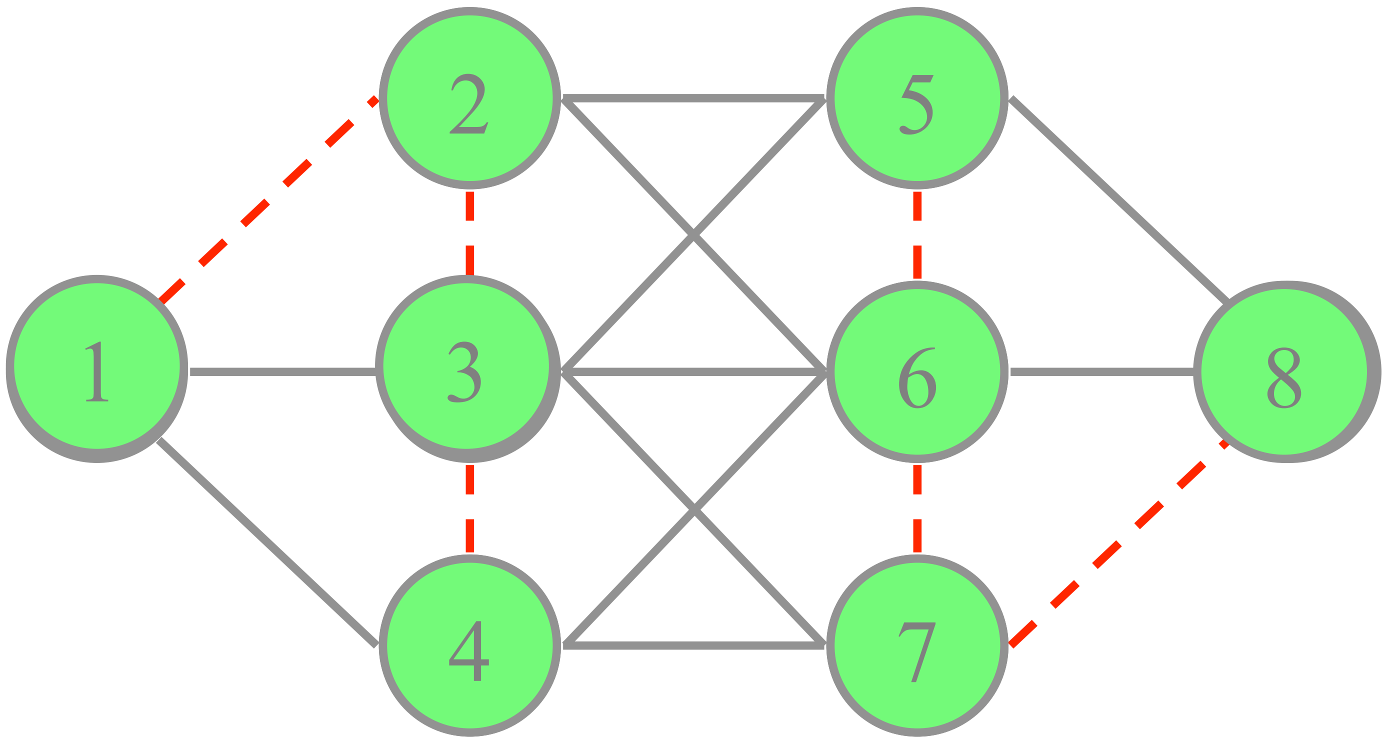
Cost Difference Table

	1	2	3	4	5	6	7	8
1	0	0	0	-1	0	-2	-3	-2
2		0	-1	1	-1	-2	-1	-3
3			0	0	0	0	-1	0
4				0	0	0	-1	0
5					0	0	1	-1
6						0	-1	0
7							0	0
8								0

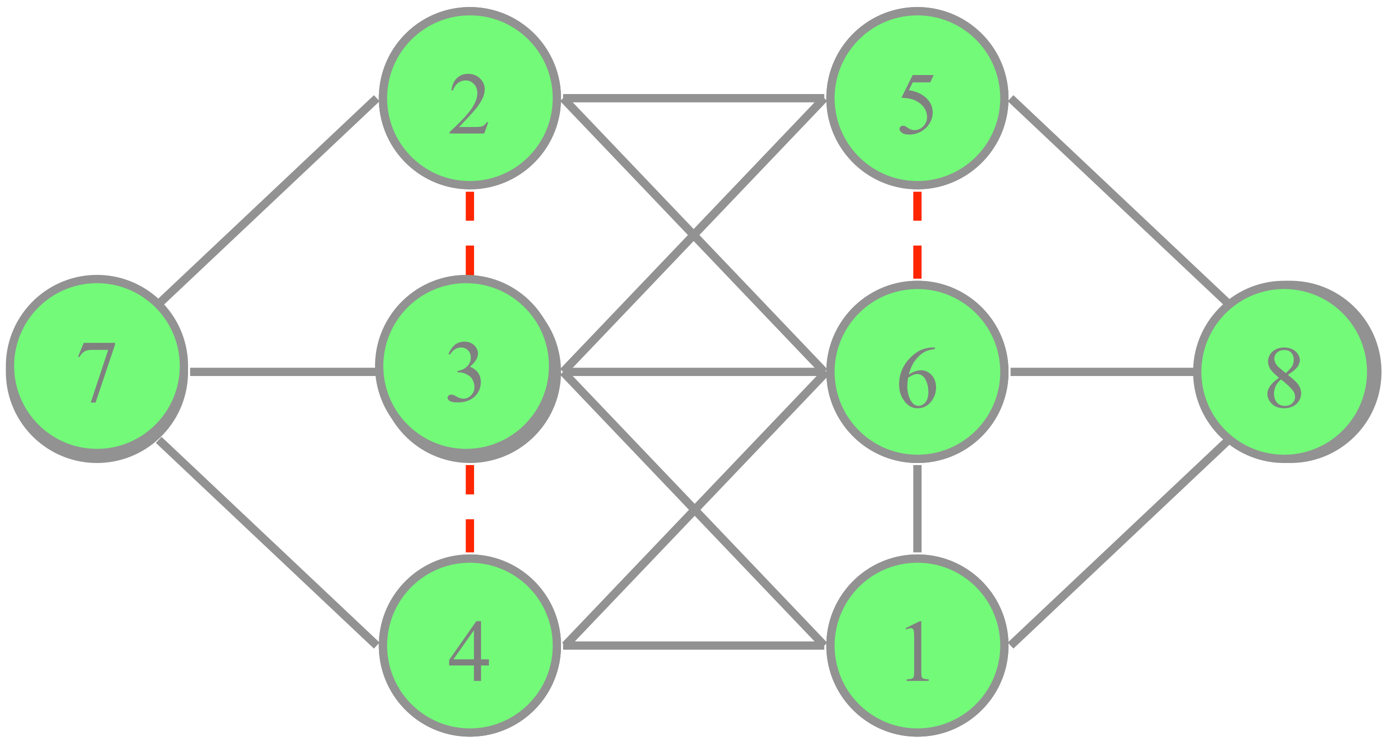
Cost Difference Table

	1	2	3	4	5	6	7	8
1	0	0	0	-1	0	-2	-3	-2
2		0	-1	1	-1	-2	-1	-3
3			0	0	0	0	-1	0
4				0	0	0	-1	0
5					0	0	1	-1
6						0	-1	0
7							0	0
8								0

Current State



Swap 1 & 7: Cost 3

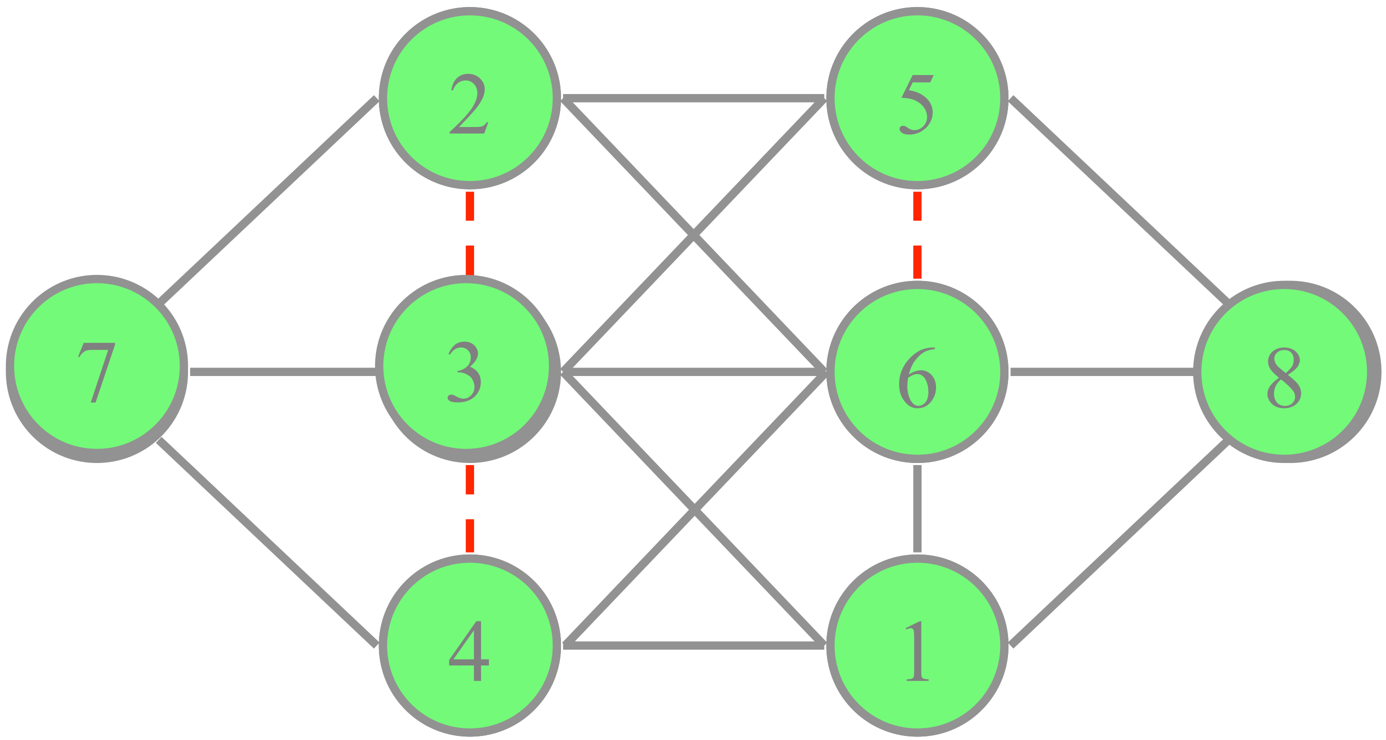


New Cost Difference Table

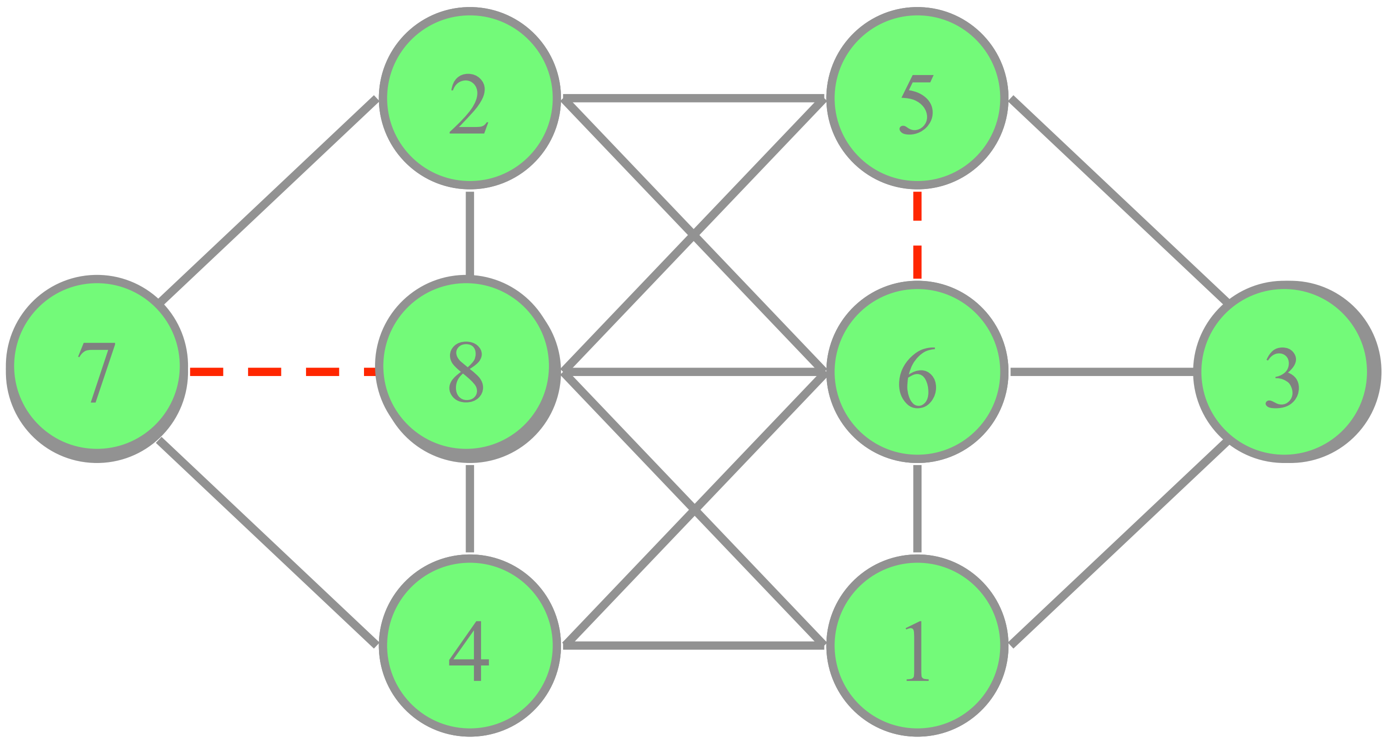
Incremental updates are important

	1	2	3	4	5	6	7	8
1	0	0	0	0	2	0	3	0
2		0	0	2	0	1	1	1
3			0	0	0	1	1	-1
4				0	0	1	1	1
5					0	1	2	0
6						0	0	0
7							0	1
8								0

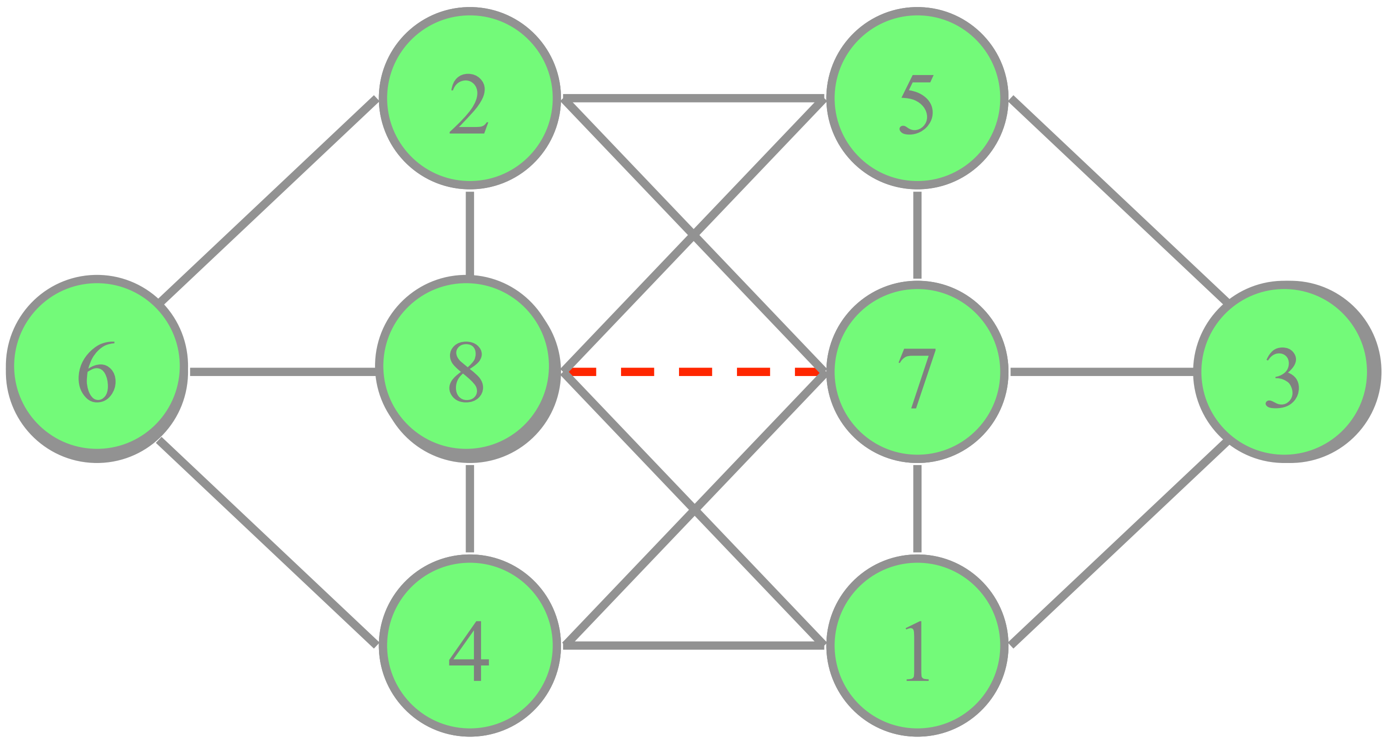
Current State



Swap 3 & 8: Cost 2



Swap 6 & 7: Cost 1



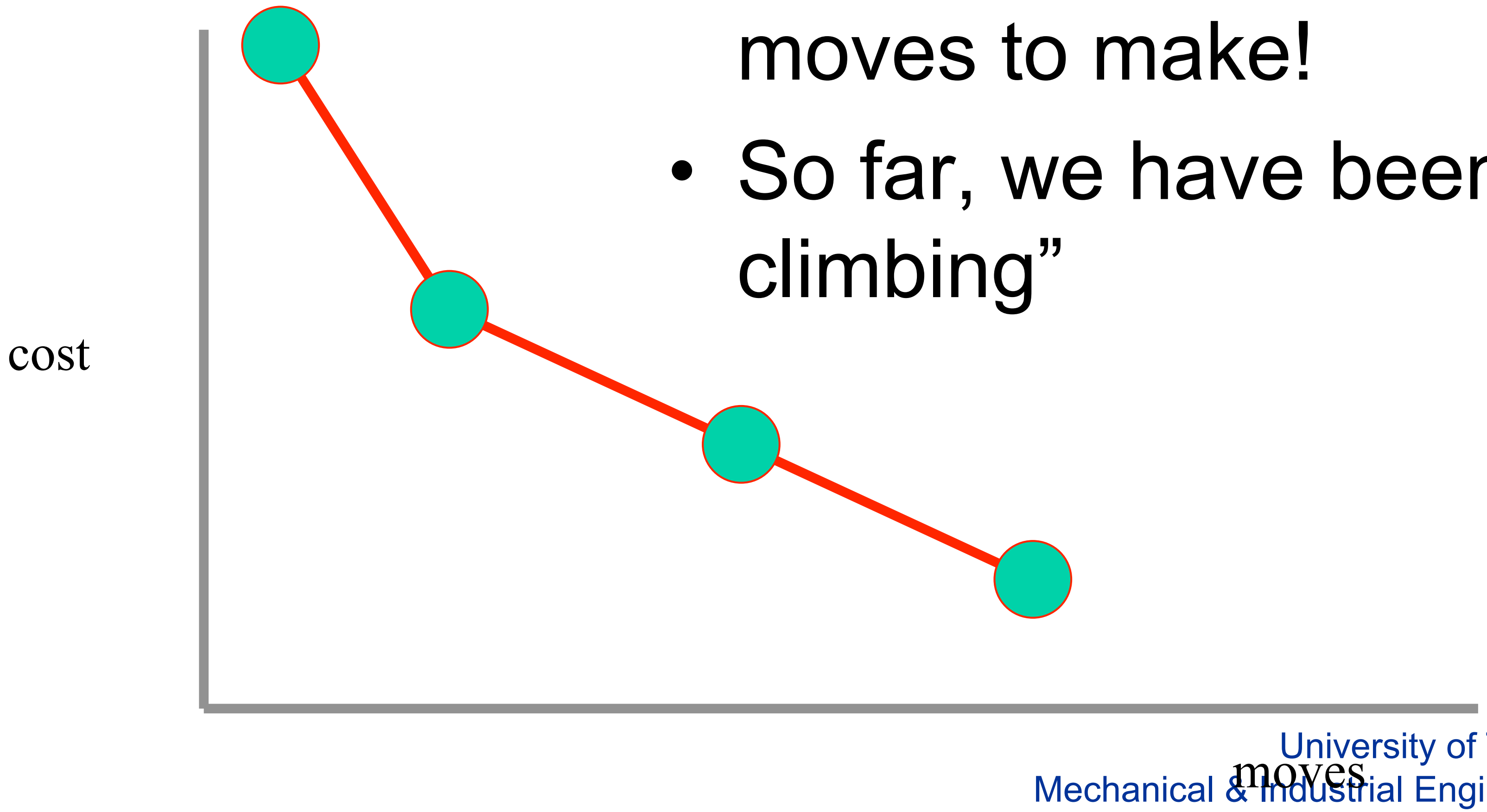
Moves

- Initial State: Cost 6
- Swap 1 & 7: Cost 3
- Swap 3 & 8: Cost 2
- Swap 6 & 7: Cost 1

Cost Difference Table

	1	2	3	4	5	6	7	8
1	0	1	1	1	2	2	1	1
2		0	1	2	2	1	3	1
3			0	1	1	4	1	2
4				0	2	1	3	1
5					0	2	1	2
6						0	1	1
7							0	1
8								0

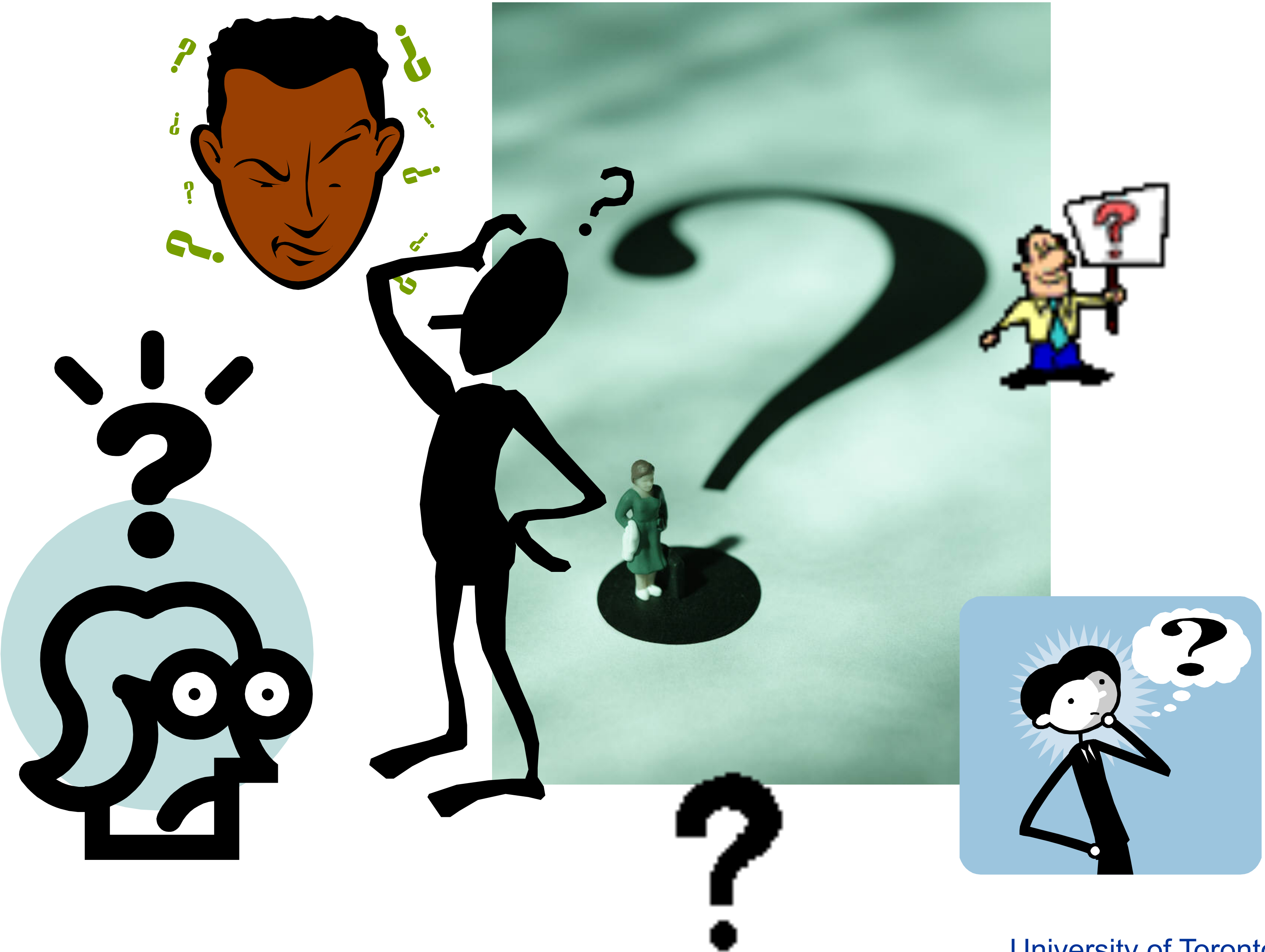
Now what?



