Dependable Real-Time Systems

Lecture #7

Professor Jan Jonsson

Department of Computer Science and Engineering
Chalmers University of Technology

Generating schedules

Approaches for searching for a feasible schedule:

- Guided search
  - Organize the set of possible solutions in a search tree.
  - Traverse the search tree in a deterministic fashion.
  - Capable of finding an optimal solution (if one exists), at the price of exponential worst-case time complexity.
- Non-guided search
  - View the set of possible solutions as a search landscape.
  - Traverse the solution landscape using heuristics with elements of randomness.
  - Tractable time complexity, but no guarantee of optimality.

Guided search

Branch-and-bound algorithms:

- Basic idea:
  - A set of solutions to a given problem is organized in a search tree.
  - A vertex in the search tree corresponds to a specific solution structure.
  - A goal vertex corresponds to a complete solution to the problem and is located at the highest level of the search tree.
  - The root vertex corresponds to an initial solution at the lowest level of the search tree.
  - The search for a solution starts with only the root vertex.
  - Search objective is to find a goal vertex that optimizes a given cost (performance metric).

Guided search

Branch-and-bound algorithms:

- Basic idea (cont'd):
  - For each vertex, a set of child vertices is generated by modifying the structure of the current vertex ("branching").
  - To check if a tree branch may lead to an acceptable solution, a lower-bound function is applied to each of the child vertices.
  - If a child vertex looks promising, it will be further investigated.
  - If a child vertex will only lead to inferior solutions, that entire branch is pruned ("bounding").

Note: An initial solution could be used for making good bounding operations early in the search. When an acceptable goal vertex is reached the bounding operation can be made more accurate.

Guided search

Branch-and-bound algorithms:

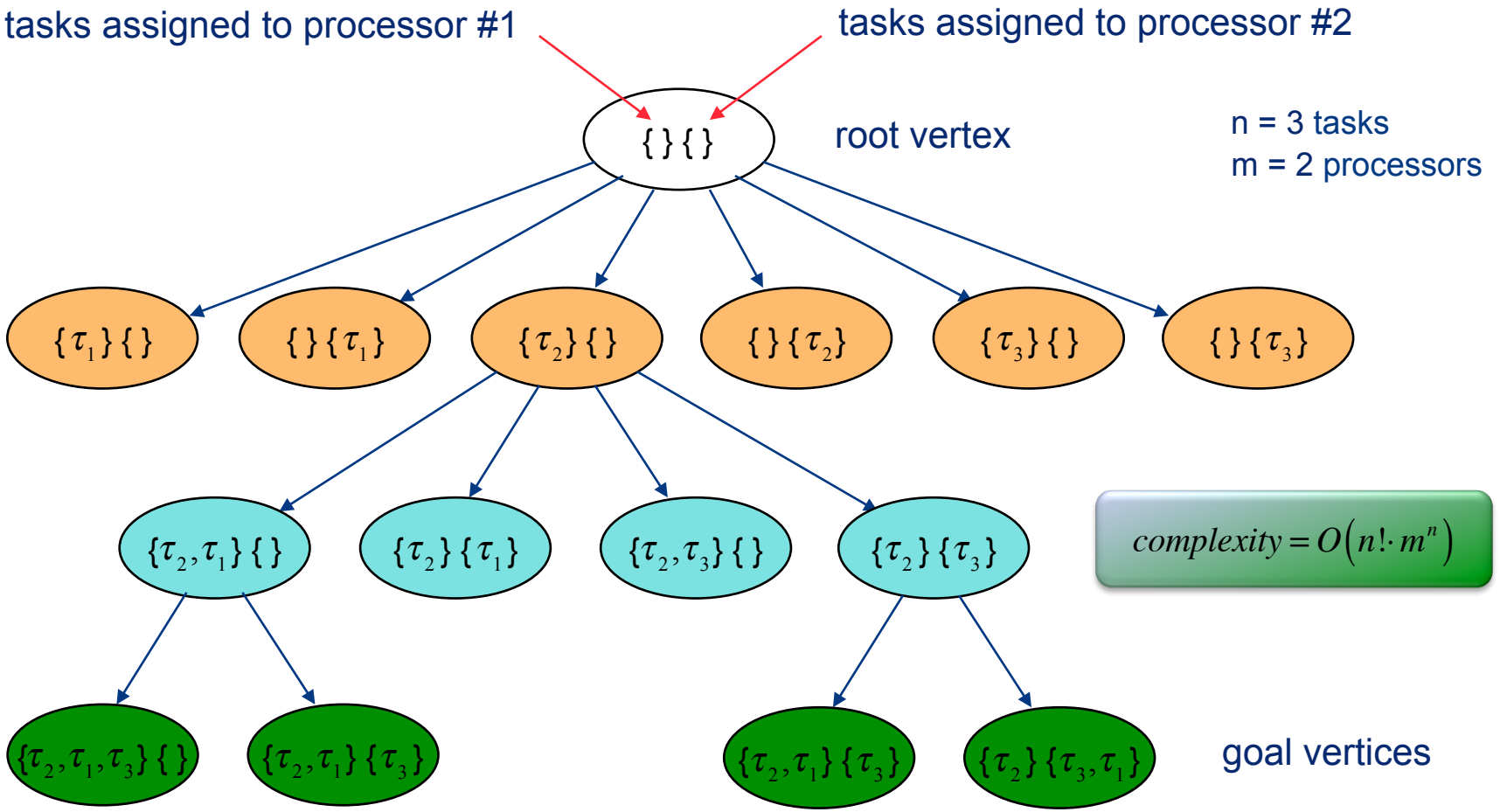
- Application to multiprocessor scheduling:
  - The search tree represents the set of all task-to-processor assignments for a given set of tasks and processors.
  - A vertex in the search tree is a partial or complete assignment of tasks to processors.
  - The root vertex corresponds to an initial (empty or complete) schedule.
  - A goal vertex corresponds to a complete schedule.
  - The purpose of the lower-bound function is to assess whether a child vertex is feasible, that is, whether the corresponding branch in the search tree contains a feasible schedule.