- There are no improving moves to make!
- So far, we have been "hill-climbing"

Now what?

- This is when you need a metaheuristic
  - Simulated Annealing
  - Tabu Search
- [Blum & Roli 2003]



Local Search (or Iterative Improvement or Hill-Climbing)

$s \leftarrow$ GenerateInitialSolution()
repeat
  $s \leftarrow$ Improve($\mathcal{N}(s)$)
until no improvement is possible

OR

first improvement (aka first accept)

best improvement (aka best accept)

Fig. 1. Algorithm: Iterative Improvement.

There is a lot that has been left unsaid!

Simulated Annealing

$s \leftarrow$ GenerateInitialSolution()
$T \leftarrow T_0$ ← "temperature"
while termination conditions not met **do**
  $s' \leftarrow$ PickAtRandom($\mathcal{N}(s)$)
  if $(f(s') < f(s))$ **then** ← f(s) is the cost of solution s
    $s \leftarrow s'$ % s' replaces s
  else
    Accept s' as new solution with probability $p(T, s', s)$
  endif
  Update(T) ← "cooling schedule"
endwhile

Fig. 2. Algorithm: Simulated Annealing (SA).

Probability of Acceptance

- Typically:

$$p(T, s', s) = \exp(-\frac{f(s') - f(s)}{T})$$

- at a fixed T, the higher the difference in cost the lower the prob. of acceptance
- at a fixed cost difference, the higher the temperature, the higher the prob. of acceptance

Cooling Schedule

- Typically the temperature starts out high and gradually decreases
- A lot of theoretical work here
- Often, in practice

$$T_{k+1} = \alpha T_k$$

$$\alpha \in (0,1)$$

Tabu Search

Could also do first instead of best

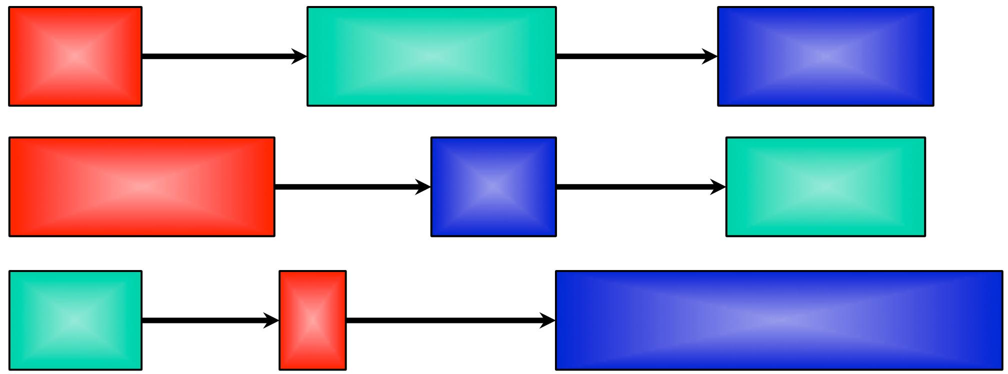
$s \leftarrow$ GenerateInitialSolution()
$TabuList \leftarrow \emptyset$
while termination conditions not met **do**
  $s \leftarrow$ ChooseBestOf($\mathcal{N}(s) \setminus TabuList$)
  Update($TabuList$)
endwhile

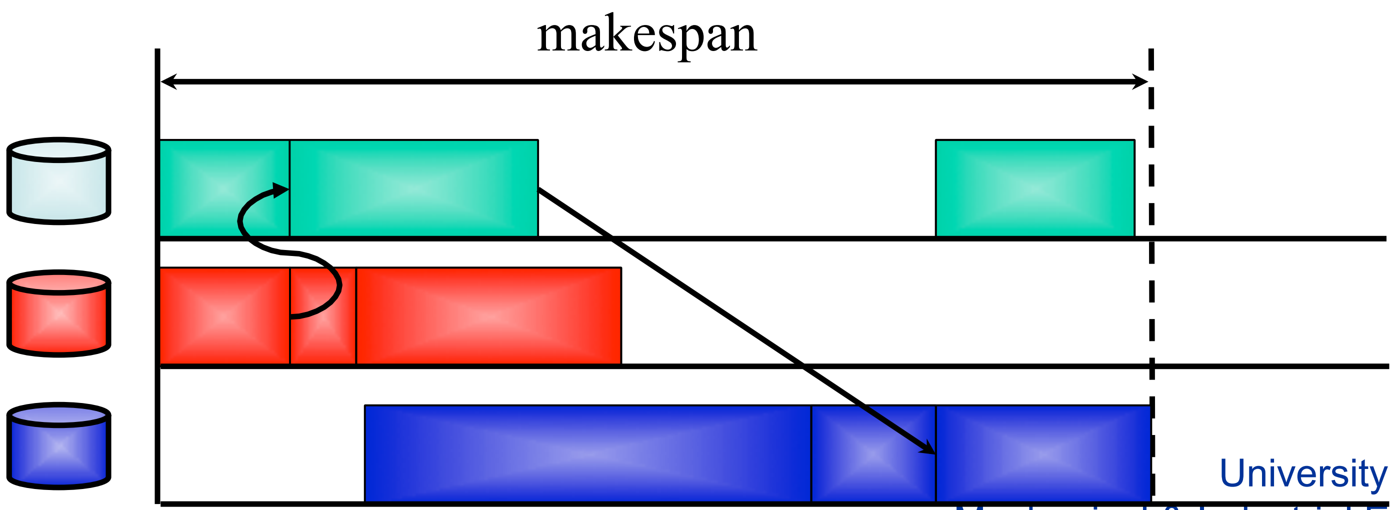
Fig. 3. Algorithm: Simple Tabu Search (TS).

Tabu List

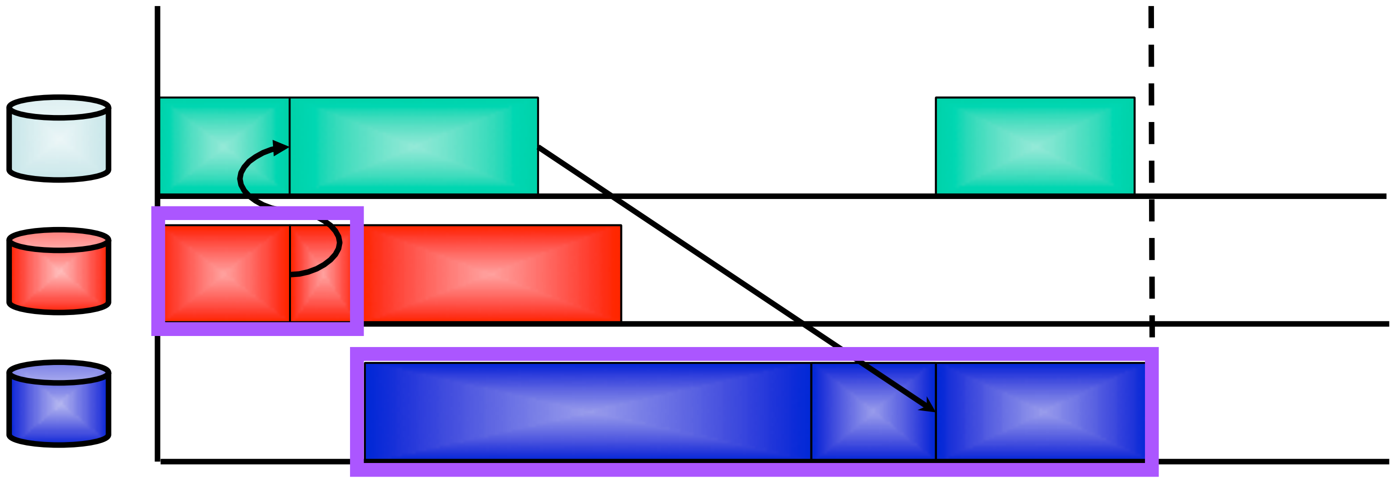
- What is the format of an element?
- What is the tabu tenure?
  - Variations?
- What are aspiration criteria?

Job Shop Scheduling





Critical Path



A critical “block” is a contiguous set of critical activities on the same resource

nufacturing Scheduling

page

emo

oblem

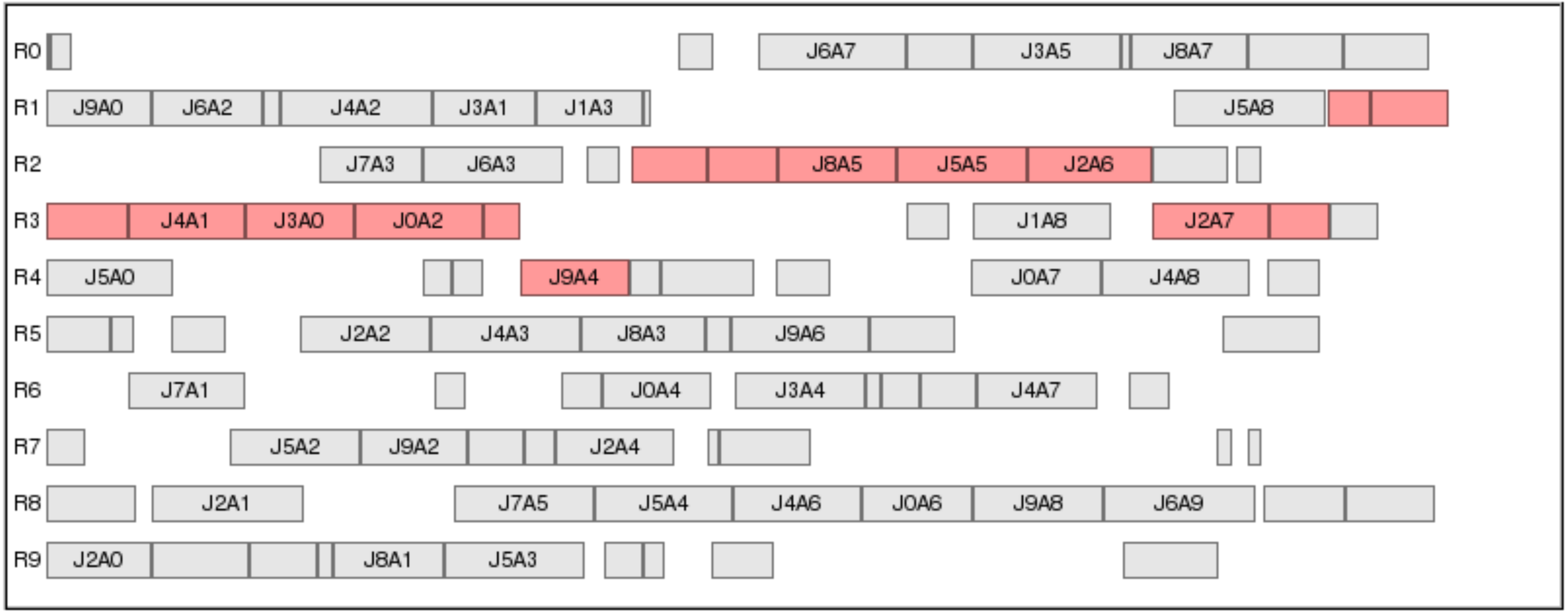
ions

esigned to show
traint
can be used to
scheduling

Finished

(Problem Category: 10x10 - Filename: jsp0 - Time Limit : 10) Prev Best Next

Initial Cost = 974 Makespan = 903 Improvement = 7% Scaling to fit (80%)



Program Output

5 Time: 9.2216 Makespan: 898 Failures: 0 ChPts: 100 Random: 45 Activities: 45 Models: 306

6 Time: 9.2246 Makespan: 897 Failures: 29 ChPts: 124 Random: 45 Activities: 45 Models: 306

7 Time: 9.91249 Makespan: 896 Failures: 0 ChPts: 85 Random: 45 Activities: 45 Models: 328

improvement summary

N1 & N5 Neighborhoods

- N1: Swap all pairs of adjacent activities in a critical block

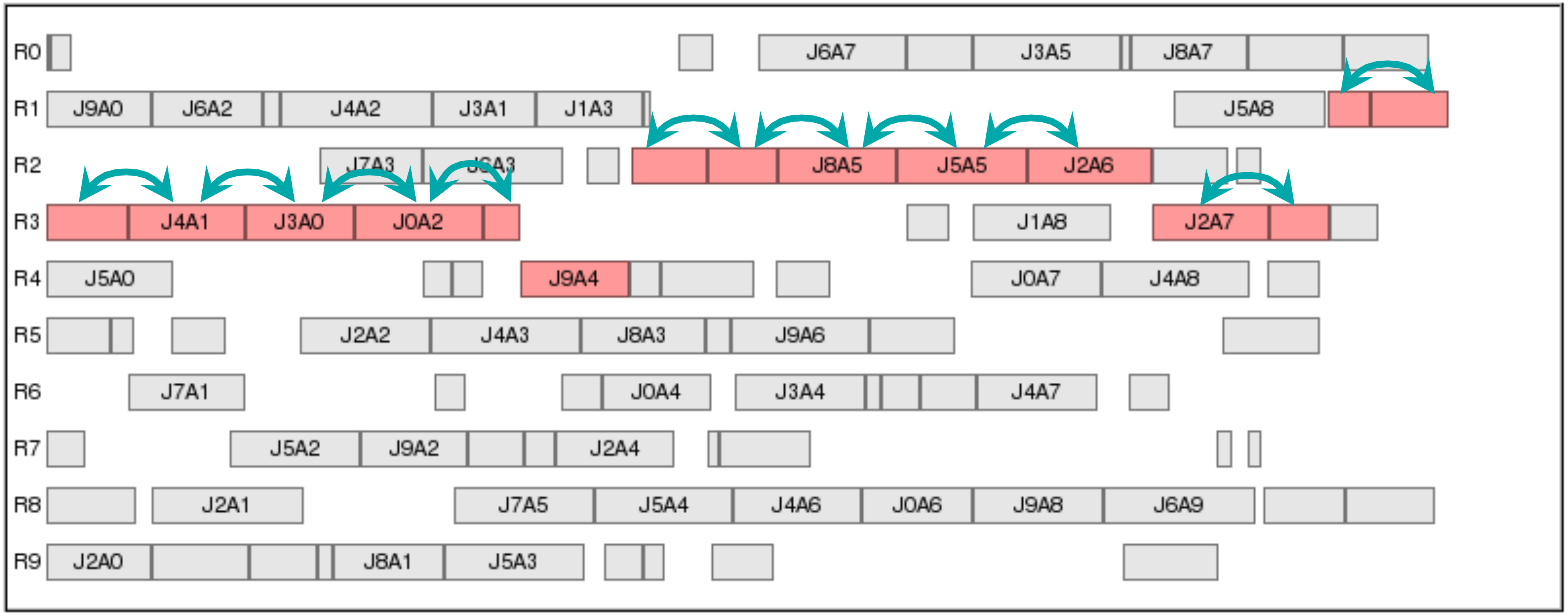
nufacturing Scheduling

esigned to show
traint
can be used to
scheduling

N1 Neighborhood
10 neighbors

Prev | Best | Next

Initial Cost = 974 | Makespan = 903 | Improvement = 7% | Scaling to fit (80%)



Program Output

5 Time: 9.2216 Makespan: 898 Failures: 0 ChPts: 100 Random: 45 Activities: 45 Models: 306

6 Time: 9.2246 Makespan: 897 Failures: 29 ChPts: 124 Random: 45 Activities: 45 Models: 306

7 Time: 9.91249 Makespan: 896 Failures: 0 ChPts: 85 Random: 45 Activities: 45 Models: 328

improvement summary

N1 & N5 Neighborhoods

- N1: Swap all pairs of adjacent activities in a critical block
- N5: Swap first and last adjacent pair in each critical block
  - but only last pair in first block and first pair in last block

nufacturing Scheduling

N5 Neighborhood
5 neighbors

page

emo

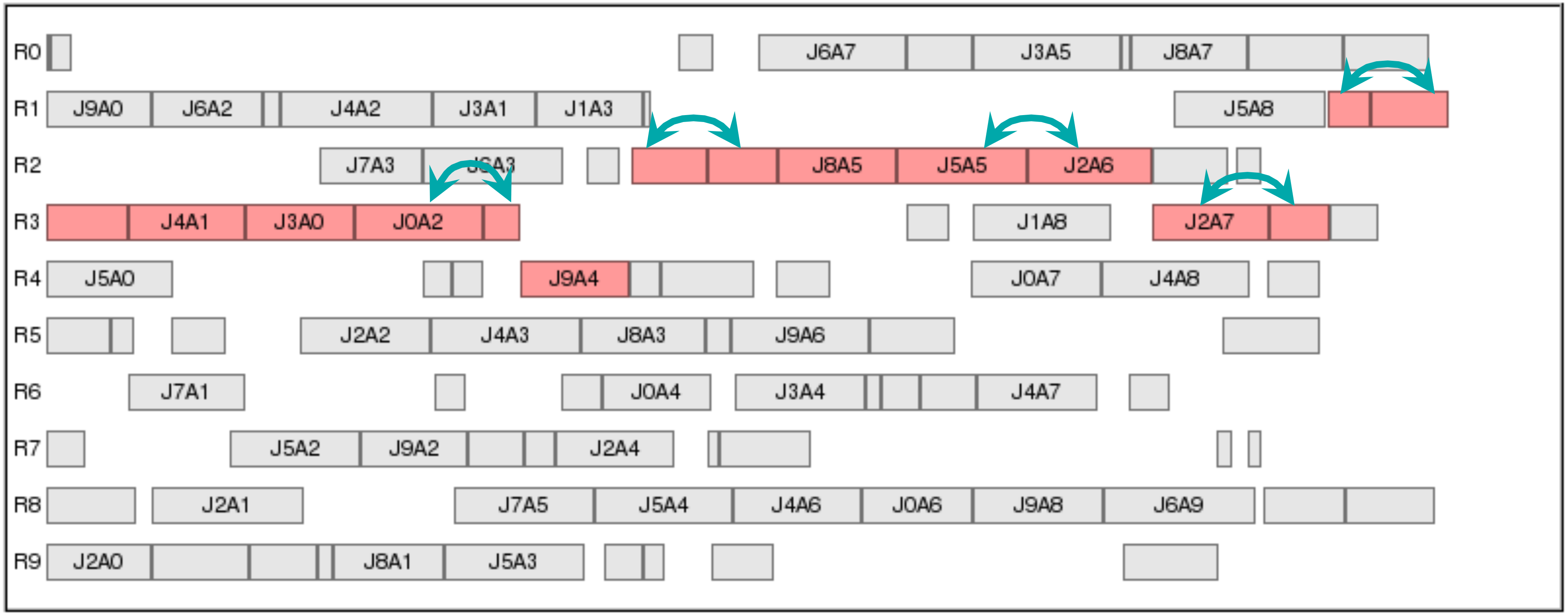
oblem

ions

esigned to show
traint
can be used to
scheduling

Prev | Best | Next

Initial Cost = 974 Makespan = 903 Improvement = 7% Scaling to fit (80%)



Program Output

5 Time: 9.2216 Makespan: 898 Failures: 0 ChPts: 100 Random: 45 Activities: 45 Models: 306

6 Time: 9.2246 Makespan: 897 Failures: 29 ChPts: 124 Random: 45 Activities: 45 Models: 306

7 Time: 9.91249 Makespan: 896 Failures: 0 ChPts: 85 Random: 45 Activities: 45 Models: 328

improvement summary

Simple Tabu Search (STS)

- Tabu tenure
  - randomly drawn from an interval [6,10] every 15 moves
- Elite solutions
  - maintain *e* elite solutions
  - if best solution hasn't improved in a while, jump back to one of the elite solutions and start over
- Other sophisticated components

Metaheuristics

- Start with random or heuristic solution
- Make moves following the cost gradient
  - Might need some short term memory (e.g., tabu list) to avoid cycling
- Go until you find a solution or reach a bound on the number of moves

Summary

- CP
  - search + inference, rich language, domain specific inference and heuristics
- MIP
  - search + relaxation, restricted language, generic relaxation
- Metaheuristics
  - local search
  - hill-climbing + local minima escape

Which is Best?

- CP
  - scheduling is a (commercial) success story for CP
  - easy to add side-constraints (and there are always side constraints)
- However:
  - if propagation is weak, falls apart
    - more complicated cost functions or multiple decisions need to be made before inference can work
  - scaling?

Which is Best?

- MIP
  - good with complex costs
  - flexible modeling of side-constraints
    - if they are linear
- However:
  - scaling issues with time-indexed formulation
  - if resource feasibility is the main challenge, falls apart
    - especially with non-unary resources

Which is Best?

- Metaheuristics
  - can be highly customized for a given problem
  - scales well
  - state-of-the-art for JSP since mid-90s
- However:
  - hard to incorporate side-constraints
    - need new neighborhood
  - can’t prove optimality or even give a bound on solution quality

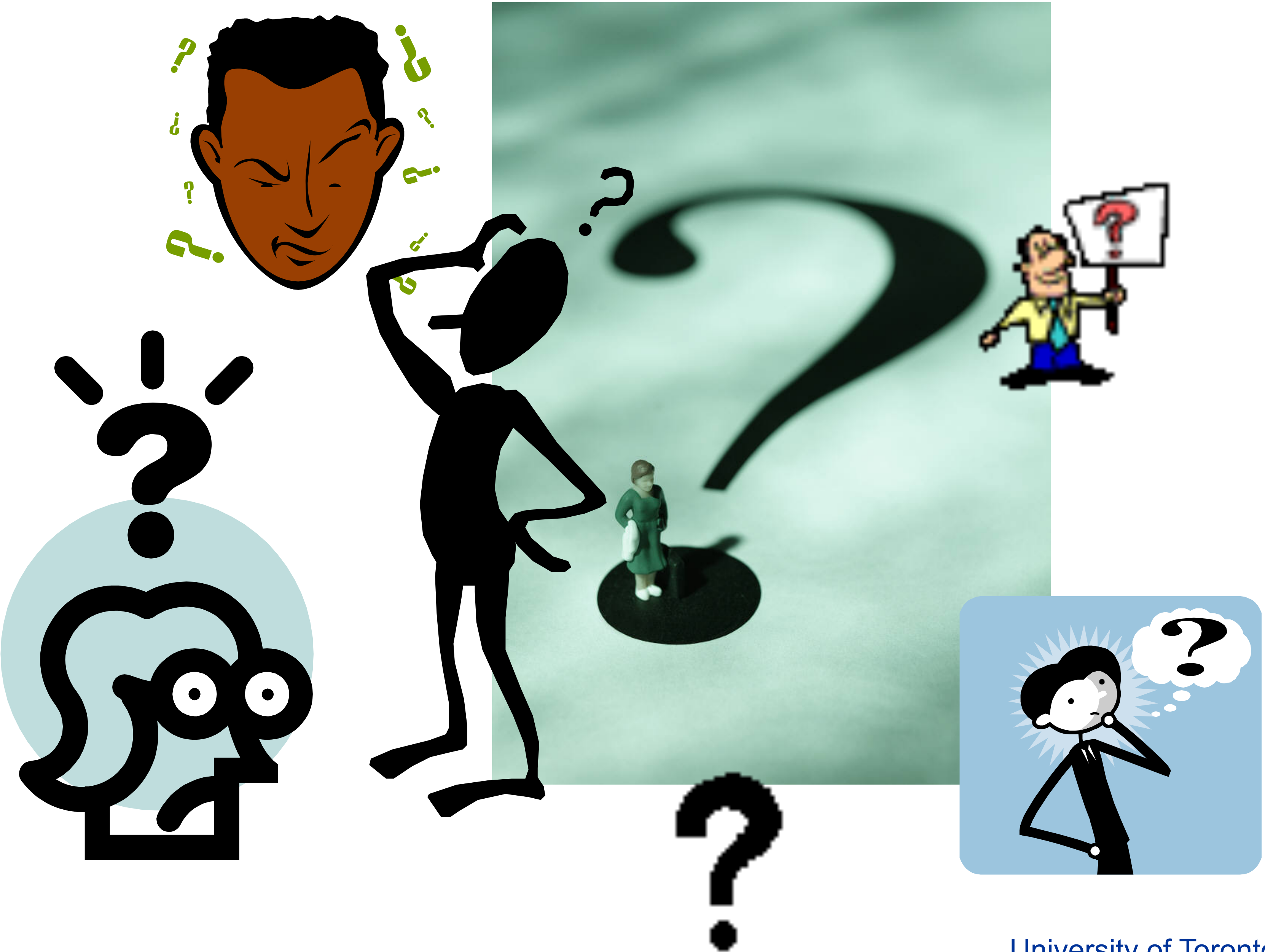
Outline: Part 2

- Remembering Yesterday
- Combining CP and Tabu Search for Job Shop Scheduling
- Combining MIP and CP for Resource Allocation/Scheduling Problems

Outline: Part 3

- The Origin of the Species
  - – Ancient History (the 70s & 80s)
  - – What's a constraint anyway?
- The 90s
- Scheduling & AI

Please come back tomorrow



Is Scheduling Still AI?
Part 2: State of the Art

J. Christopher Beck
Dept. of Mechanical & Industrial Engineering
University of Toronto
Canada

ACAI Summer School
Freiburg, Germany
June 7 – 10, 2011

Outline

- Part 1: Core Scheduling Technologies
  - CP, MIP, & Metaheuristics
  - 90 minutes
- Part 2: State of the Art
  - CP + Metaheuristics, CP + MIP
  - 60 minutes
- Part 3: Polemics & Perspectives
  - The Past and the Future?
  - 30 minutes

Outline: Part 2

- Remembering Yesterday
  - 90 minutes in 3 slides
- Combining CP and Tabu Search for Job Shop Scheduling
- Combining MIP and CP for Resource Allocation/Scheduling Problems

Scheduling is …

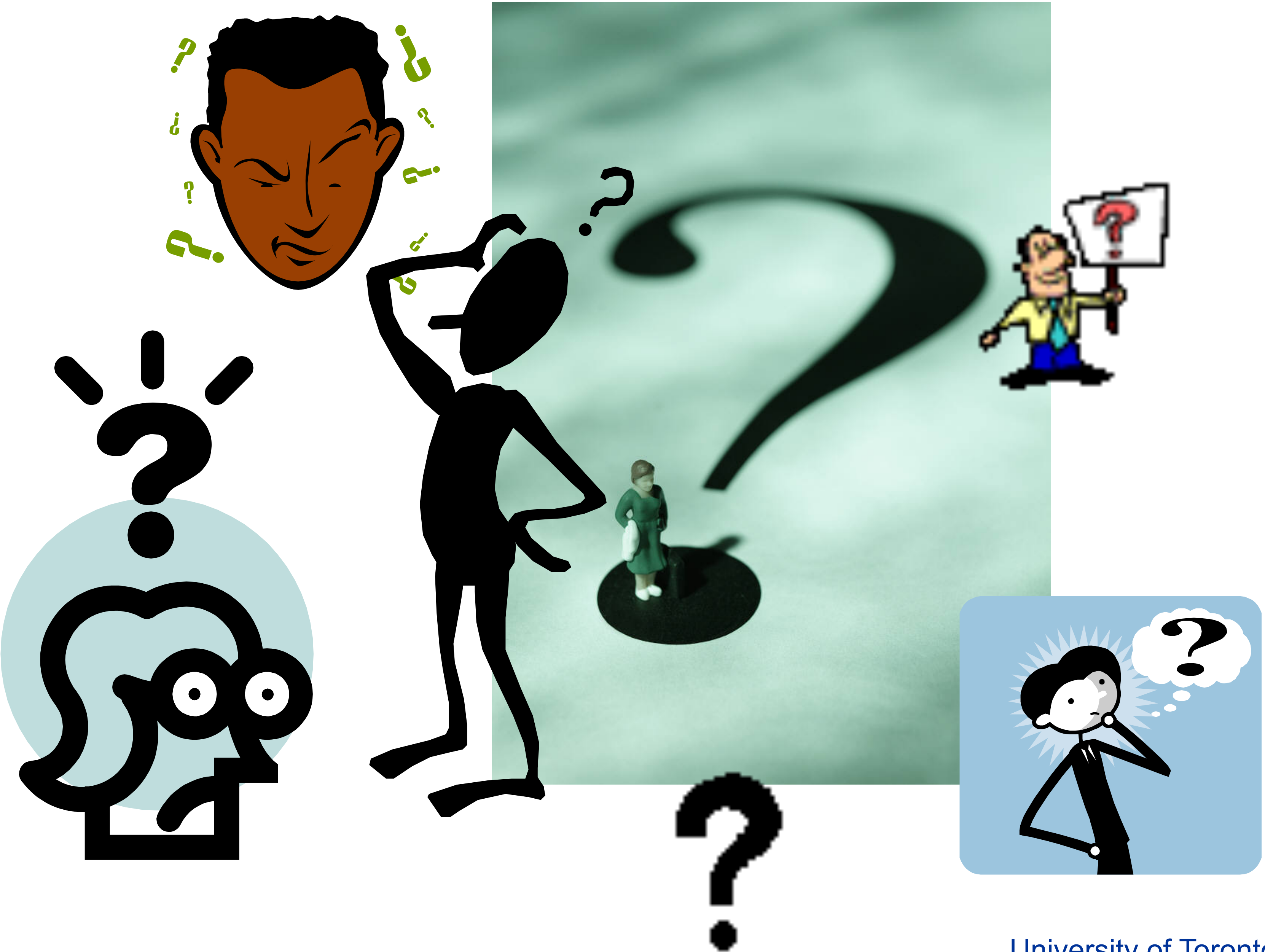
- The allocation of resources to activities over time
  - Mixing machines in food manufacturing
  - Classrooms at a university
  - Trucks & planes for FedEx
- Mathematically hard
- Industrially, economically, & environmentally important

The Key Difference with Planning

- In classical scheduling we know all the operations (e.g., flights, production jobs) at the beginning of the solving process
- Typically (and for this lecture) assume that we never add to the set of operations during search

Summary

- CP
  - search + inference, rich language, domain specific inference and heuristics
- MIP
  - search + relaxation, restricted language, generic relaxation
- Metaheuristics
  - local search
  - hill-climbing + local minima escape



Outline: Part 2

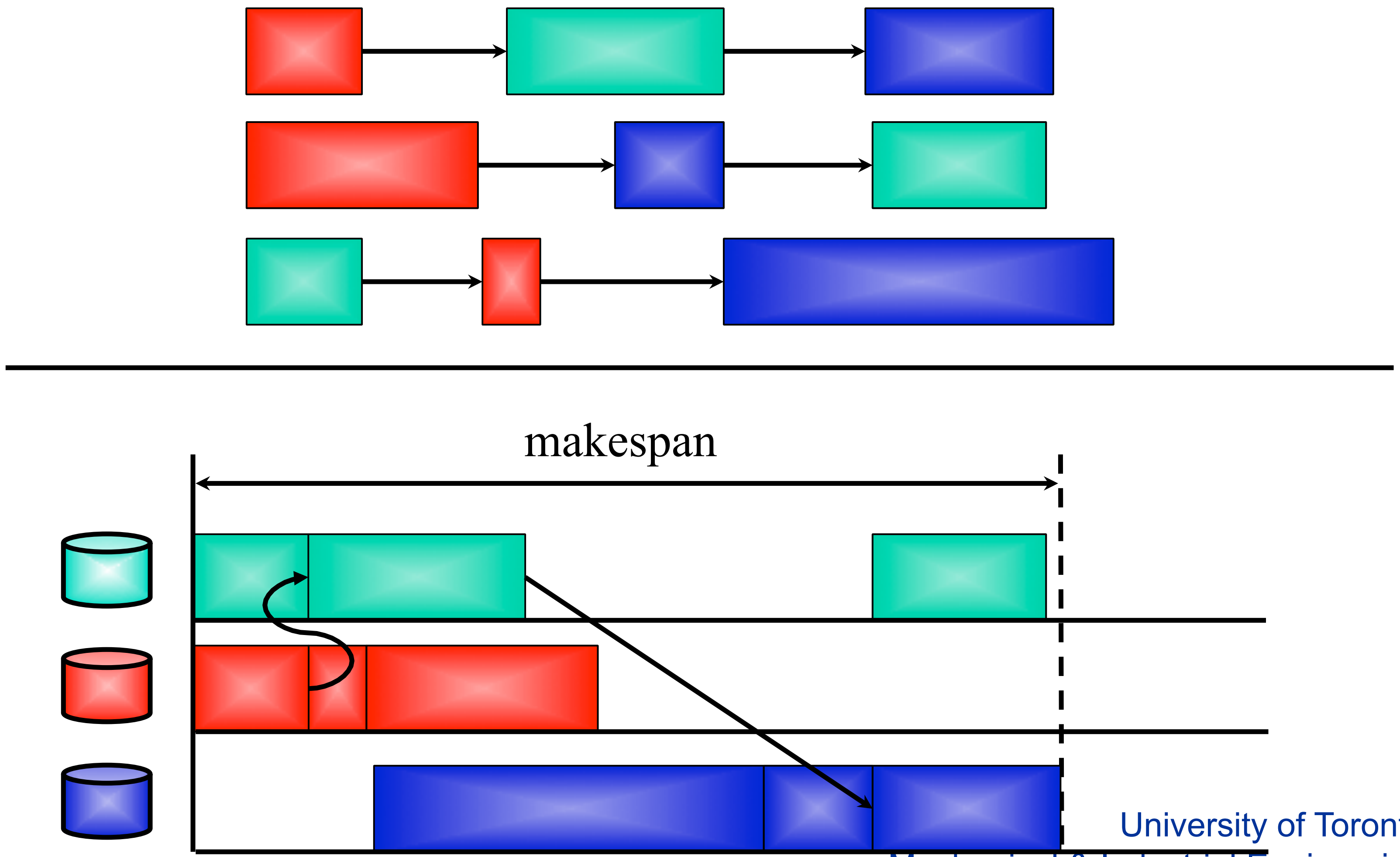
- Remembering Yesterday
  - 90 minutes in 3 slides
- Combining CP and Tabu Search for Job Shop Scheduling
- Combining MIP and CP for Resource Allocation/Scheduling Problems

Outline

- State of the Art in Job Shop Scheduling
- Iterated Simple Tabu Search (i-STS) & Solution-Guided Search (SGS)
- Hybrid i-STS/SGS

[B., Feng, & Watson, 2011]
Combining Constraint Programming and Local Search for Job-Shop Scheduling.
INFORMS Journal on Computing, **23(1),** 1-14, 2011.

Job Shop Scheduling



A CP Model for JSP

$$
\begin{aligned}
\min \quad & C_{max} && \\
\text{s.t.} \quad & S_j \geq 0 && \forall j \in \mathcal{J} \\
& S_j + p_j \leq C_{max} && \forall j \in \mathcal{J} \\
& S_j + p_j \leq S_i && \forall (j,i) \in \mathcal{E} \\
& \texttt{disjunctive}(\boldsymbol{S}_{\cdot}, \boldsymbol{p}_{\cdot}) && \forall k \in \mathcal{K} \\
& S_j \in \mathbb{Z} && \forall j \in \mathcal{J}.
\end{aligned}
$$

Minimize the makespan

All activities end before the makespan

Precedence constraints

Where:

- $\mathcal{J}$ is the set of all activities
- $\mathcal{K}$ is the set of all resources
- $\mathcal{E}$ is set of all precedence constraints
- S_j is the start-time variable of job j
- p_j is the processing time of job j

State of the Art for JSP

- TSAB (Nowicki and Smutnicki 1993, 1996)
  - Elite pool of *k* best solutions found
  - Repeated tabu search from elite solutions
- i-TSAB (Nowicki and Smutnicki 2001, 2002, 2003, 2005)
  - Elite pool of *k* best solutions
  - Path relinking to diversify, TSAB to intensify
- Tabu search / simulated annealing hybrid (Zhang et al. 2006)

State of the Art for JSP

- Constraint Programming
  - Sophisticated propagation techniques
  - Scheduling specific heuristics
  - Commercially successful in scheduling → easily model side-constraints

However

Doesn't really compete on the JSP

Taillard's 20x20 JSPs - Makespan

Instance	UB	CP - chron	CP - restart mean (best)
TA21	1644	1809	1694 (1686)
TA22	1600	1689	1654 (1649)
TA23	1557	1657	1614 (1602)
TA24	1646	1810	1698 (1694)
TA25	1595	1685	1673 (1664)
TA26	1645	1827	1707 (1701)
TA27	1680	1827	1755 (1750)
TA28	1603	1778	1664 (1656)
TA29	1625	1718	1666 (1660)
TA20	1584	1666	1647 (1641)

Outline

- State-of-the Art in Job Shop Scheduling
- Iterated Simple Tabu Search (i-STS) & Solution-Guided Search (SGS)
- Hybrid i-STS/SGS

Tabu Search

$s \leftarrow$ GenerateInitialSolution()
$TabuList \leftarrow \emptyset$
while termination conditions not met **do**
  $s \leftarrow$ ChooseBestOf($\mathcal{N}(s) \setminus TabuList$)
  Update($TabuList$)
endwhile

Fig. 3. Algorithm: Simple Tabu Search (TS).

[Blum & Roli, 2006]
Metaheuristics in combinatorial optimization:
Overview and conceptual comparison.
ACM Computing Surveys, 35(3):268-308, 2003.

i-STS: Initial Phase

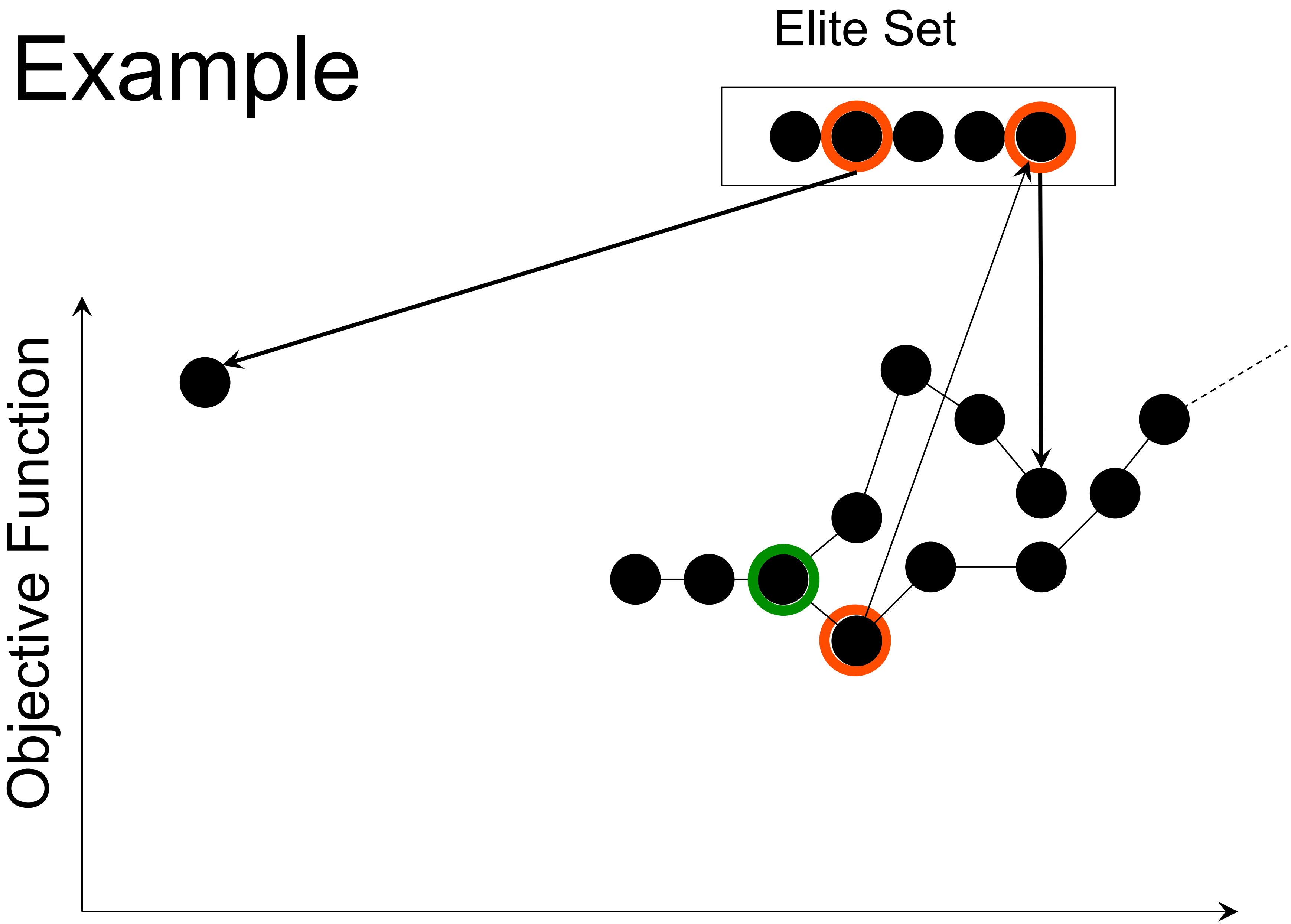
- Repeat *k* times
  – Generate random local optima, A
  – Run STS on A until no more progress is being made
  – Insert the best solution found in the STS run into the elite set

[Watson, Howe, & Whitley 2006]
Deconstructing Nowicki and Smutnicki's *i*-TSAB tabu search algorithm for the job-shop scheduling problem. *Computers and Operations Research*, **33,** 2623–2644, 2006.

i-STS: Proper Work Phase

- With 0.5 probability
  - Pick random elite solution, R, and run STS
  - If best solution is better than R, replace R
- Else
  - Pick two random elite solutions, R, S
  - Walk half way between R & S to W
  - Run STS from W
  - If best solution is better than R, replace R

Example



Another View

Starting Solution	10010011111010
Guiding Solution	10111010101011

Another View

Starting Solution

1001001111010

1001001111011

1011001111010

1001101111010

1001001011010

1001001110101 0

1001001111011

Guiding Solution

10111010101011

University of Toronto
Mechanical & Industrial Engineering

Another View

Starting Solution

10010011111010

10010011111011

10010010111011

10111001111011

10011101111011

10010010111011

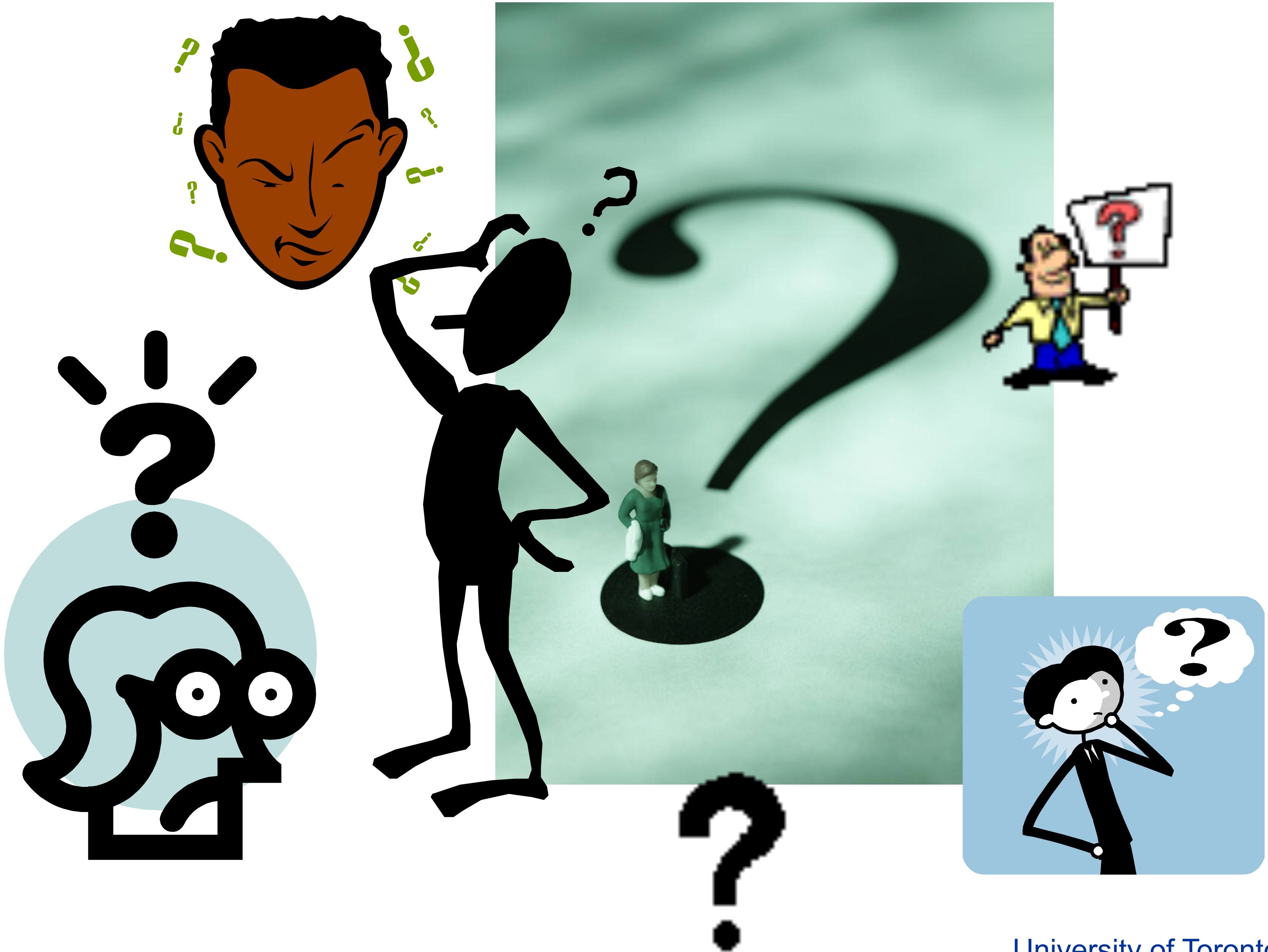
10010011101011

Guiding Solution

10111010101011

i-STS Results

- Significantly cleaner and simpler than i-TSAB
  - Test-bed for investigations about why i-TSAB really works
- Near state of the art
  - Equivalent performance to i-TSAB per iteration
  - But about 5 times slower



Solution-Guided Search

- Metaheuristics use "elite" solutions – why not tree search?
  - Keep around a small set of the "elite" solutions
  - Guide tree search with one of the elite solutions

[B. 2007]
Solution-guided multi-point constructive search for job shop scheduling.
Journal of Artificial Intelligence Research, **29,** 49–77, 2007.