Guided search

Branch-and-bound for multiprocessor scheduling:

- Initial schedule is empty:
  - At each vertex in the search tree, a set of ready tasks (candidates for execution) are available for scheduling.
  - Generation of a child vertex corresponds to adding one of the ready tasks to the schedule in the current vertex.
- Initial schedule is complete (but possibly suboptimal):
  - At each level of the search tree, a set of scheduling changes (e.g., modified constraints or assignments) are available.
  - Generation of a child vertex corresponds to applying one or more of the changes to the schedule in the current vertex.

An example search tree



Guided search

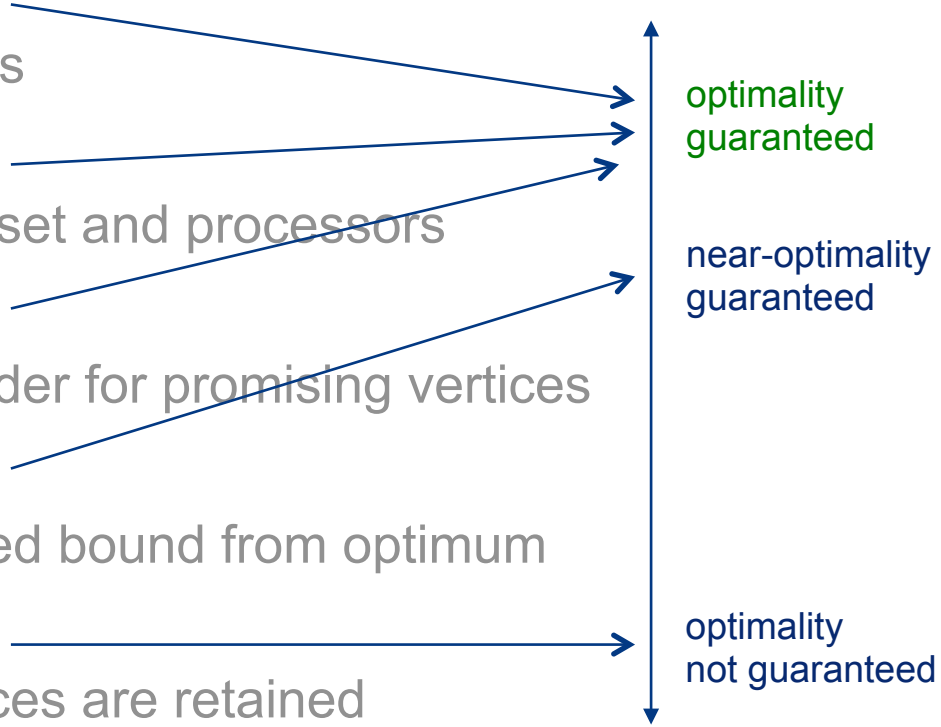
How do we avoid an exhaustive search?

- Bound pruning
  - use optimistic lower bounds
- Redundancy pruning
  - exploit symmetries in task set and processors
- Algorithm configuration
  - use suitable exploration order for promising vertices
- Performance guarantees
  - solution is within guaranteed bound from optimum
- Local optimization
  - only a subset of child vertices are retained

Guided search

How do we avoid an exhaustive search?

- Bound pruning
  - use optimistic lower bounds
- Redundancy pruning
  - exploit symmetries in task set and processors
- Algorithm configuration
  - use suitable exploration order for promising vertices
- Performance guarantees
  - solution is within guaranteed bound from optimum
- Local optimization
  - only a subset of child vertices are retained



Guided search

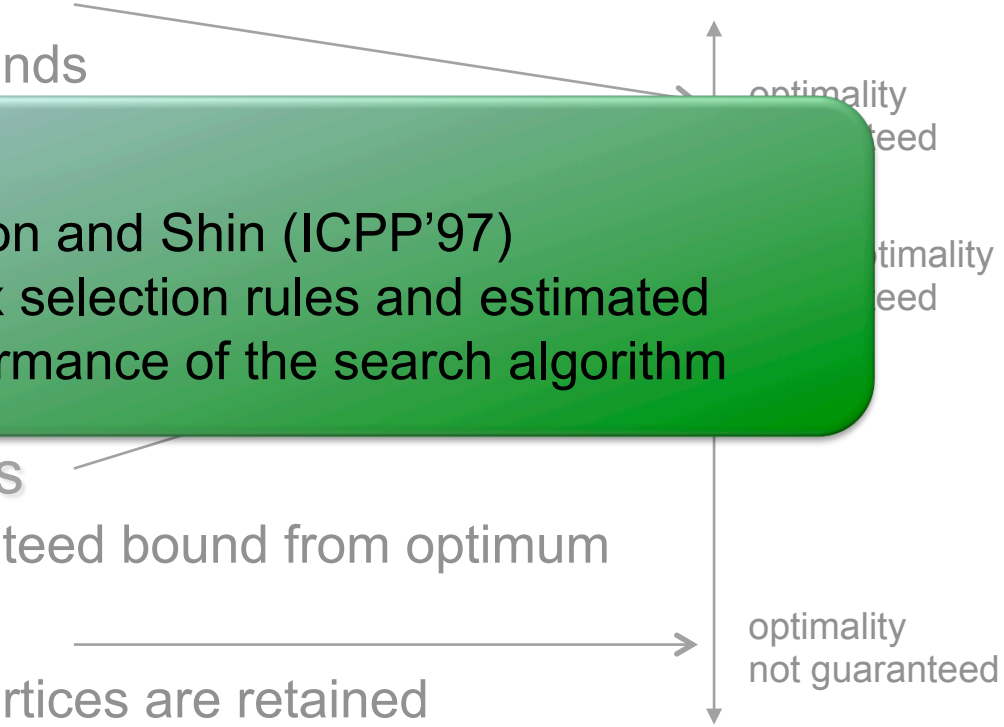
How do we avoid an exhaustive search?