SGS Algorithm

```
initialize elite set, e
while not out of time
   r := random element of e
   s := search(r)
   if s is better than r
       replace r by s
return best(e)
```

Assumes you can quickly find "solutions"

1) Limited search (e.g., by time, fails, etc)
2) Should use random-ized heuristic

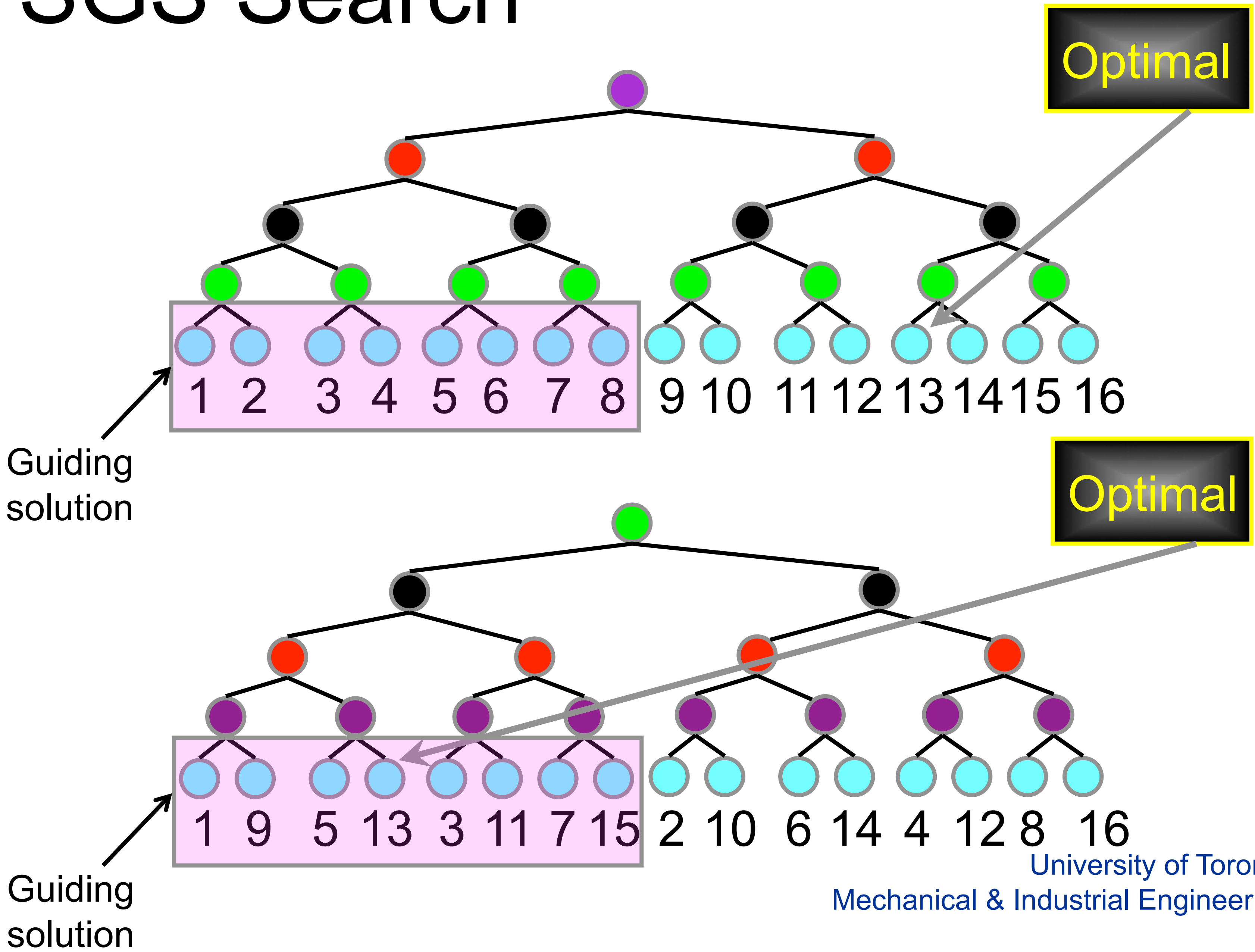
Guiding Search with a Solution

- Given a solution:

  $s = \{(v_1 = x_1), (v_2 = x_2), \ldots, (v_m = x_m)\}, m \leq n$

```
V := varHeuristic.getVariable()
if (V = x) є s AND if x є dom(V)
  branch ((V = x) OR (V ≠ x))
else
  w := valHeuristic.getValue(V)
  branch ((V = w) OR (V ≠ w))
```

SGS Search


Optimal
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Guiding solution
Optimal
1 9 5 13 3 11 7 15 2 10 6 14 4 12 8 16
Guiding solution

Taillard's 20x20 JSPs - Makespan

Instance	UB	CP - chron	CP - restart mean (best)	SGS mean (best)
TA21	1644	1809	1694 (1686)	1666 (1649)
TA22	1600	1689	1654 (1649)	1632 (1621)
TA23	1557	1657	1614 (1602)	1571 (1561)
TA24	1646	1810	1698 (1694)	1664 (1652)
TA25	1595	1685	1673 (1664)	1620 (1608)
TA26	1645	1827	1707 (1701)	1669 (1656)
TA27	1680	1827	1755 (1750)	1716 (1706)
TA28	1603	1778	1664 (1656)	1628 (1619)
TA29	1625	1718	1666 (1660)	1642 (1626)
TA20	1584	1666	1647 (1641)	1607 (1598)

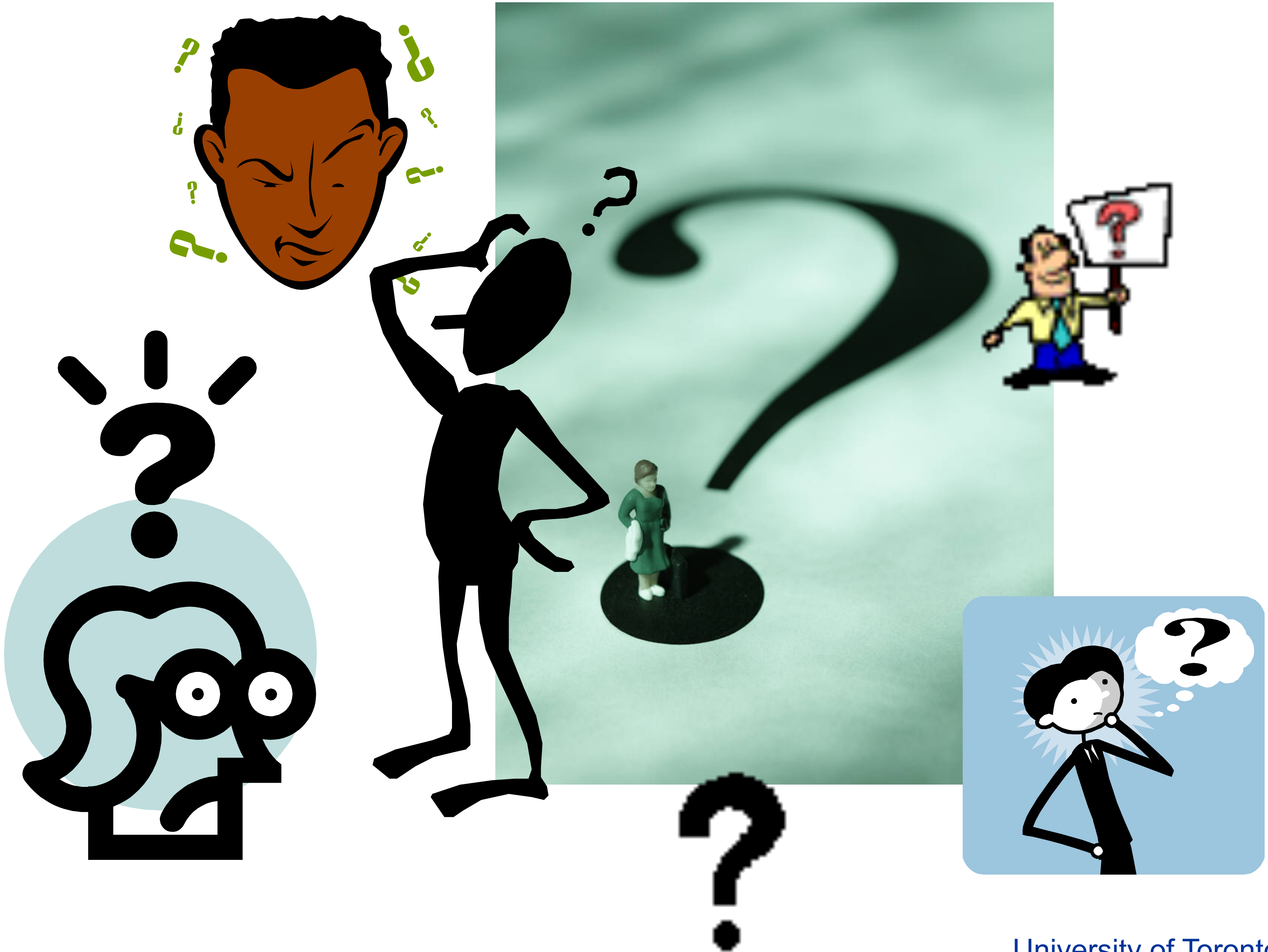
Taillard's 20x20 JSPs - Makespan

Instance	UB	CP - chron	CP - restart mean (best)	SGS mean (best)	i-STS mean (best)
TA21	1644	1809	1694 (1686)	1666 (1649)	1648 (1647)
TA22	1600	1689	1654 (1649)	1632 (1621)	1614 (1600)
TA23	1557	1657	1614 (1602)	1571 (1561)	1560 (1557)
TA24	1646	1810	1698 (1694)	1664 (1652)	1653 (1647)
TA25	1595	1685	1673 (1664)	1620 (1608)	1599 (1595)
TA26	1645	1827	1707 (1701)	1669 (1656)	1653 (1651)
TA27	1680	1827	1755 (1750)	1716 (1706)	1690 (1687)
TA28	1603	1778	1664 (1656)	1628 (1619)	1617 (1614)
TA29	1625	1718	1666 (1660)	1642 (1626)	1628 (1627)
TA20	1584	1666	1647 (1641)	1607 (1598)	1587 (1584)

Conclusion

SGS significantly improves standard CP approaches

But is not competitive with i-STS



Outline

- State-of-the Art in Job Shop Scheduling
- Iterated Simple Tabu Search (i-STS) & Solution-Guided Search (SGS)
- Hybrid i-STS/SGS

Why Hybridize?

- Propagation algorithms work better in a more constrained state
  - CP can't find good solutions, but given a good solution can it find a better one?
- We hypothesize that SGS strongly intensifies around a solution
  - better than tabu search at intensification?
  - does this bring us anything?

The Simplest Hybrid We Could Think Of

- Given T seconds
- Run i-STS for T/2
- Use final elite set from i-STS as initial elite set for SGS
- Run SGS for T/2

Results – Taillard's 20x20

Instance	UB	SGS mean (best)	i-STS mean (best)
TA21	1644	1666 (1649)	1648 (1647)
TA22	1600	1632 (1621)	1614 (1600)
TA23	1557	1571 (1561)	1560 (1557)
TA24	1646	1664 (1652)	1653 (1647)
TA25	1595	1620 (1608)	1599 (1595)
TA26	1645	1669 (1656)	1653 (1651)
TA27	1680	1716 (1706)	1690 (1687)
TA28	1603	1628 (1619)	1617 (1614)
TA29	1625	1642 (1626)	1628 (1627)
TA20	1584	1607 (1598)	1587 (1584)

Results – Taillard's 20x20

Instance	UB	SGS mean (best)	i-STS mean (best)	Hybrid mean (best)
TA21	1644	1666 (1649)	1648 (1647)	1644 (1642)
TA22	1600	1632 (1621)	1614 (1600)	1613 (1610)
TA23	1557	1571 (1561)	1560 (1557)	1559 (1557)
TA24	1646	1664 (1652)	1653 (1647)	1648 (1645)
TA25	1595	1620 (1608)	1599 (1595)	1601 (1595)
TA26	1645	1669 (1656)	1653 (1651)	1649 (1647)
TA27	1680	1716 (1706)	1690 (1687)	1684 (1680)
TA28	1603	1628 (1619)	1617 (1614)	1616 (1613)
TA29	1625	1642 (1626)	1628 (1627)	1626 (1625)
TA30	1584	1607 (1598)	1587 (1584)	1589 (1584)

Results – 4 Taillard Sets

Instance Set	UB	i-TSAB	Zhang		Hybrid		
			best	mean	best	mean	worst
TA11-20	2.29	2.81	2.37	2.92	2.26	2.42	2.69
TA21-30	5.38	5.68	5.44	5.97	5.50	5.70	5.89
TA31-40	0.46	0.78	0.55	0.93	0.49	0.72	0.98
TA41-50	4.02	4.70	4.07	4.84	4.17	4.70	5.28
Overall	3.04	3.49	3.11	3.67	3.11	3.38	3.71

Statistic

Mean relative error to best-known lower bound

Results – 4 Taillard Sets

Instance Set	UB	i-TSAB	Zhang		Hybrid		
			best	mean	best	mean	worst
TA11-20	2.29	2.81	2.37	2.92	2.26	2.42	2.69
TA21-30	5.38	5.68	5.44	5.97	5.50	5.70	5.89
TA31-40	0.46	0.78	0.55	0.93	0.49	0.72	0.98
TA41-50	4.02	4.70	4.07	4.84	4.17	4.70	5.28
Overall	3.04	3.49	3.11	3.67	3.11	3.38	3.71

Statistic

Mean relative error to best-known lower bound

Results – 4 Taillard Sets

Instance Set	UB	i-TSAB	Zhang		Hybrid		
			best	mean	best	mean	worst
TA11-20	2.29	2.81	2.37	2.92	2.26	2.42	2.69
TA21-30	5.38	5.68	5.44	5.97	5.50	5.70	5.89
TA31-40	0.46	0.78	0.55	0.93	0.49	0.72	0.98
TA41-50	4.02	4.70	4.07	4.84	4.17	4.70	5.28
Overall	3.04	3.49	3.11	3.67	3.11	3.38	3.71

Statistic

Mean relative error to best-known lower bound

Overall Results

- Able to find and prove optimality for 6 instances
- 10 new best solutions found out of 40 problem instances
  - – across different parameterizations

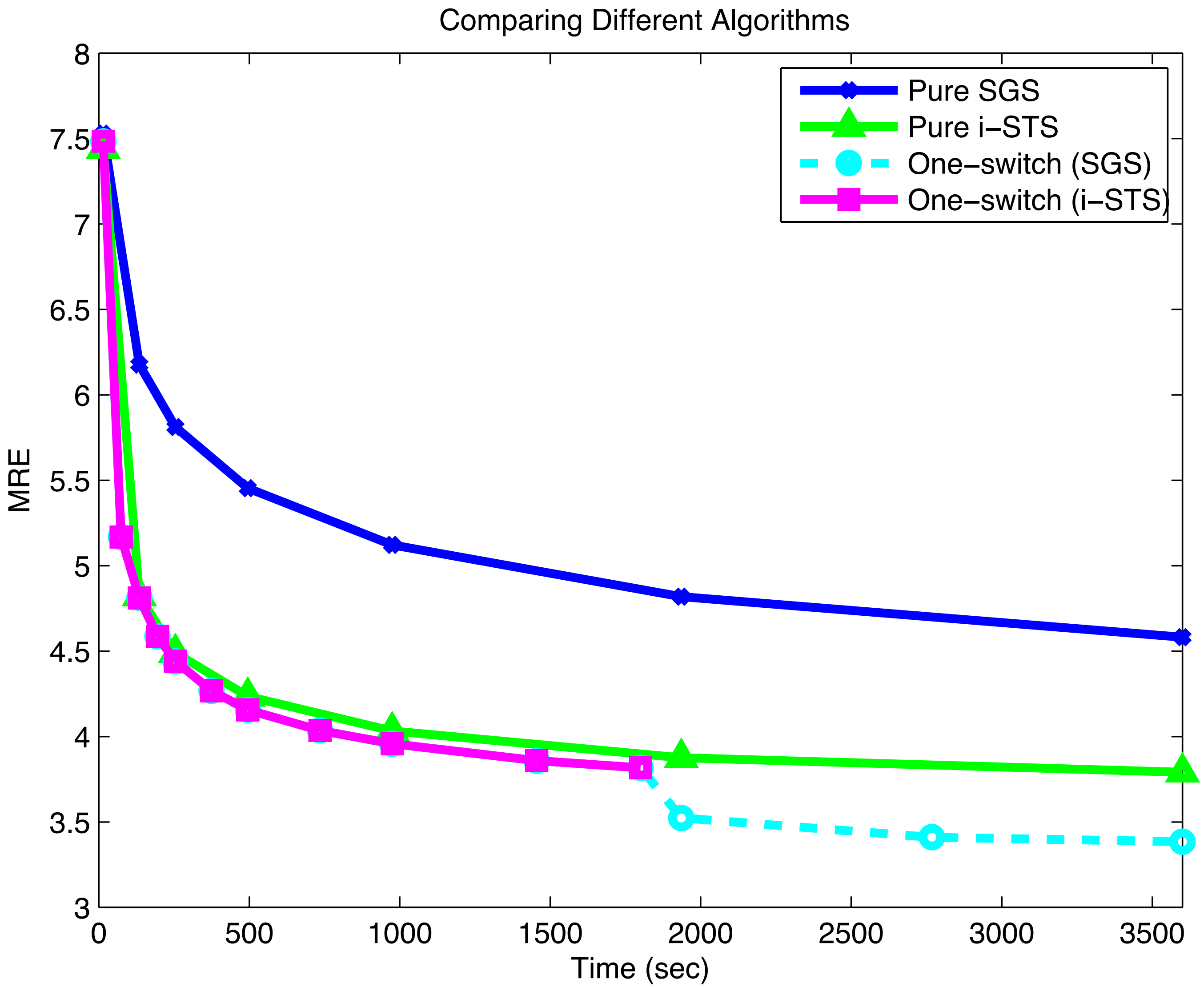
What About More Sophistication?

- Switch back-and-forth, communicating the elite set
- Longer intervals later in the run
- Reinforcement learning to give more time to the better performer

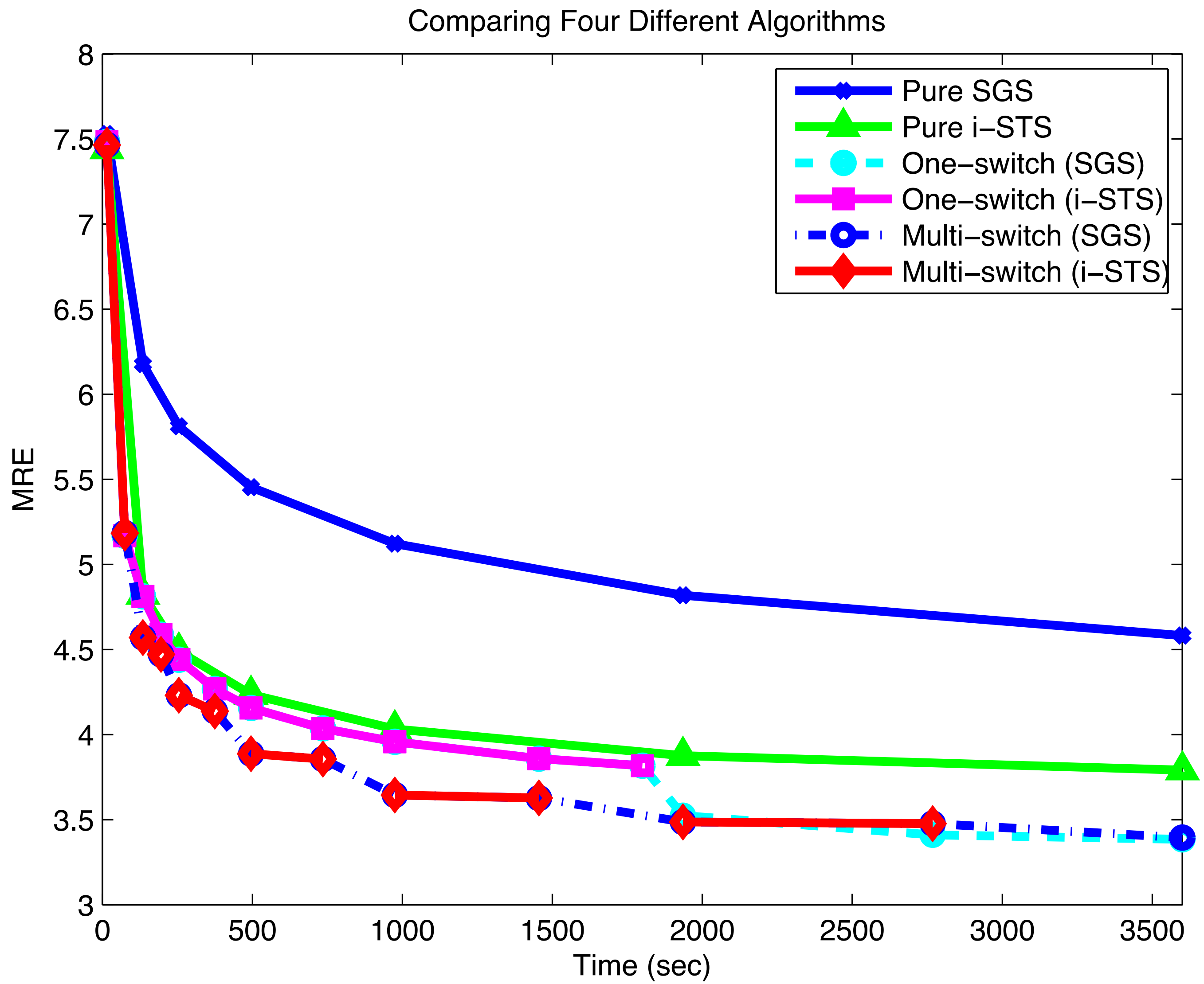
Nothing significantly improved over simple hybrid

[Carchrae & B. 2005]
Applying Machine Learning to low-knowledge control of optimization algorithms.
Computational Intelligence, **21**(4) 372-387, 2005.