- Bound pruning
  - use optimistic lower bounds
- Performance guarantees
  - solution is within guaranteed bound from optimum
- Local optimization
  - only a subset of child vertices are retained

Additional reading:

Read the paper by Jonsson and Shin (ICPP'97)

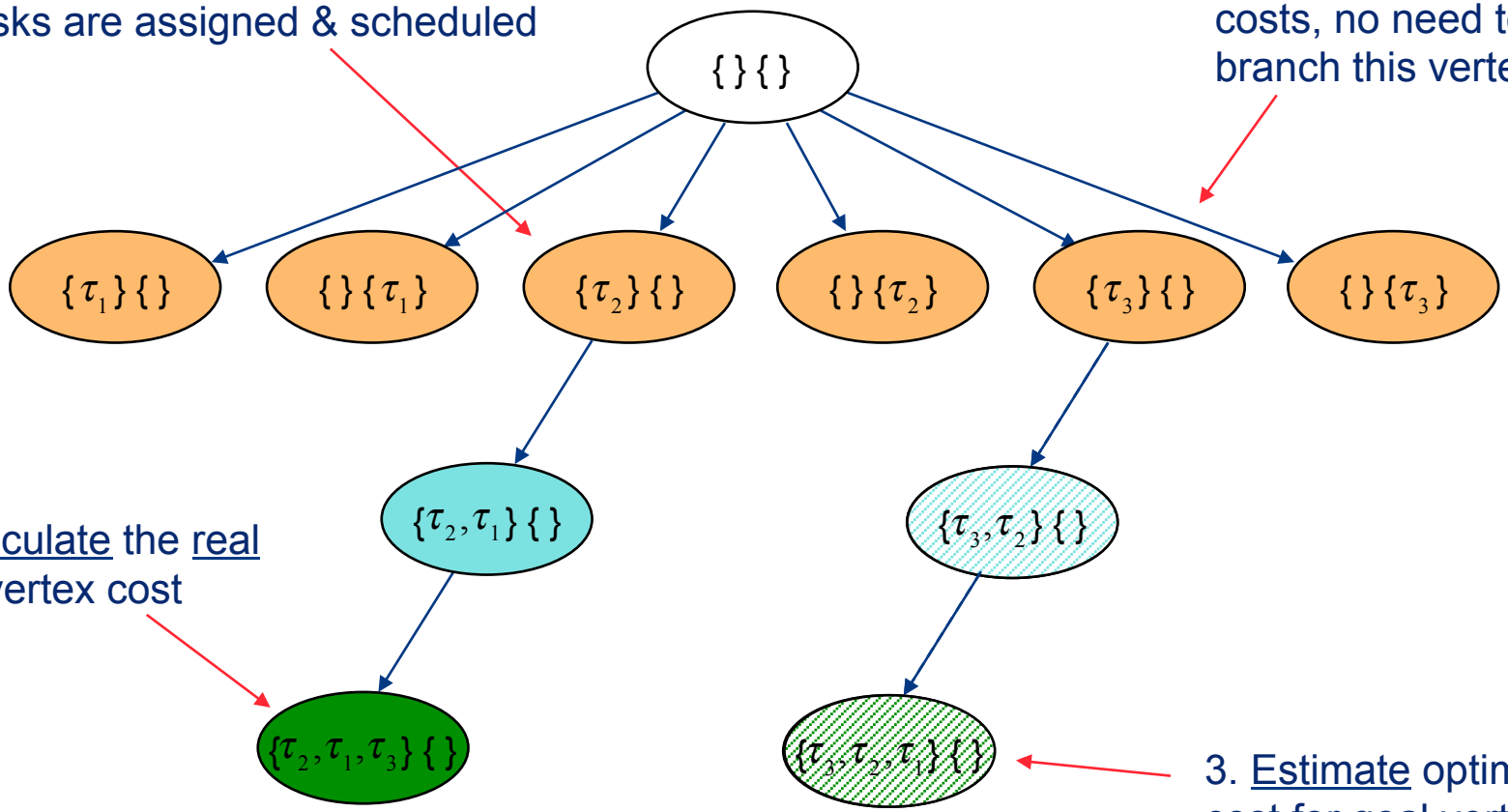
Study how different vertex selection rules and estimated bounds affect the performance of the search algorithm



An example of bound pruning

1. Assume this branch is chosen and all tasks are assigned & scheduled

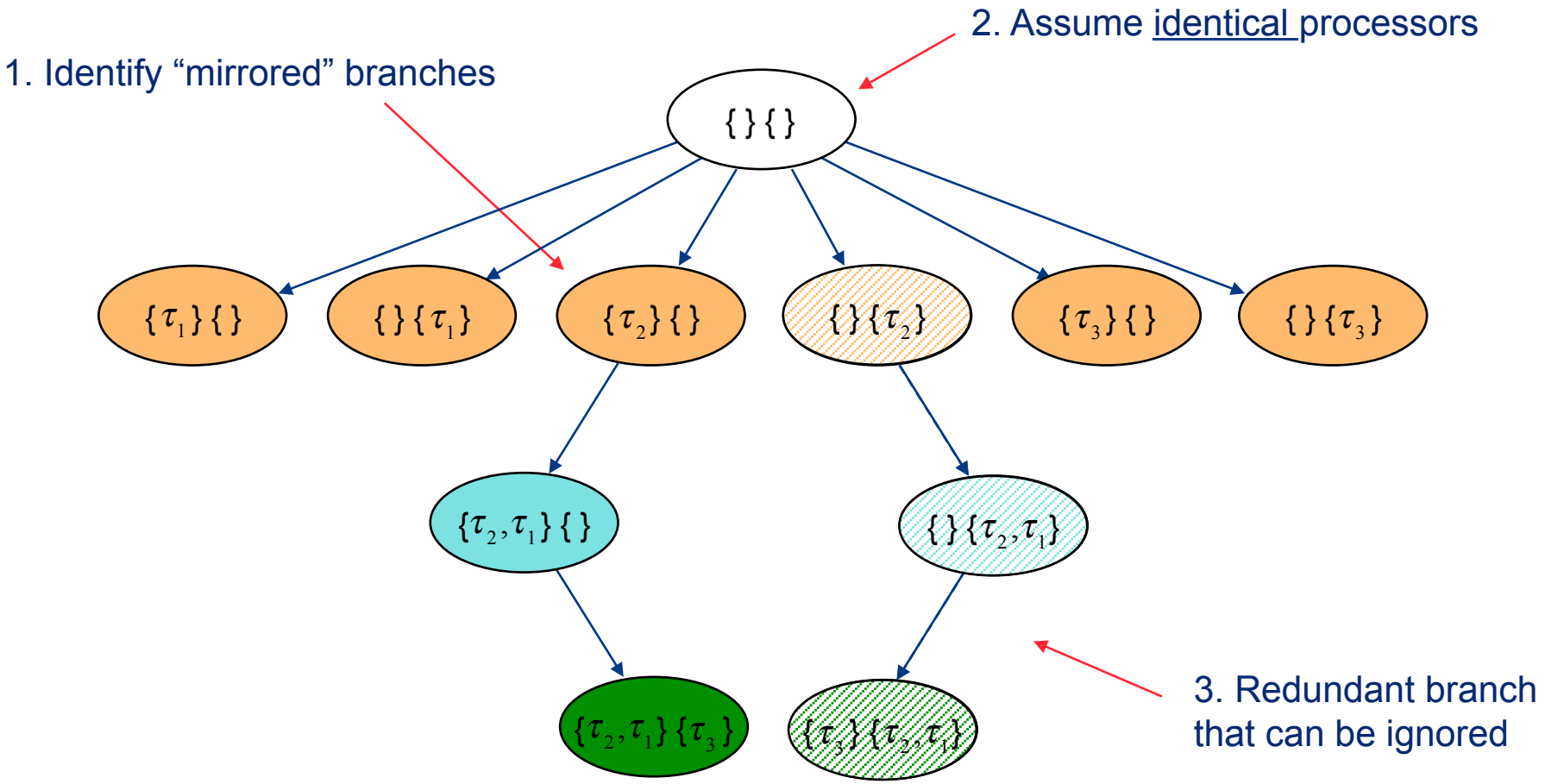
4. If all goal vertices originating from this vertex will have inferior costs, no need to further branch this vertex