Results – 4 Taillard Sets



Results – 4 Taillard Sets



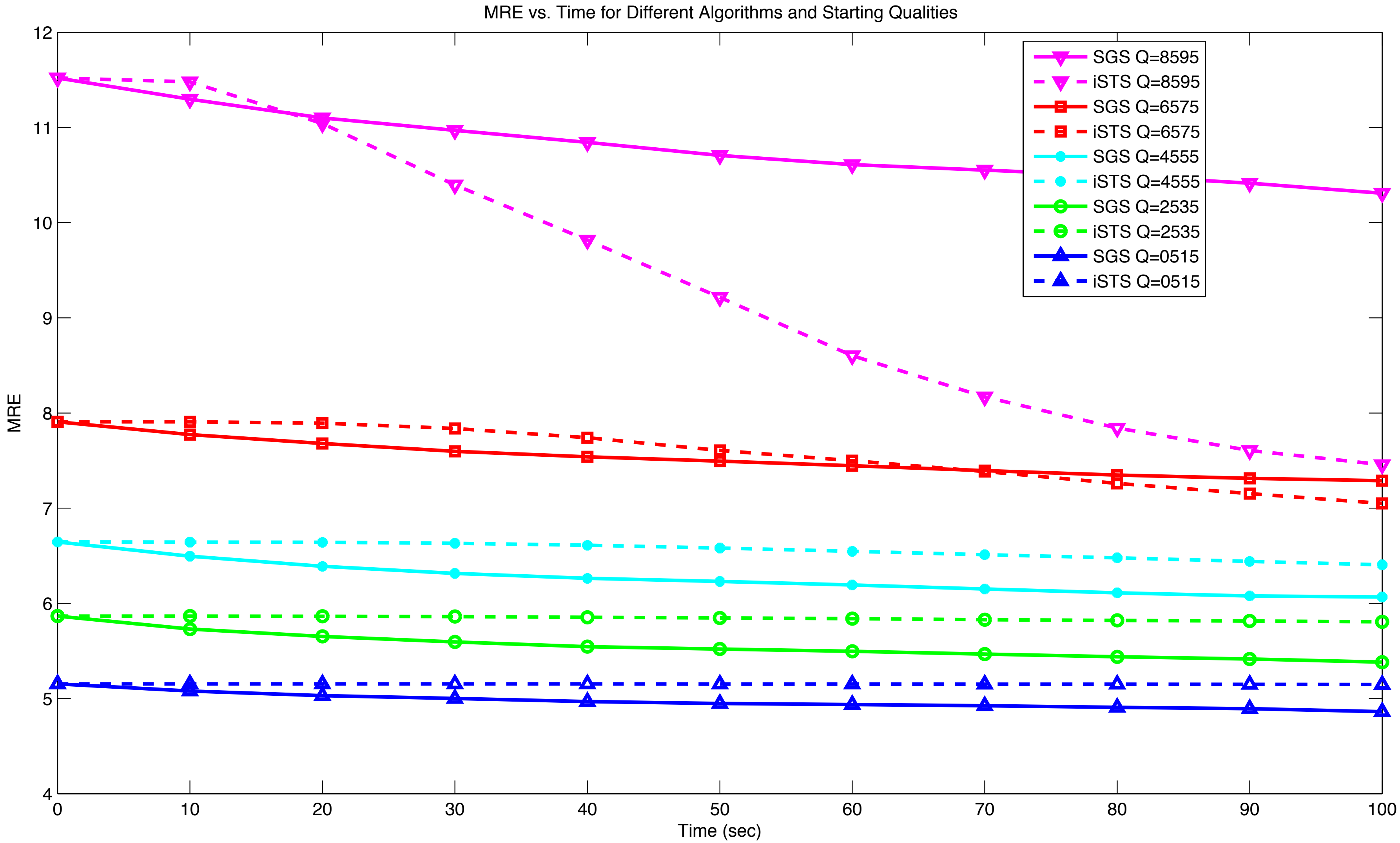
What is Going On?

- Not completely sure
- Both i-STS and SGS are doing a form of large-neighborhood search around good solutions
  - i-STS much more biased by cost gradient but gets further away from seed solution faster

Experiment (ta41-ta50)

- Gather all (feasible) solutions from all runs and bucket them by quality
  - 5-15 %tile, 25-35 %tile, etc.
- Randomly draw an elite pool from each bucket
- Run pure i-STS and pure SGS

Improving Elite Pools

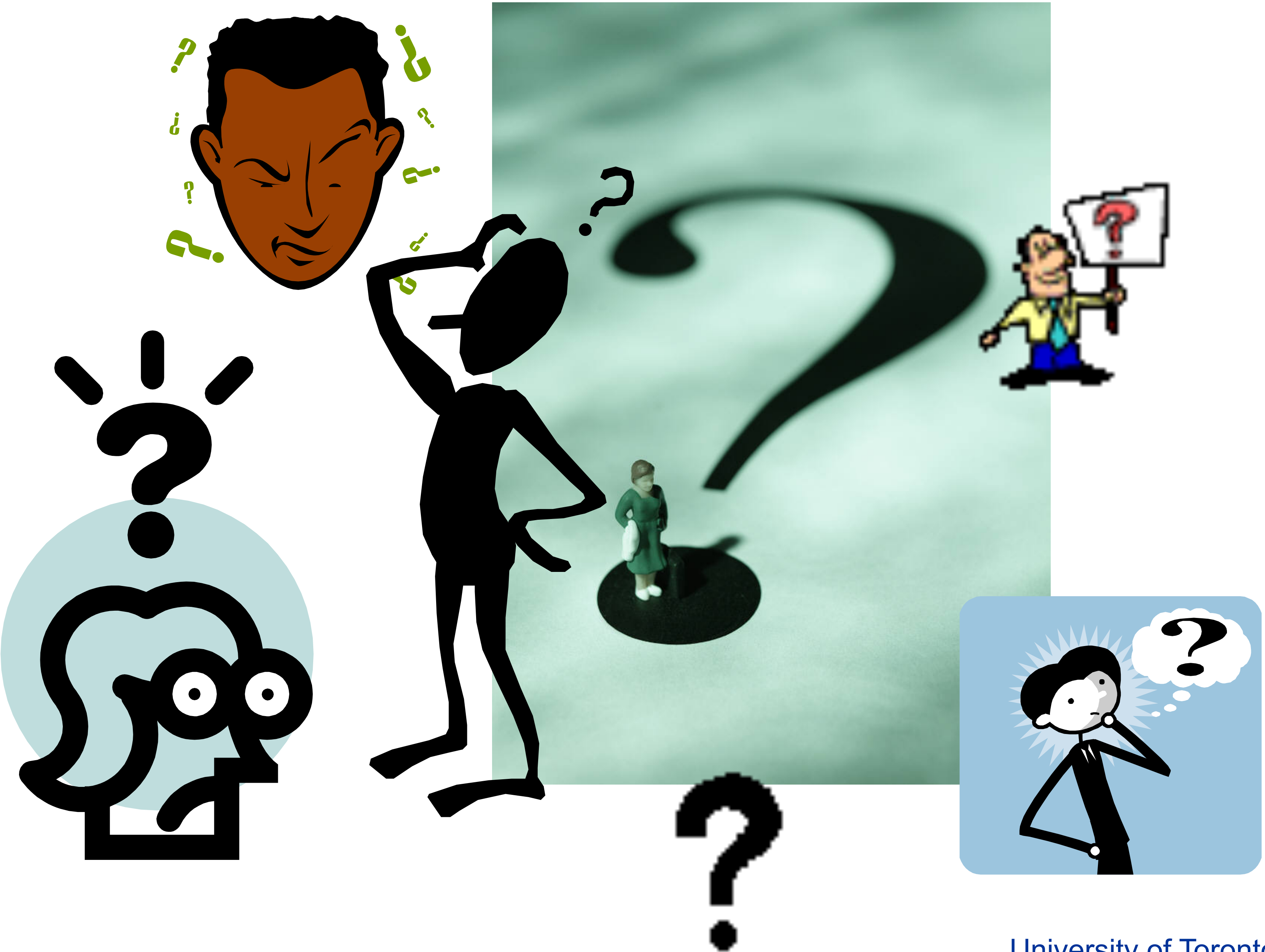


Questions

- Why does this simple hybrid work?
  - Is SGS just doing independent intensification around each elite solution?
    - Grabbing the low-hanging fruit that i-STS misses?
  - How specific is this to JSP search space topology?
- Example of a larger hybrid pattern?
  - Heuristic search then optimize [F. Soumis]

Conclusion

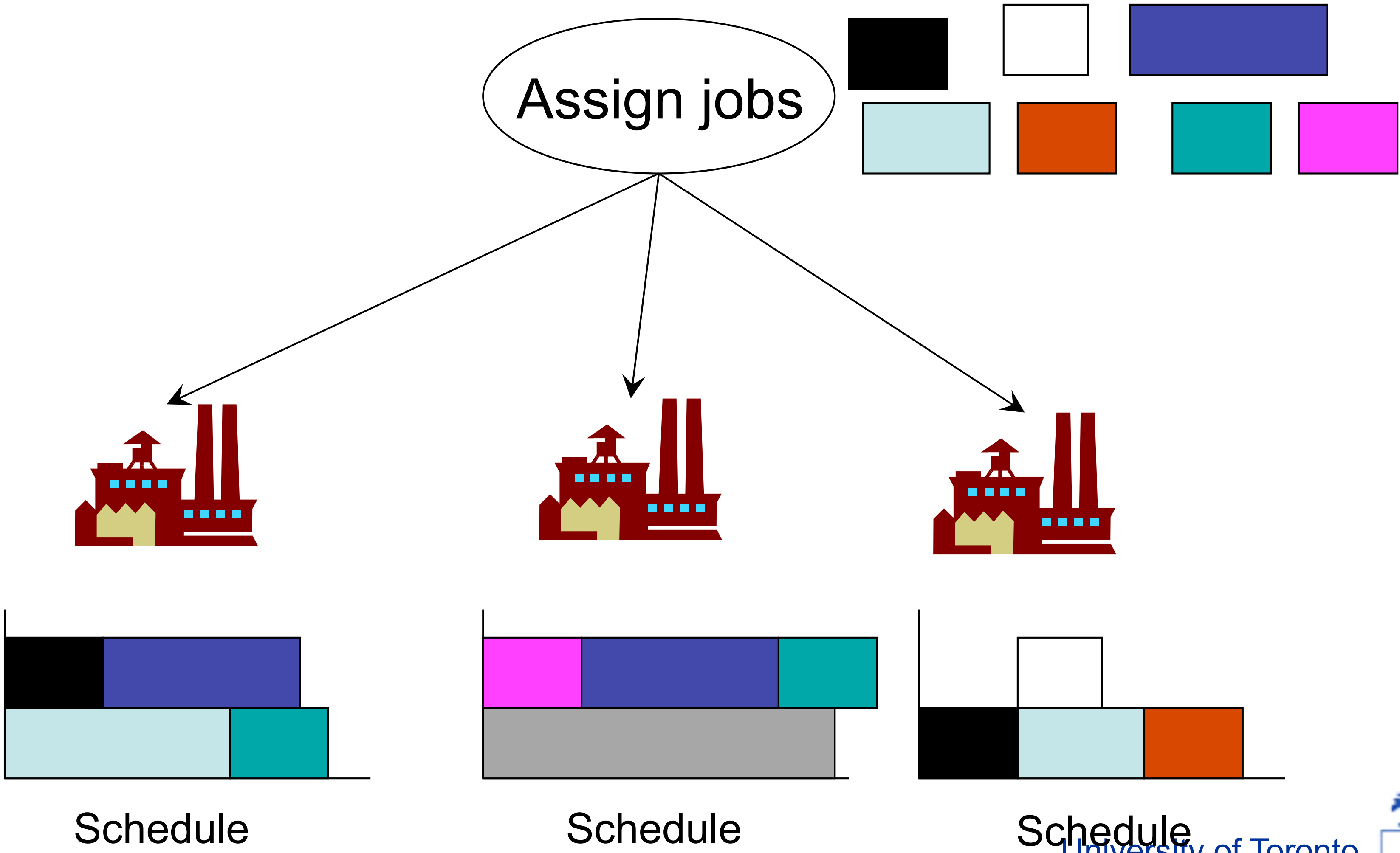
- i-STS/SGS is a state-of-the-art hybrid of tabu search and constraint programming for job-shop scheduling
- Consistently yields very high quality solutions



Outline: Part 2

- Remembering Yesterday
  - 90 minutes in 3 slides
- Combining CP and Tabu Search for Job Shop Scheduling
- Combining MIP and CP for Resource Allocation/Scheduling Problems

Planning & Scheduling



[Hooker 2005]
A Hybrid Method for Planning and Scheduling. *Constraints*, **10**, 385-401, 2005.

CP Model

$$\min \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{J}} c_{jk} x_{jk}$$

Minimize resource assignment cost

$$\text{s.t.} \sum_{k\in\mathcal{K}} x_{jk} = 1$$

Each activity is assigned to one resource

$$\texttt{optcumulative}(\boldsymbol{S}_{\cdot\boldsymbol{k}}, \boldsymbol{x}_{\cdot\boldsymbol{k}}, \boldsymbol{p}_{\cdot\boldsymbol{k}}, \boldsymbol{r}_{\cdot\boldsymbol{k}}, C_k)$$

Resource capacity constraint

$$\mathcal{R}_j \leq S_j \leq \mathcal{D}_j - p_{jk}$$

Time-window constraints

$$x_{jk} \in \{0,1\} \qquad \forall j \in \mathcal{J}, \forall k \in \mathcal{K}$$

$$S_{jk} \in \mathbb{Z} \qquad \forall j \in \mathcal{J}, \forall k \in \mathcal{K}.$$

The `optcumulative` Global Constraint

- Generalizes `disjunctive` to enforce resource capacity including:
  - non-unary capacity (unary = one)
  - non-unary requirements
  - optional activities
- A number of the `disjunctive` inference algorithms have been extended

MIP Model

$$\min \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{J}} \sum_{t=\mathcal{R}_j}^{\mathcal{D}_j - p_{jk}} c_{jk}\, x_{jkt}$$

Minimize resource assignment cost

$$\text{s.t.} \sum_{k\in\mathcal{K}} \sum_{t=\mathcal{R}_j}^{\mathcal{D}_j - p_{jk}} x_{kjt} = 1$$

Each activity starts once on one resource

$$\sum_{j\in\mathcal{J}} \sum_{t'\in T_{jkt}} r_{jk}\, x_{jkt'} \leq C_k$$

Resource capacity constraint

$$x_{jkt} \in \{0,1\} \qquad \forall k \in \mathcal{K},\ \forall j \in \mathcal{J},\ \forall t,$$

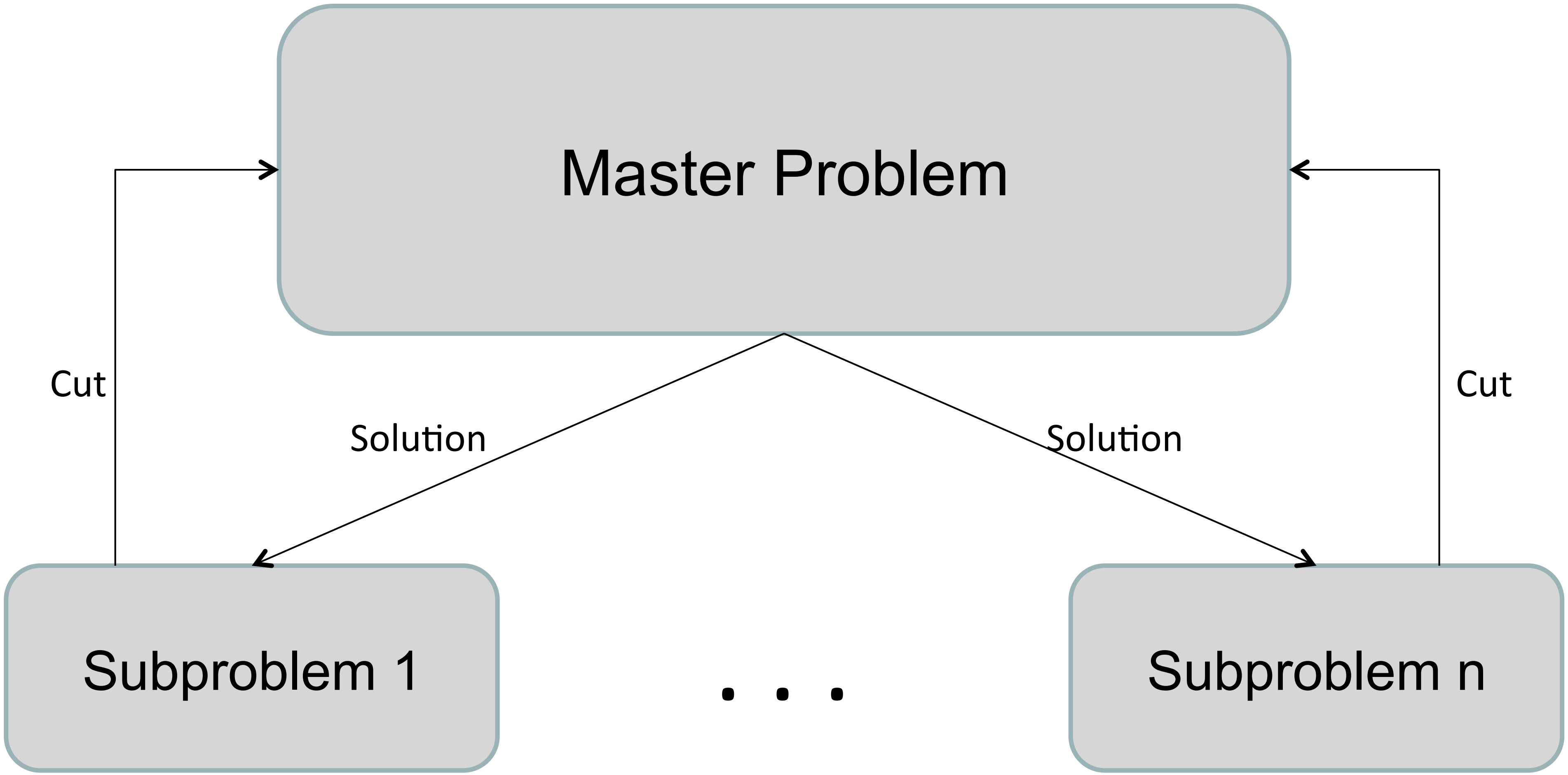
with $T_{jkt} = \{t - p_{jk}, \ldots, t\}$.

Logic-Based Benders Decomposition (LBBD)

Monolithic Model
(e.g., MIP, CP, …)

[Hooker 2005]

LBBD



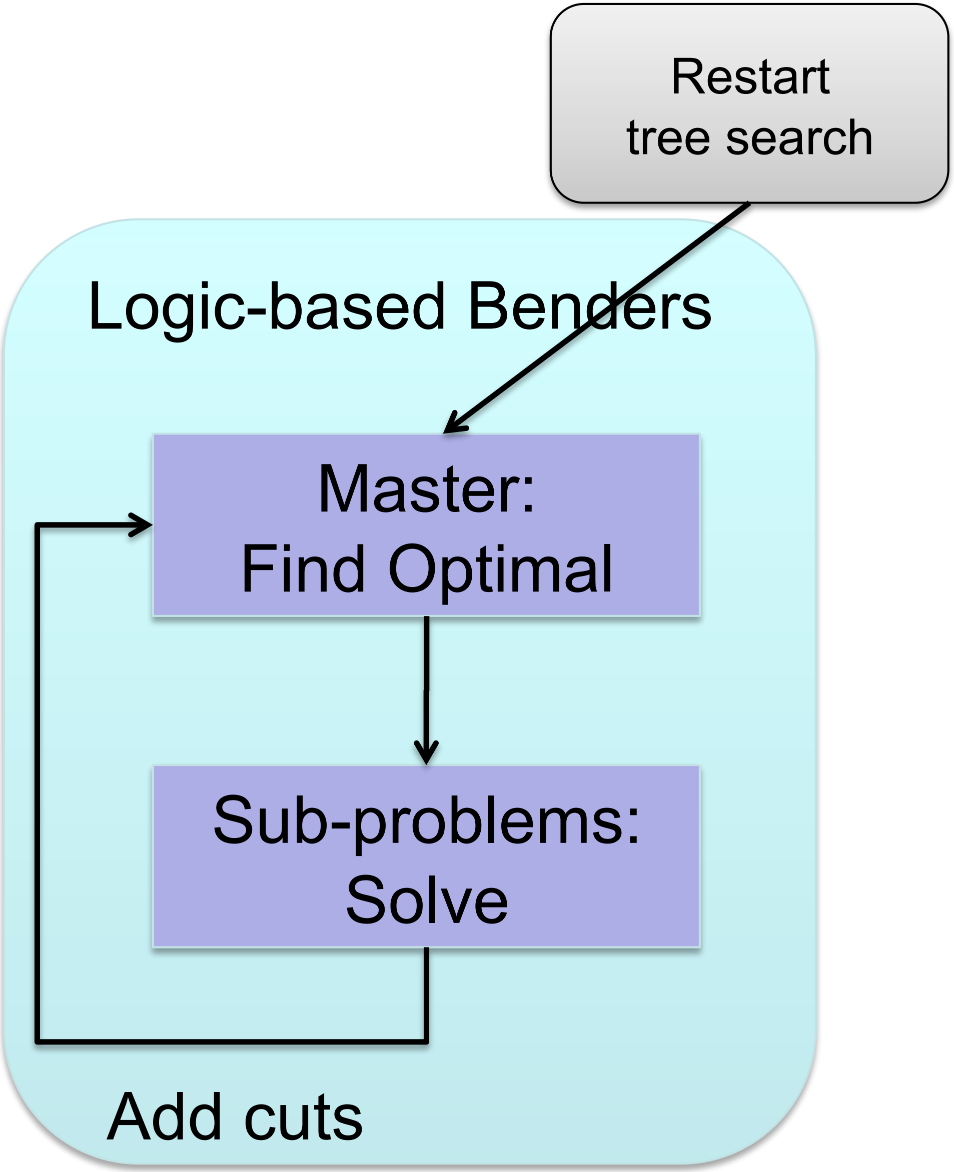
[Hooker 2005]

Logic-Based Benders

- Partition problem into
  - Master problem with decision variables, *y*
  - Sub-problem(s) with decision variables, *x*
- When the *y*'s are fixed (to say, $\hat{y}$), sub-problems are formed
- Each sub-problem is an inference dual
  - What is the max. LB that can be inferred assuming $y = \hat{y}$?