2. Calculate the real goal vertex cost

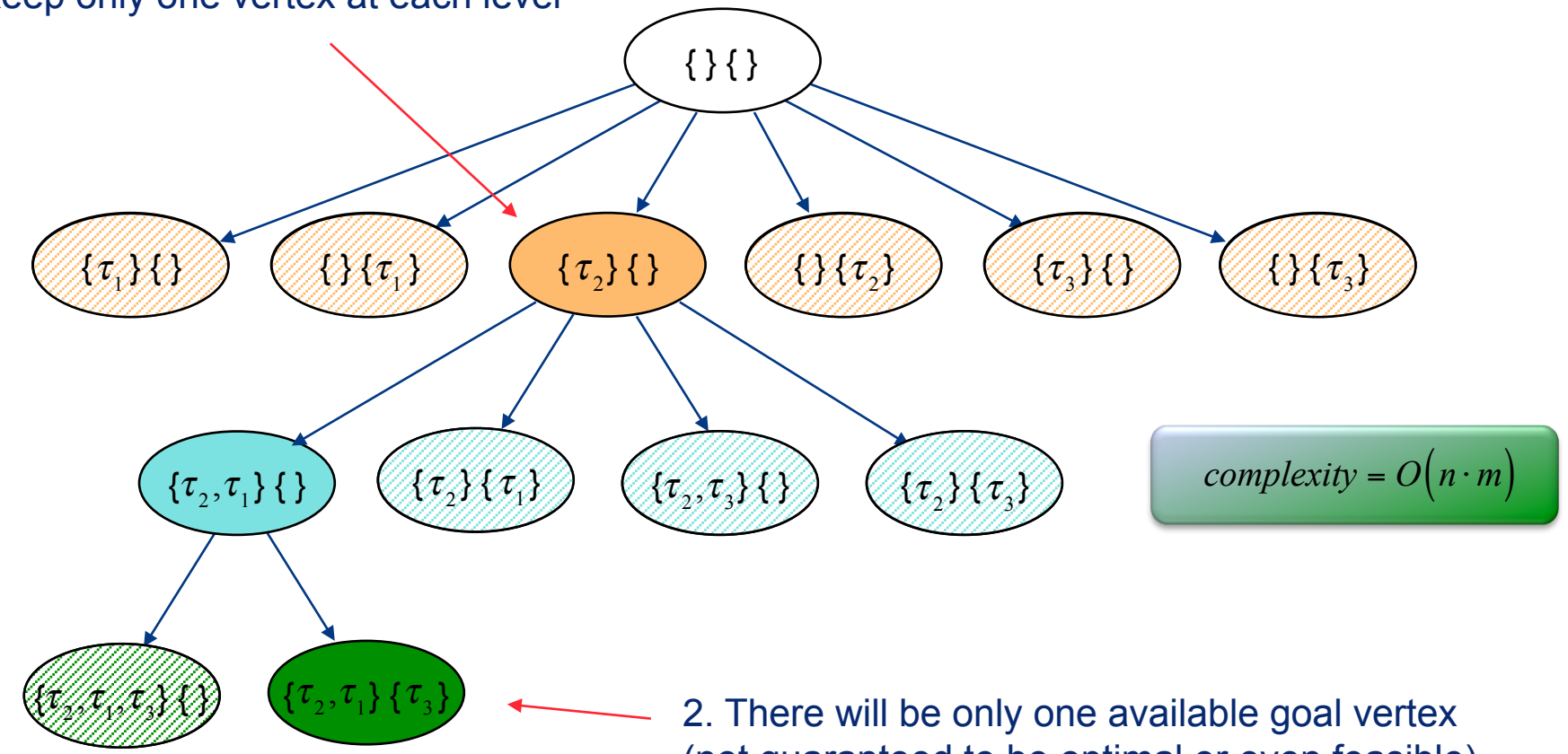
3. Estimate optimistic cost for goal vertex