[Hooker 2000]

LBBD


Restart
tree search
Logic-based Benders
Master:
Find Optimal
Sub-problems:
Solve
Add cuts

LBBD Master (MIP)

$$\min \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{J}} c_{jk} x_{jk}$$

Minimize resource assignment cost

$$\text{s.t.} \quad \sum_{k\in\mathcal{K}} x_{jk} = 1$$

Each activity is assigned to one resource

$$\sum_{j\in\mathcal{J}} x_{jk} p_{jk} r_{jk} \leq \hat{C}_k \quad \forall k$$

Sub-problem relaxation

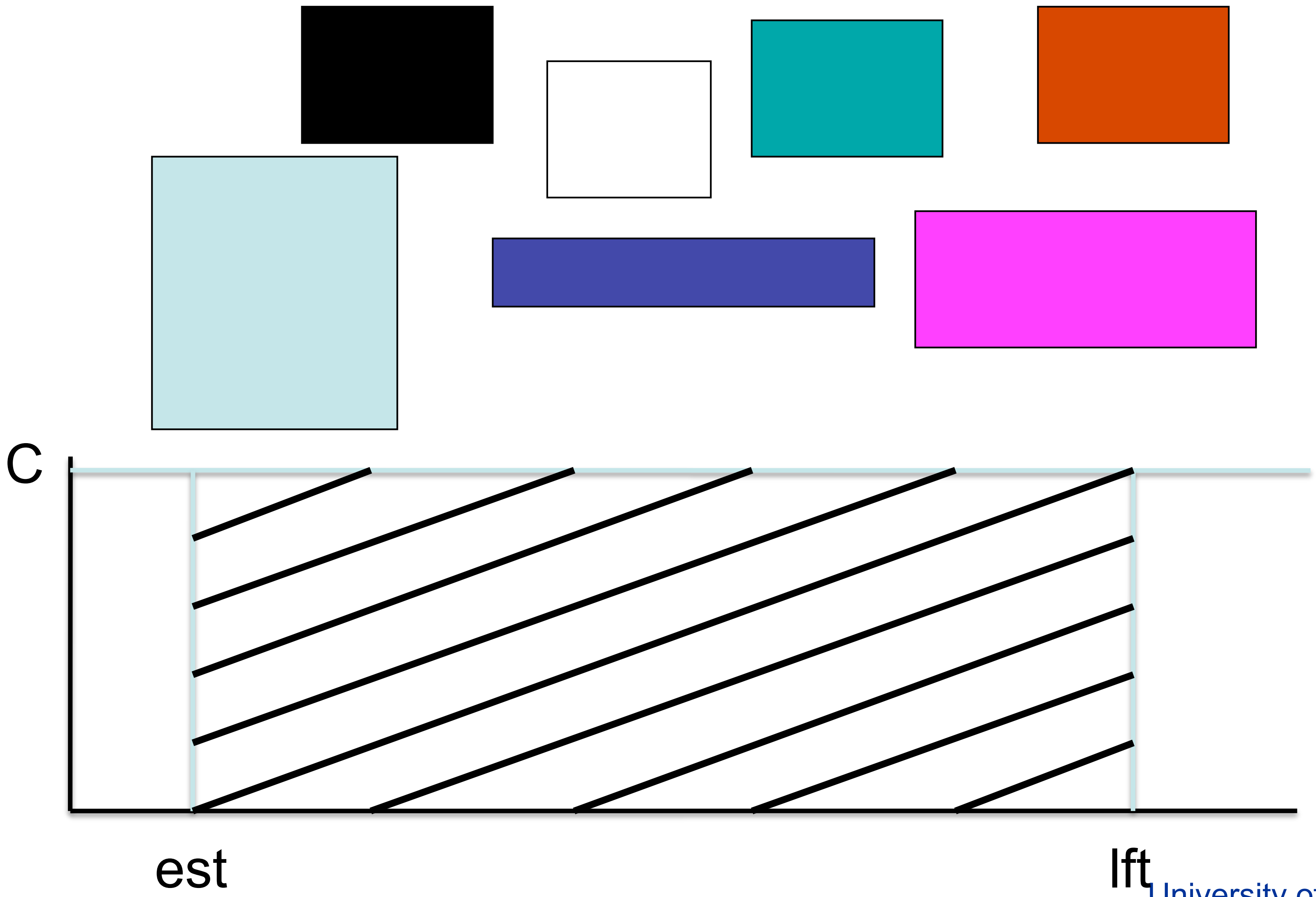
$$\sum_{j\in\mathcal{J}_{hk}} (1 - x_{jk}) \geq 1 \quad \forall k \in \mathcal{K},\ h \in [H-1]$$

Benders cut

$$x_{kj} \in \{0,1\} \quad \forall k \in \mathcal{K},\ \forall j \in \mathcal{J},$$

with $\hat{C}_k = C_k \cdot (\max_{j\in\mathcal{J}}\{\mathcal{D}_j\} - \min_{j\in\mathcal{J}}\{\mathcal{R}_j\})$.

Sub-problem Relaxation



Benders Cut

$$\sum_{j \in \mathcal{J}_{hk}} (1 - x_{jk}) \geq 1 \qquad \forall k \in \mathcal{K},\ h \in [H-1]$$

- Do not allow same assignment of activities (or a superset)

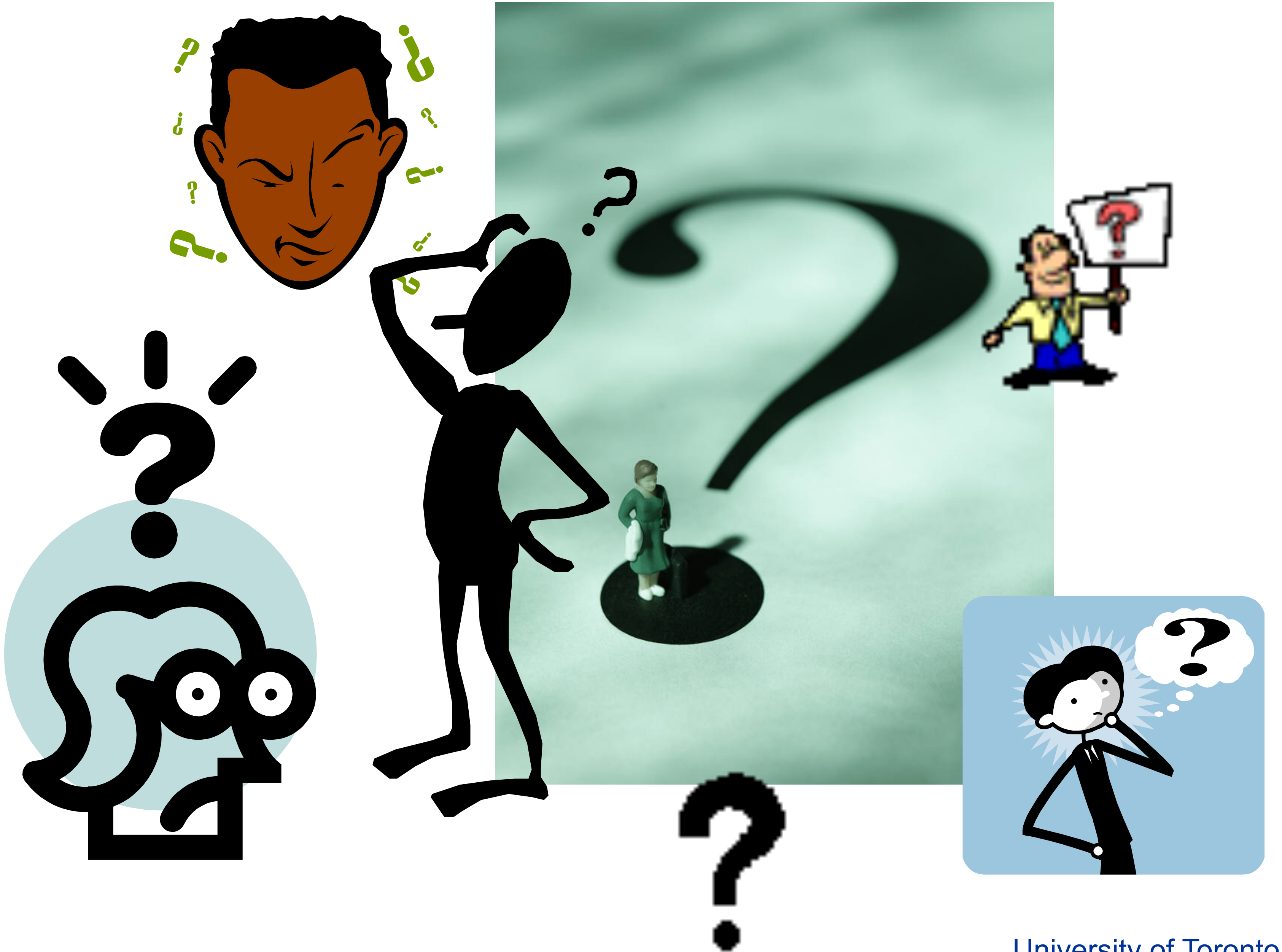
Benders Subproblem (CP)

$$\texttt{cumulative}(\boldsymbol{S}, \boldsymbol{p_{\cdot k}}, \boldsymbol{r_{\cdot k}}, C_k)$$

$$\mathcal{R}_j \leq S_j \leq \mathcal{D}_j - p_{jk} \qquad \forall j \in \mathcal{J}_k$$

$$S_j \in \mathbb{Z} \qquad \forall j \in \mathcal{J}_k$$

- Single-machine, feasibility problem



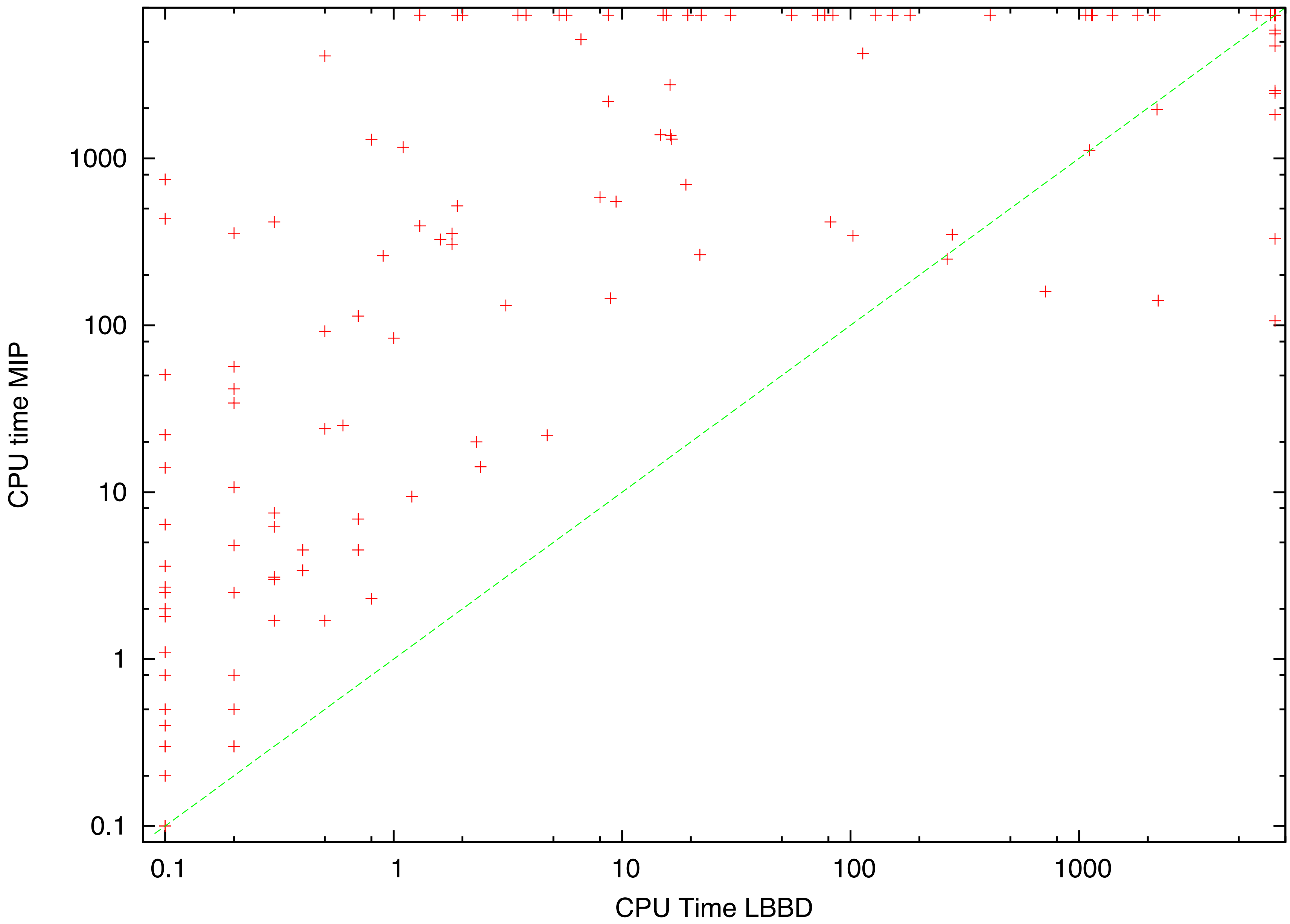
Hooker’s Instances

Model	# Optimal	# Feasible	Run-time (secs) geo-mean
CP	62	69	1311.4
MIP	98	195	778.8
LBBD	119	119	227.3

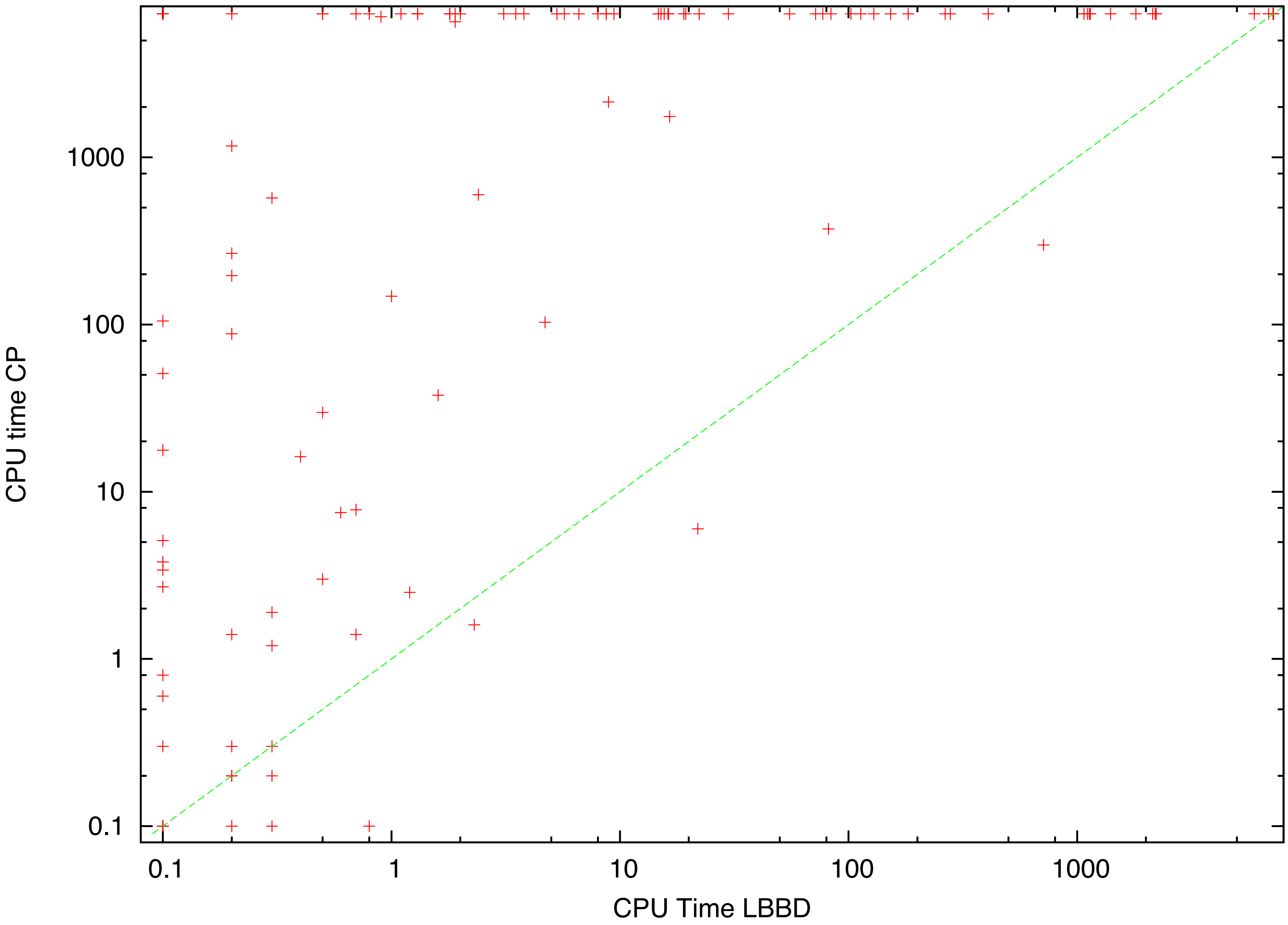
- 195 instances
  - 2 – 4 resources, 10 – 38 jobs

[Heinz & B. 2011]
Solving Resource Allocation/Scheduling Problems with Constraint Integer Programming. *ICAPS 2011 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems*, 2011.

MIP vs LBBD



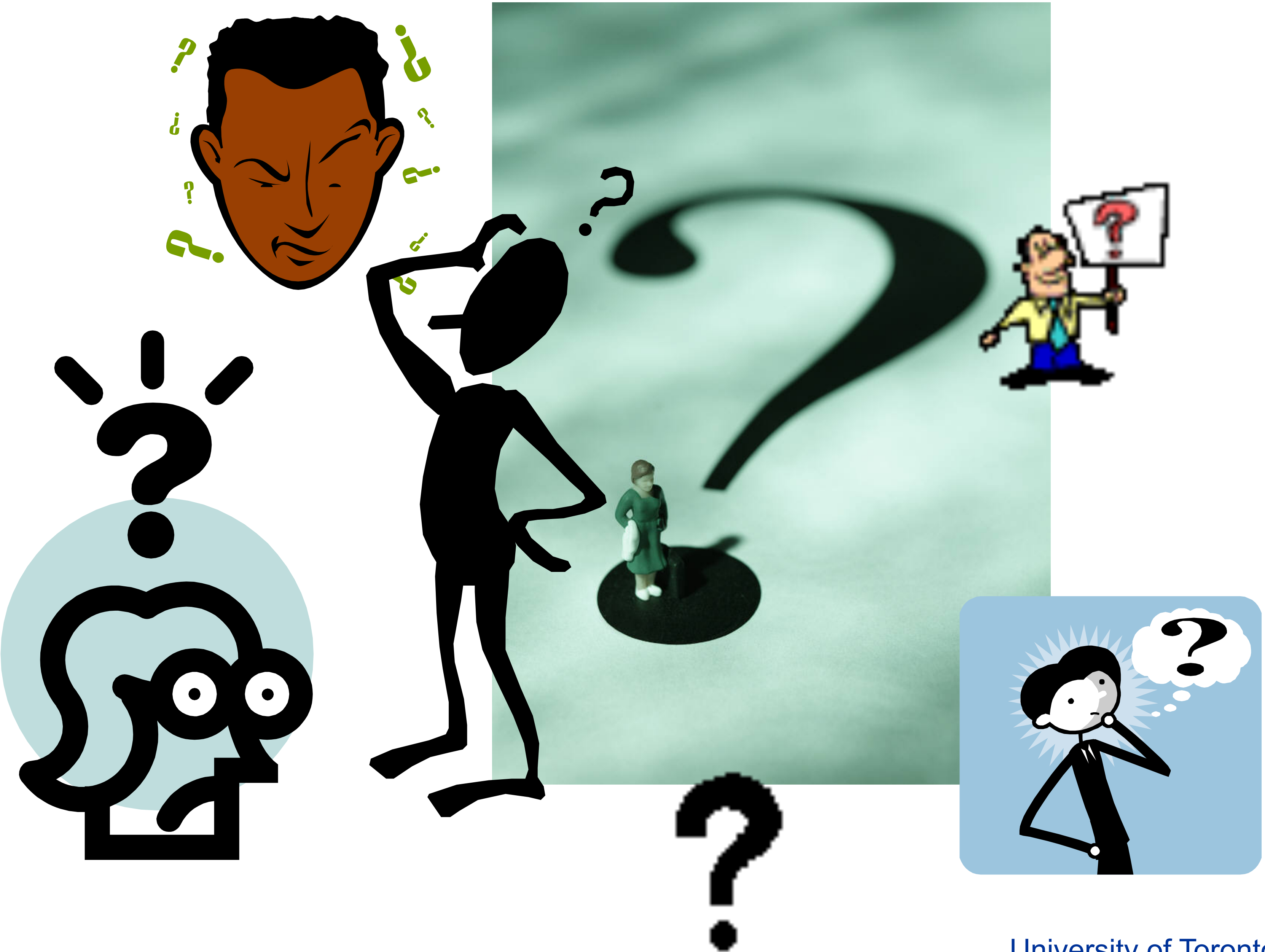
CP vs LBBD



Hooker's Instances

Model	# Optimal	# Feasible	Run-time (secs) geo-mean
CP	62	69	1311.4
MIP	98	195	778.8
LBBD	119	119	227.3

- 195 instances
  - 2 – 4 resources, 10 – 38 jobs



Summary

- Tabu + CP results in state-of-the-art job shop scheduling
  - good solutions guide both Tabu and CP
  - need a deeper understanding of neighborhood search
- MIP + CP in LBBD for state-of-the-art resource allocation/scheduling
  - feasible solutions still a challenge (vs. MIP)
  - generic (but manual) decomposition technique

Themes

- Cost-driven vs. feasibility-driven
  – Cost: Tabu and MIP; Feasibility: CP
- Decomposition vs. whole problem
- Using good solutions for guidance
  – SGS as a form of large-neighborhood search
  – Tabu
- Relaxation (MIP) vs. inference (CP)

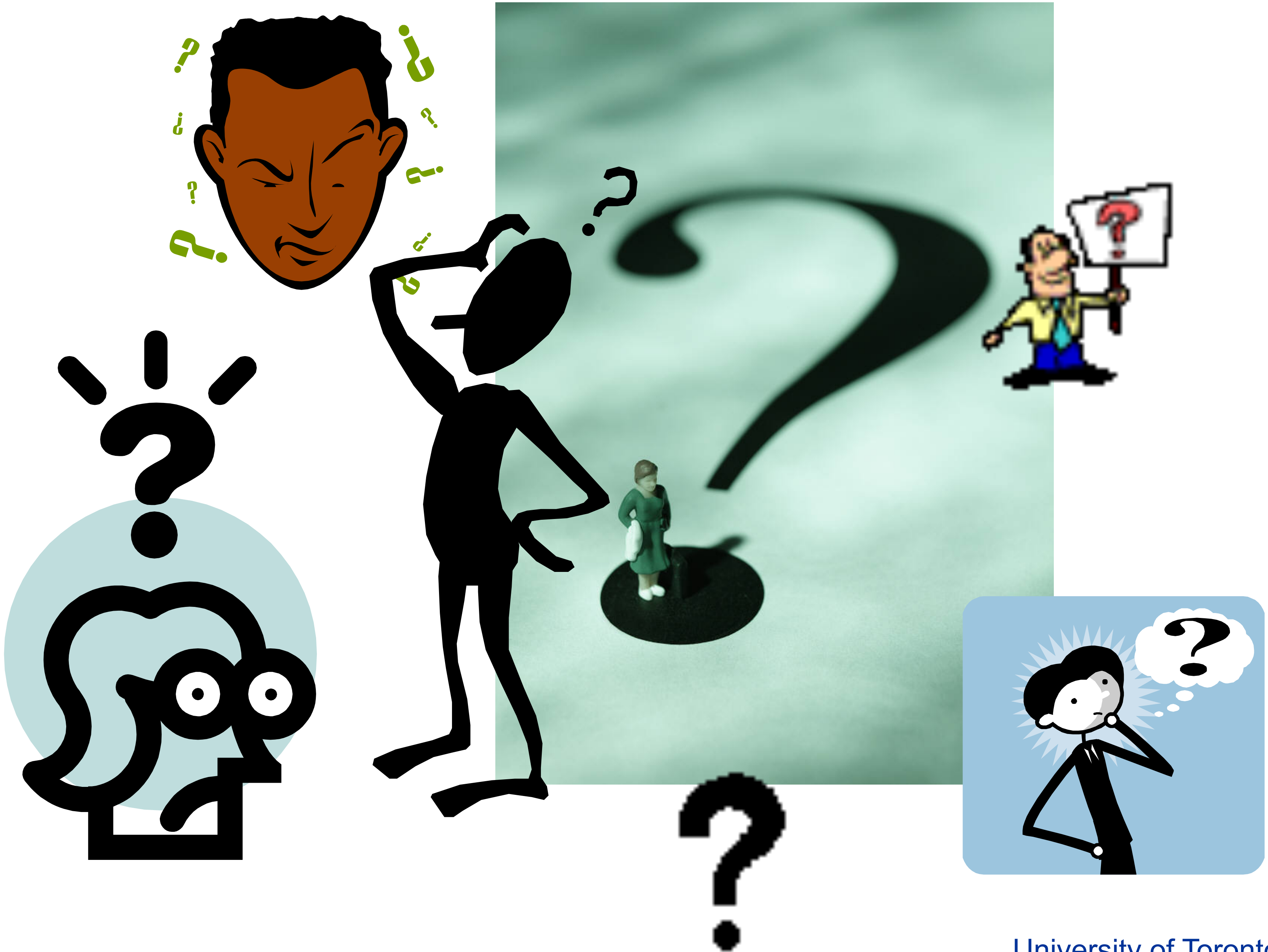
References

[Hooker 2005]
A Hybrid Method for Planning and Scheduling. *Constraints*, **10**, 385-401, 2005.

[Hooker 2007]
Integrated Methods for Optimization, Springer, 2007.

[B. 2010]
Checking-up on Branch-and-Check.
Proceedings of the Sixteenth International Conference of Principles and Practice of Constraint Programming, 84-98, 2010.

[Heinz & B. 2011]
Solving Resource Allocation/Scheduling Problems with Constraint Integer Programming.
ICAPS 2011 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems, 2011.



Is Scheduling Still AI? Part 3: Polemics & Perspectives

J. Christopher Beck
Dept. of Mechanical & Industrial Engineering
University of Toronto
Canada

ACAI Summer School
Freiburg, Germany
June 7 – 10, 2011

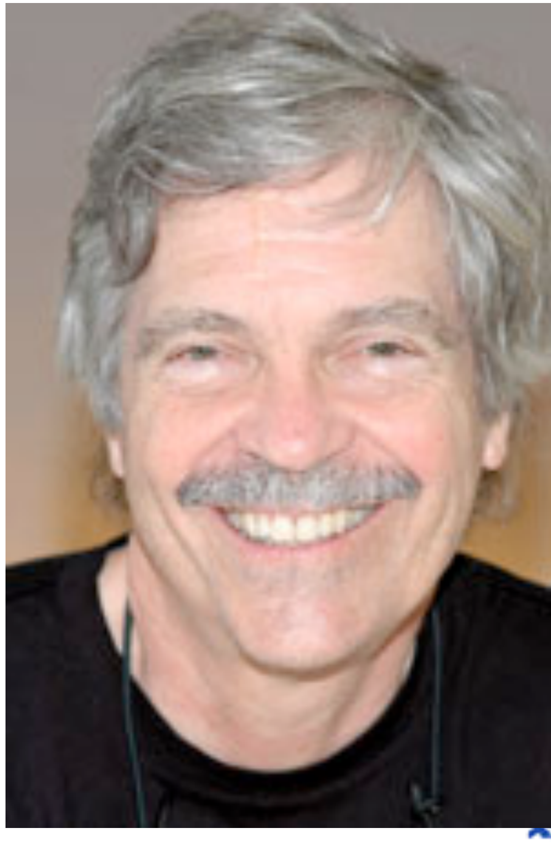
Outline

- Part 1: Core Scheduling Technologies
  – CP, MIP, & Metaheuristics
  – 90 minutes
- Part 2: State of the Art
  – CP + Metaheuristics, CP + MIP
  – 60 minutes
- Part 3: Polemics & Perspectives
  – The Past and the Future?
  – 30 minutes

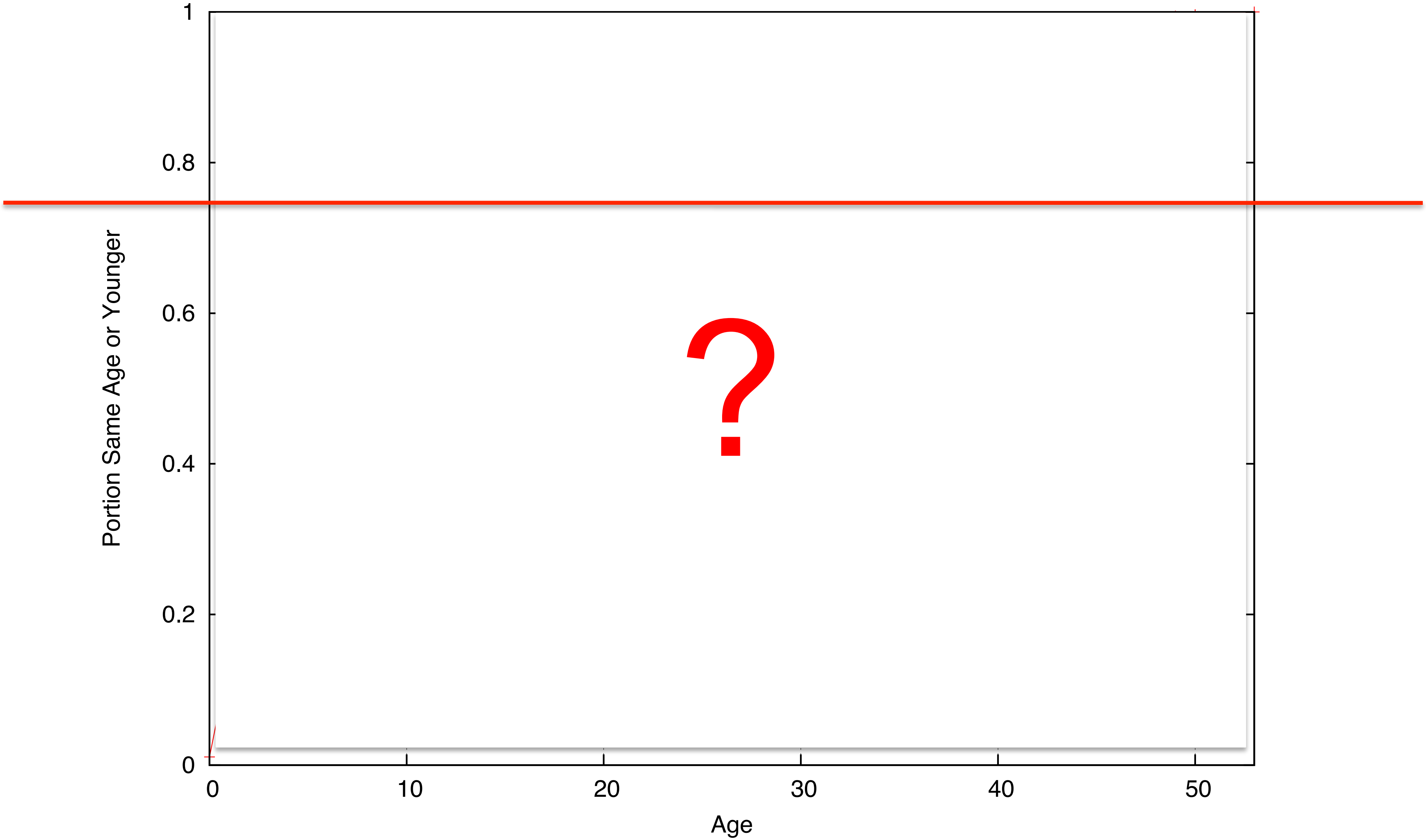
Outline: Part 3

- The Origin of the Species
  - Ancient History (the 70s & 80s)
  - What’s a constraint anyway?
- The 90s
- Scheduling & AI

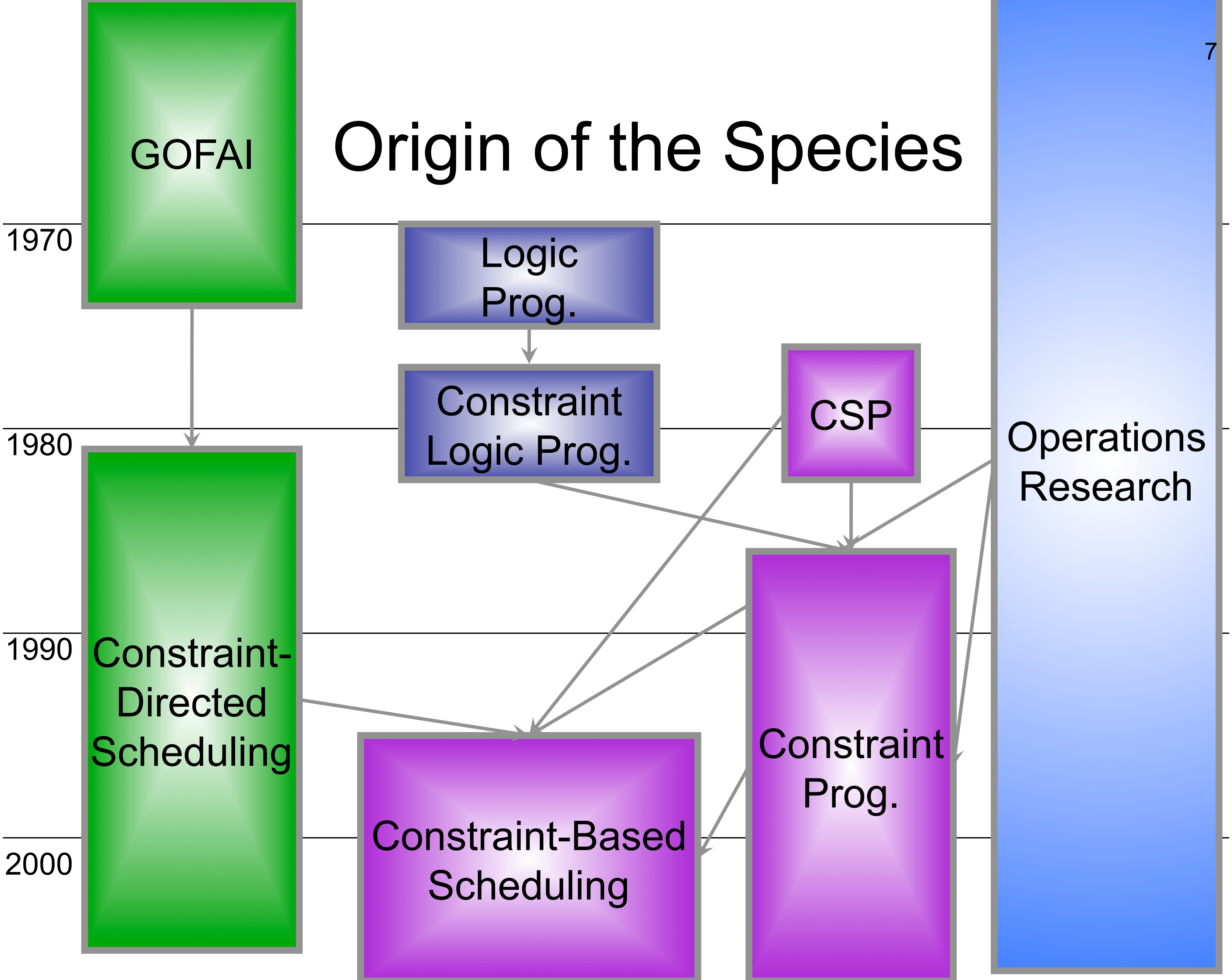
- Computer science is a discipline that ignores its history
  – Alan Kay



History: ICAPS 2010 references



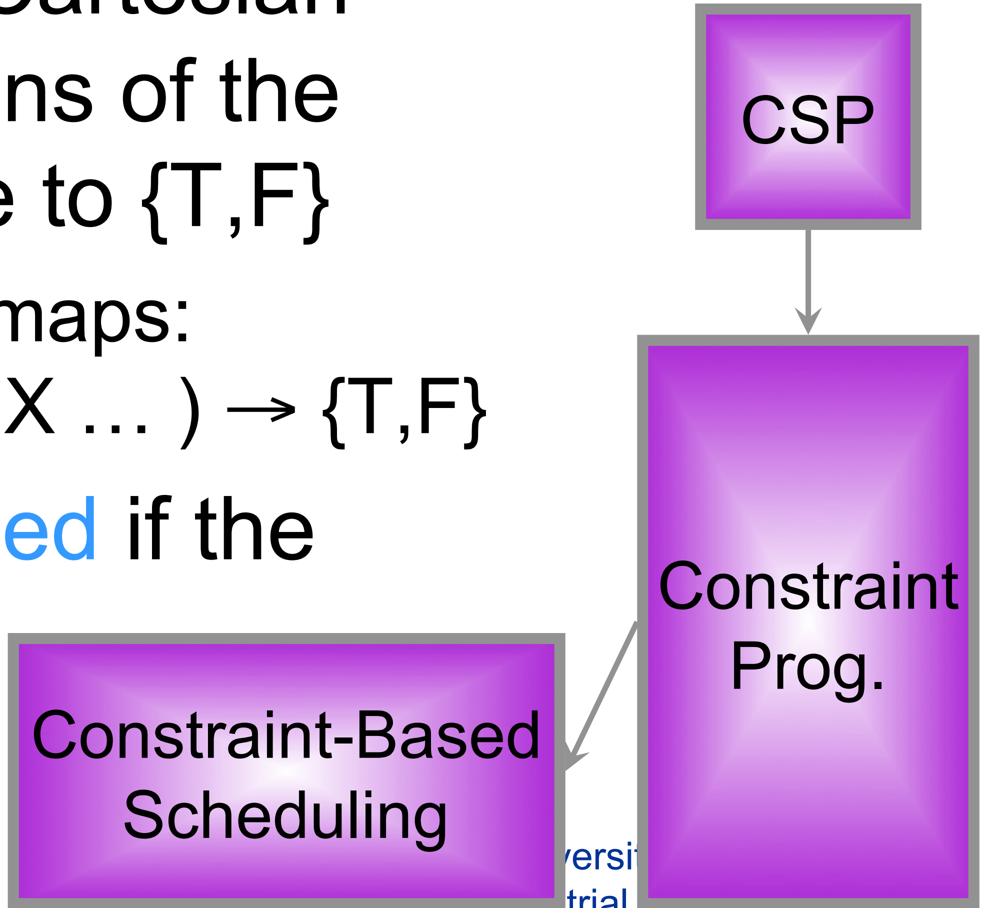
Origin of the Species



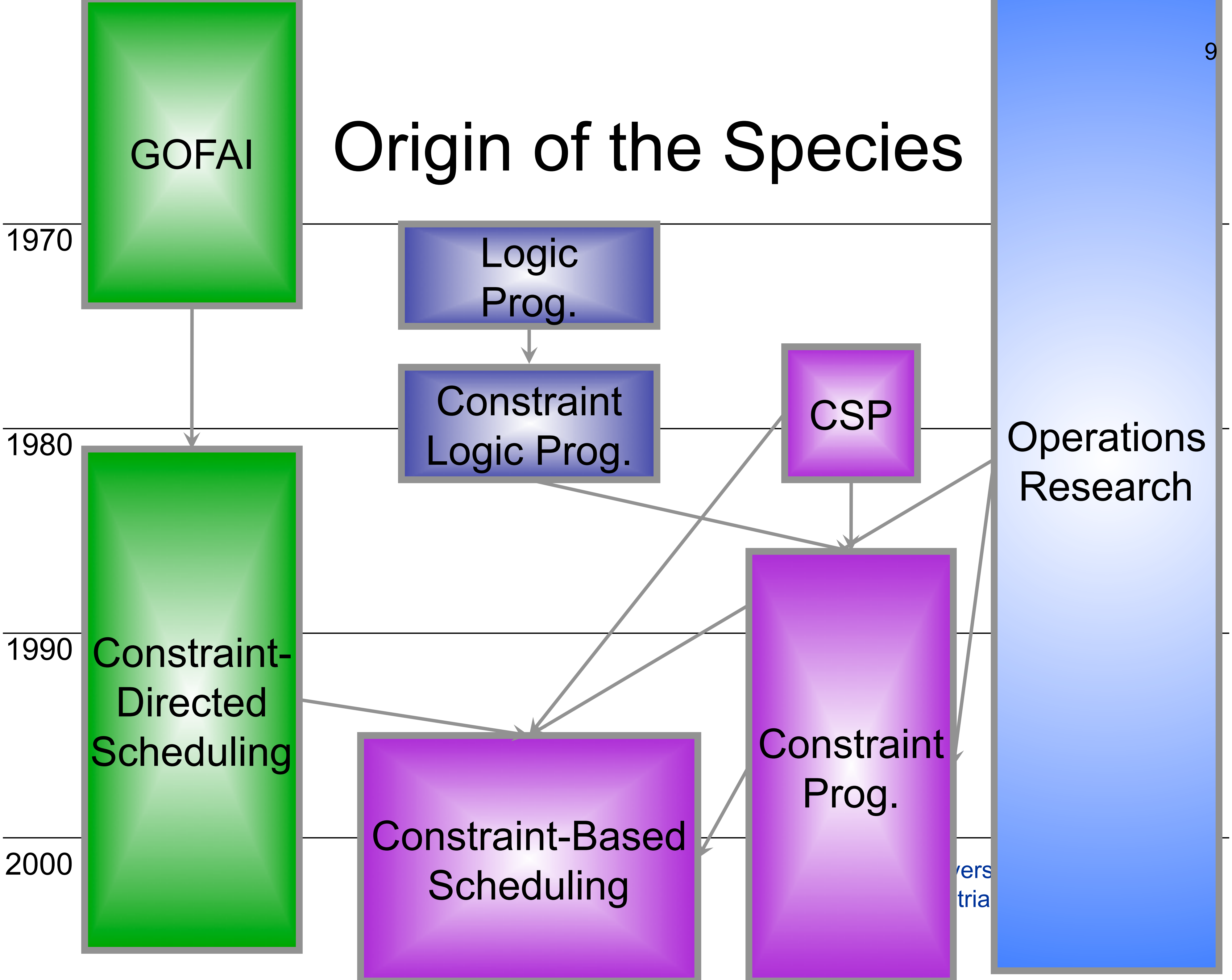
What's a Constraint?

"purple constraint"

- A constraint, c_i, is a mapping from the elements of the Cartesian product of the domains of the variables in its scope to {T,F}
  - $c_i(v_0, v_2, v_4, v_{117}, \ldots)$ maps: $(D_0 \times D_2 \times D_4 \times D_{117} \times \ldots) \rightarrow \{T,F\}$
- A constraint is satisfied if the assignment of the variables in its scope map to T

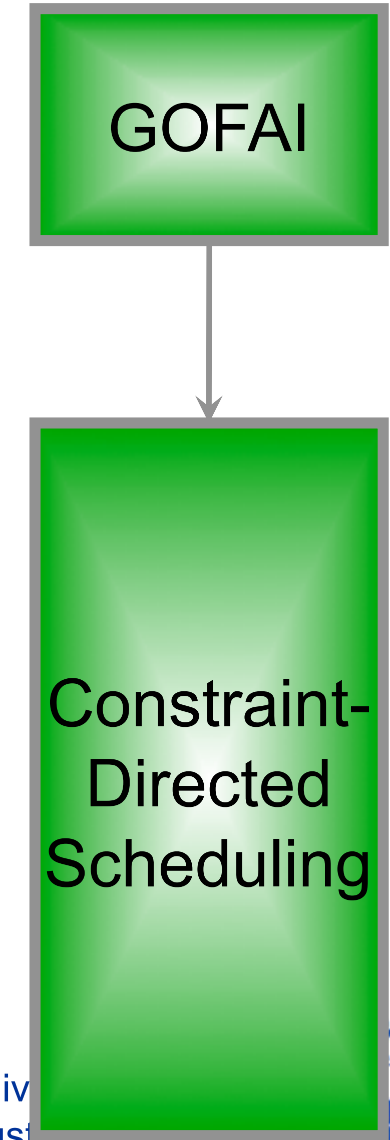


Origin of the Species



What’s a “Green Constraint”?

- A rich, generative object that represents all sorts of knowledge about a problem
  - preferences
  - relevance
  - relaxations
  - descriptions of complex interactions
  - organizational responsibility & authority

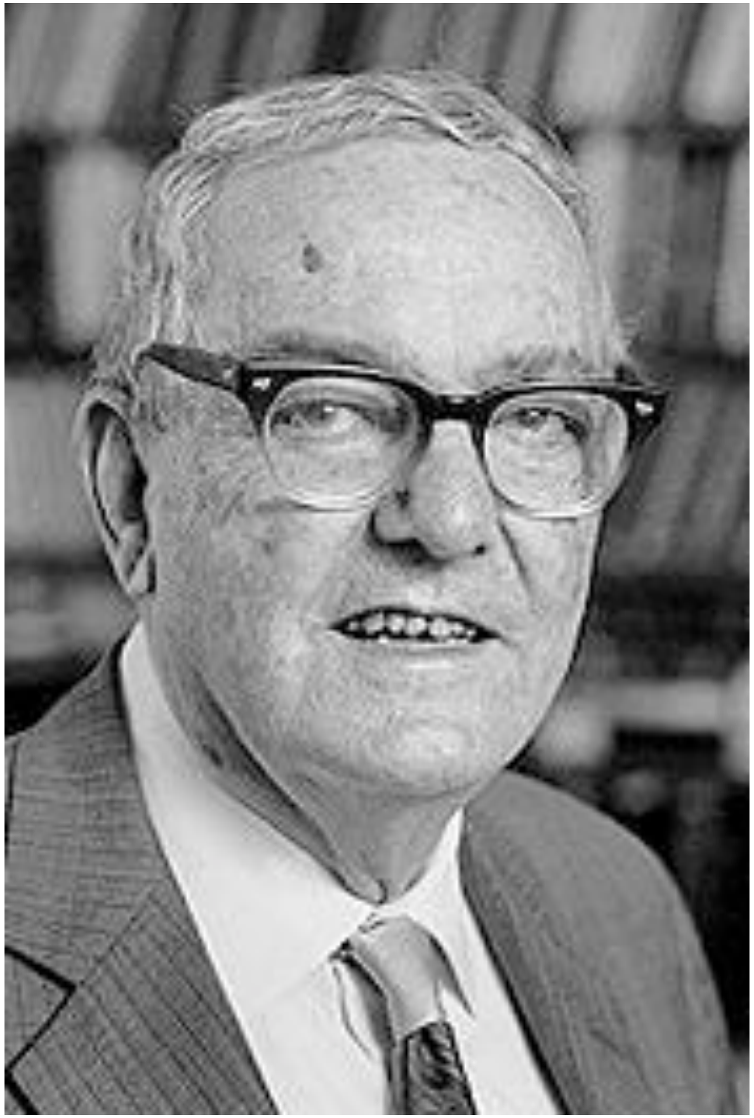


In the Beginning was the Word

- … and the word was “constraint”

- H. Simon, “The Structure of Ill-Structured Problems”, *Artificial Intelligence*, 4, 181-201, 1973
  - W.R. Reitman, *Cognition and Thought*, Wiley, New York, 1965

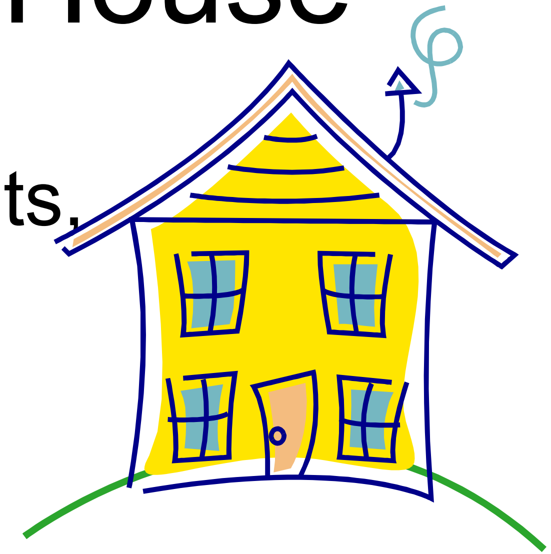
Herbert Simon



[Simon 73]: Constraints

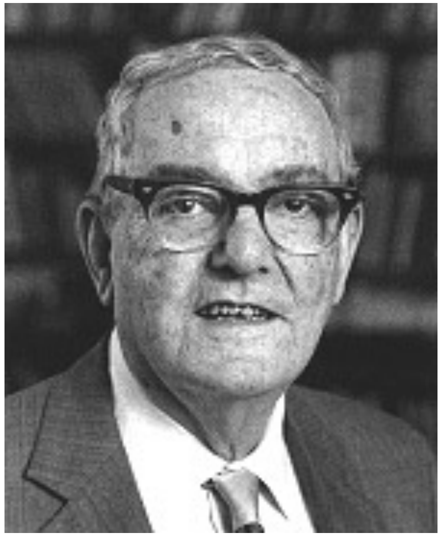
- “Reitman uses the term ‘constraints’ quite broadly to refer to any or all of the elements that enter into a definition of a problem.”
- [Reitman 65] “… even though [problem instances] would be considered complex, they include very few constraints as given. Composing a fugue is a good example. Here the main initial constraint … is that the end product be a fugue.”

[Simon 73]: Designing a House



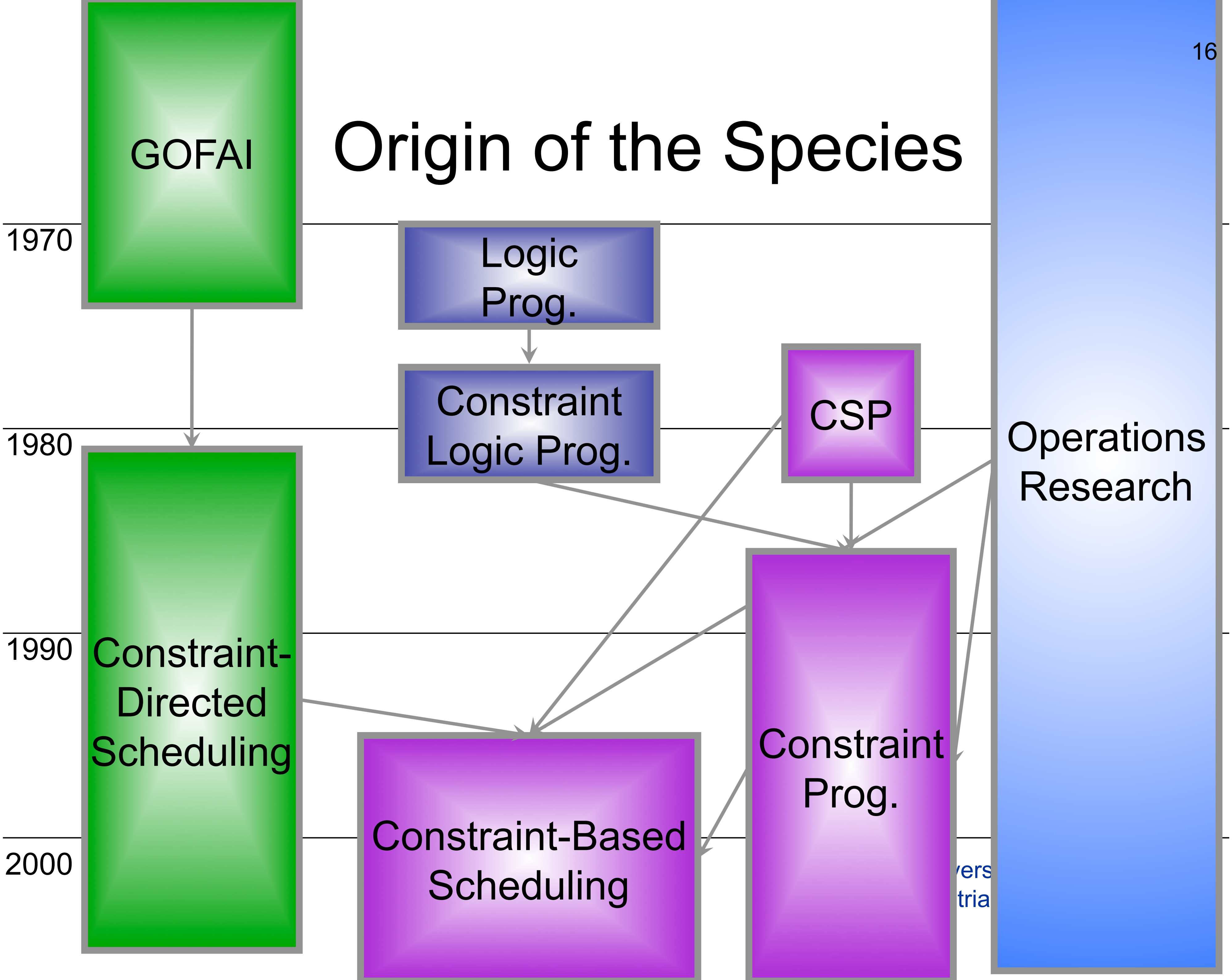
- Taking the initial goals and constraints, the architect begins to derive some global specifications from them – perhaps the square footage … of the house …. But the task itself, "designing a house", evokes from his long-term memory a list of other attributes that will have to be specified at an early stage of the design: characteristics of the lot on which the house is to be built, its general style, ….

Simon Says



- Constraints & goals evoke (or contain) ways to satisfy them (solution components)
- Solution components in turn create sub-goals and constraints
- Implications
  - Constraints are rich objects within a KR system
  - (Green) constraints don't look a lot like (purple) constraints

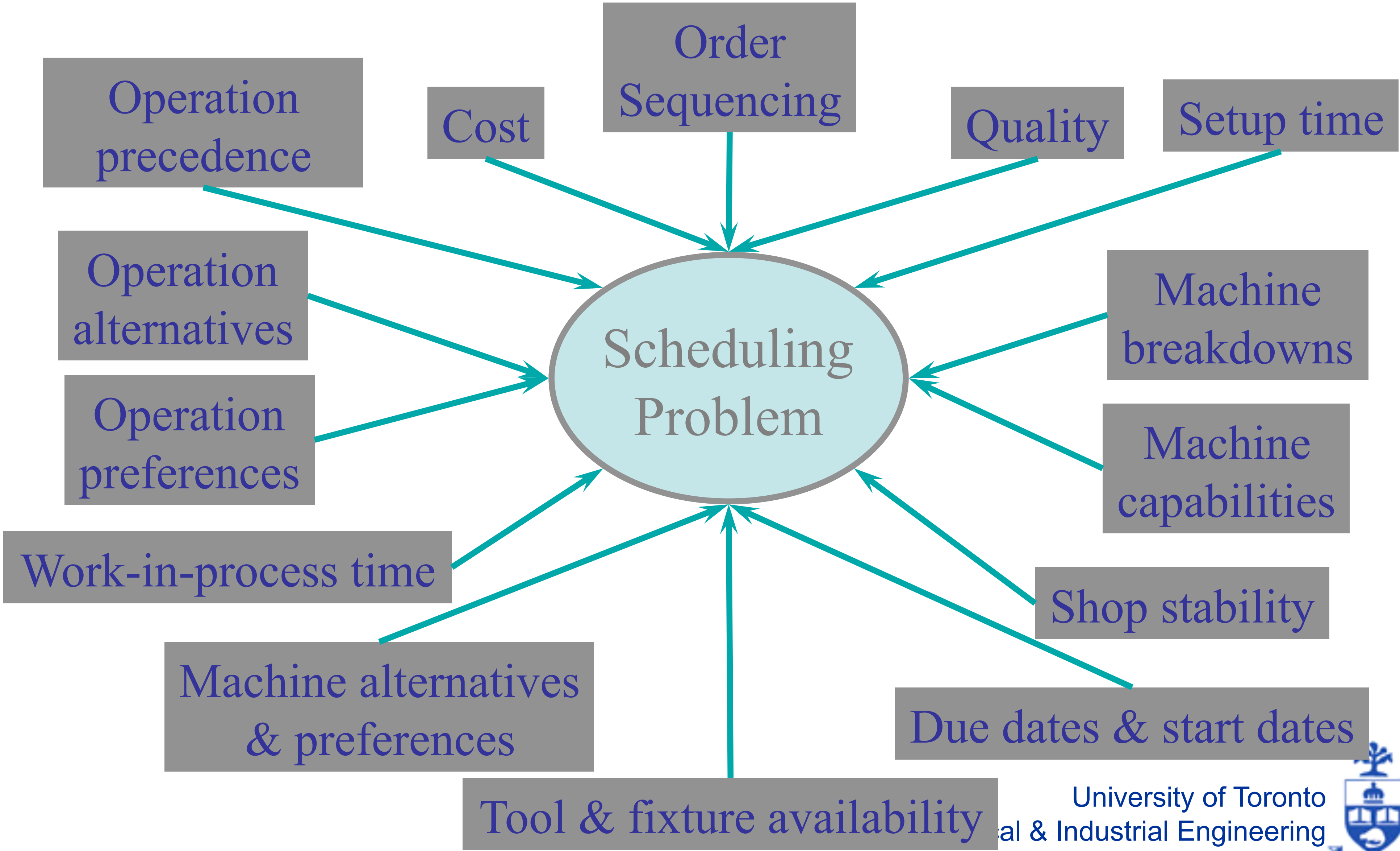
Origin of the Species



The Real Scheduling Problem?

- Fox, M., *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*, PhD Thesis, 1983.
- "… the [human] scheduler was spending 10%-20% of his time scheduling, and 80%-90% of his time communicating with other employees to determine what additional "constraints" could affect an order's schedule."

The Real Scheduling Problem?



Scheduling is …

- … a dynamic, multi-agent process that seeks to satisfy a diverse set of constraints from within (and beyond) an organization
- The real problem must be aware of:
  - organizational structure & authority
  - history & commitments
  - preferences
  - uncertainty & risk
  - …

Scheduling is …

- The allocation of resources to activities over time
  - Mixing machines in food manufacturing
  - Classrooms at a university
  - Trucks & planes for FedEx
- Mathematically hard
- Industrially, economically, & environmentally important

Constraints are…

- … key representations of all this knowledge to be exploited to heuristically guide the search for a solution

- Compare:
  - $c_i(v_0, v_2, v_4, v_{117}, \ldots)$ maps: $(D_0 \times D_2 \times D_4 \times D_{117} \times \ldots) \rightarrow \{T,F\}$

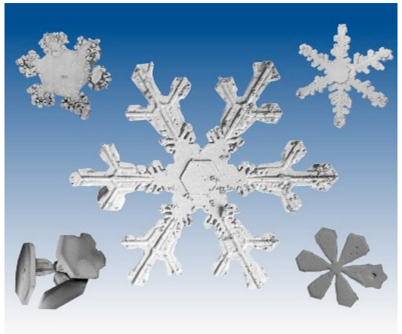
Constraint-Directed Scheduling

- System-wide reasoning
  - "anti-reductionist"
  - difficult to do controlled empirical analysis
  - difficult to generalize from success
  - difficult to publish traditional algorithmic papers
- Series of systems
  - ISIS, OPIS, Ozone, ….

Outline: Part 3

- The Origin of the Species
  - Ancient History (the 70s & 80s)
  - What's a constraint anyway?
- The 90s
- Scheduling & AI

"AI Winter"



- The "visions" of the 80s became hard to support as they were not being achieved
  - purple visions: declarative problem solving
  - green visions: system-wide reasoning
- Narrowing of ambitions to the easily testable and commercially rewarding
  - purple: CP becomes an organizational paradigm for OR algorithms
  - green: system building and the lure of the purple

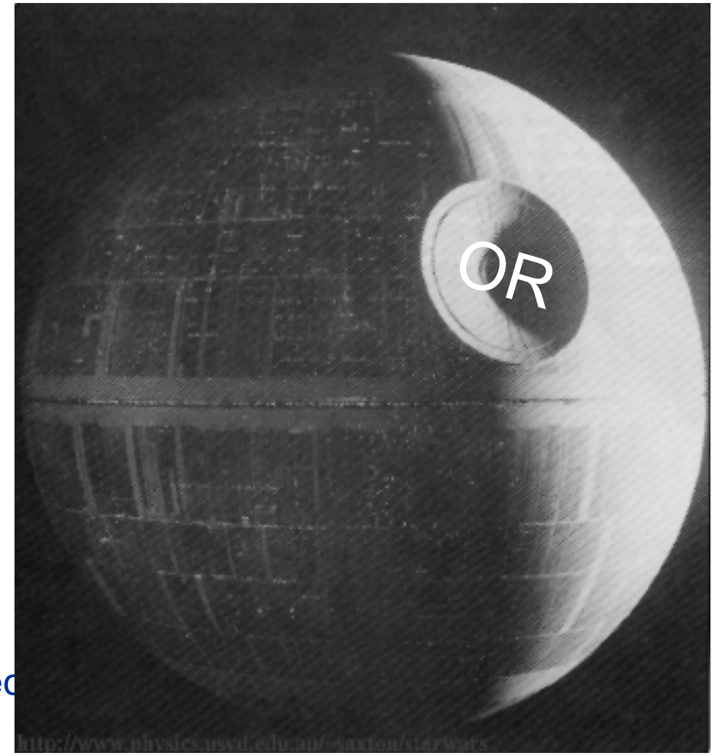
A bit of an exaggeration

Commercial Success

- Resource allocation system based on constraint propagation used in Desert Storm more than paid for all the DARPA AI research funding ever
  - Patrick Winston, 9th IEEE Conference on Artificial Intelligence for Applications, 1993
- ILOG Scheduler (started ~1994)
  - embedded in SAP and Oracle supply chain optimization products

The Darkside

- Have we solved the problem by ignoring those aspects that were interesting from an AI perspective to begin with?
- Competing with OR exactly where OR is strongest: well-defined "narrow" problems
  - "The darkside grows strong"



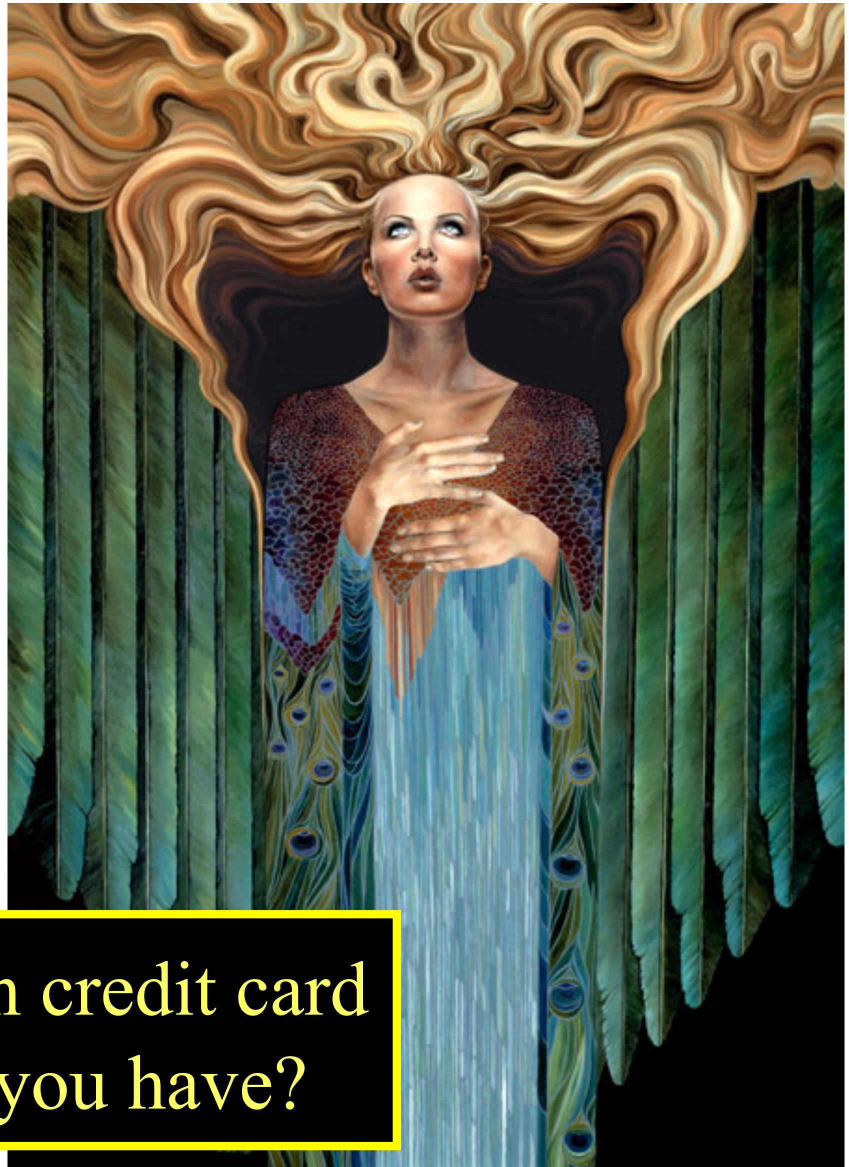
Mec

Outline: Part 2

- The Origin of the Species
  - Ancient History (the 70s & 80s)
  - What's a constraint anyway?
- The 90s
- Scheduling & AI

For the GOFAI Believers …

- Reasoning about time and resources is surely necessary for true AI
  - unclear that AI scheduling has developed anything cognitively meaningful
  - how do people reason about time and resources?



How much credit card debt do you have?

AI Scheduling Opportunities

- Richer Problem Models
  - robustness & uncertainty
  - alternative/optional activities … AI planning
- Meta-level Reasoning
  - Back to the Future?
- Information Engineering

Richer Problem Models: Uncertainty

- Don't know the activity duration, machines breakdown, new orders arrive, …
- Notion of a solution changes to the ongoing control of the schedule execution
- A bunch of work here both in AI and OR

Richer Activity Models

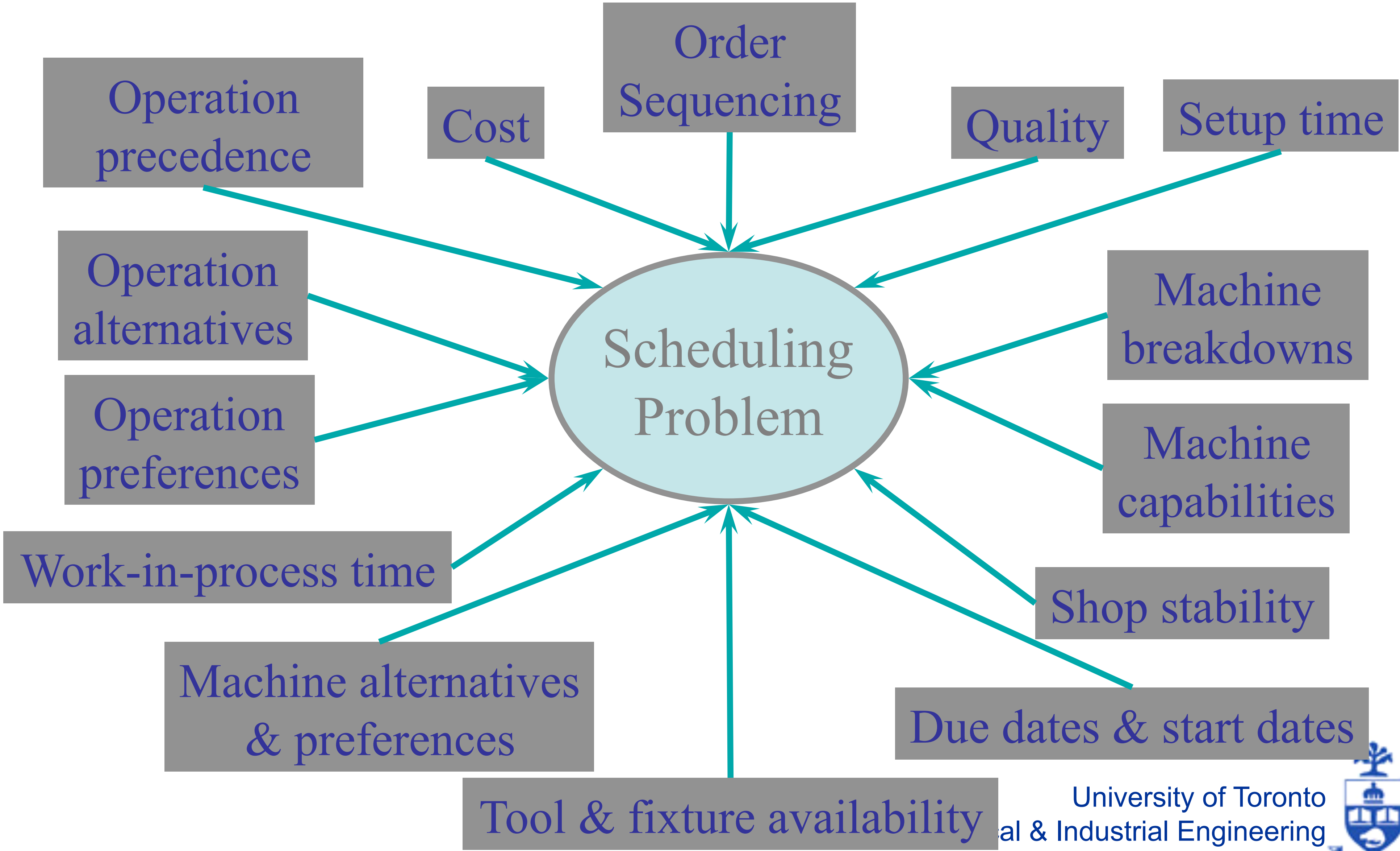
- Activity alternatives
- Cost/benefit or quality depends on exection time and resource choice

- AI planning & scheduling
  - a lot of work in planning with time & resources
  - scheduling with goals

Meta-level Reasoning

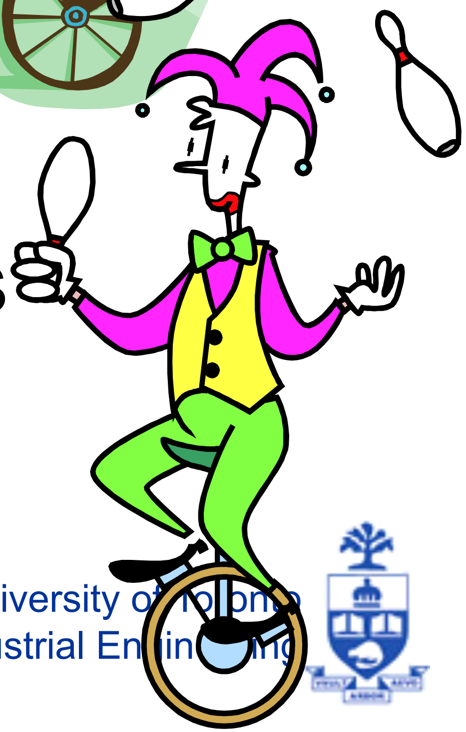
- Knowing when to use what algorithms
- Use machine learning to select the best algorithm or to form a control policy to switch among algorithms
  - much work here recently

Information Engineering



What Does a Human Scheduler Do?

- Negotiates
  - Can I deliver half now and half later?
- Prioritizes
  - Job X is more important because the customer is very big
- Spends money to relax constraints
  - Can we go below safety stock to meet this order?



Changing the Problem

- Traditional optimization techniques try to solve the problem → a human changes the problem so it can be solvable!
- What the human scheduler does is based on knowledge not represented in the scheduling problem!
  - Think of the experience and *information* that the human needs

Another View of Scheduling

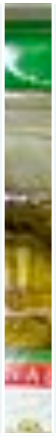
- We should be building information systems
  - that give humans the information required to make better decisions
  - that automate what the human scheduler really does

And Me?

Research Directions

AI

Planning with time and resources
Partial-order planning
Modeling in Planning
Design Systems for Planning



OR

Queueing theory and optimization
Constraint integer programming

Both

Uncertainty & robustness
Multi-agent, linked scheduling problems
Problem decomposition
Hybrid algorithms
Solution-guided search