An example of redundancy pruning



An example of local optimization

1. Keep only one vertex at each level



2. There will be only one available goal vertex
(not guaranteed to be optimal or even feasible)

Guided search

Example of a good local-optimization algorithm:

- Myopic scheduling: (Ramamritham, Stankovic and Shiah, 1990)
  - Promising vertices are explored in the order of decreasing search-tree level; within each level, exploration order is given by a heuristic function that calculates a weighted sum of task execution time, deadline, earliest start time and laxity.
  - Reduces search complexity by investigating only the *k* child vertices with closest deadline within each search-tree level.
  - Reduces search complexity by limiting the number of allowed backtracks (to vertices at lower search-tree levels)

$$complexity = O\left(k \cdot n \cdot m\right)$$

Guided search

Some (optimal) branch-and-bound algorithms:

- Single-processor constraint adjustment: (Xu and Parnas, 1990)
  - Minimizes maximum task lateness
  - Starts with an initial (complete) single-processor schedule
  - Modifies preemption, precedence and exclusion constraints of selected tasks to improve schedule quality
- Multiprocessor constraint adjustment: (Xu, 1993)
  - Minimizes maximum task lateness
  - Starts with an initial (complete) multiprocessor schedule
  - Modifies preemption, precedence and exclusion constraints of selected tasks to improve schedule quality

Guided search

Some (optimal) branch-and-bound algorithms:

- Quick-recovery algorithm: (Krishna and Shin, 1986)
  - Minimizes cost functions related to task lateness
  - Starts with an initial (complete) multiprocessor schedule
  - Examines gaps in the initial schedule and inserts passive backups of critical tasks to provide fault tolerance
- Replication-constrained allocation: (Hou and Shin, 1994)
  - Maximizes probability of no dynamic failure (probability that all deadlines are met in the presence of component failures)
  - Starts with an empty multiprocessor schedule
  - Inserts active backups of critical tasks, using either spatial or temporal replication depending on tightness of task deadline

Guided search

Quick-recovery algorithm: (Krishna & Shin, 1986)

Replicas of critical tasks are called clones. A primary clone is executed in the normal course of things. A ghost clone is a passive backup task which lies domant until it is activated to take the place of a corresponding primary whose processor has failed.

For reliability reasons, the system runs a certain number $n(i)$ of clones of critical task i in parallel on separate processors.

A system is said to sustain up to N_{sust} failures if, despite the failure of up to N_{sust} processors in any sequence, the system is able to schedule critical tasks so that $n(i)$ clones of task i can be executed in parallel without deadlines being missed.

Guided search

Quick-recovery algorithm:

Necessary and sufficient conditions for reliability guarantee:

C1: Each critical task must have ghost clones scheduled on N_{sust} processors, and a ghost and a primary of the same critical task may not be scheduled on the same processor.

C2: Ghosts are conditionally transparent:

- two ghost clones may overlap in the schedule if none of their corresponding primary clones are scheduled on the same processor
- primary clones may overlap ghosts on the same processor only if there is sufficient slack in the schedule to continue to meet the deadlines of all the primary and activated ghosts on that processor

Guided search

Replication-constrained allocation: (Hou & Shin, 1994)

For reliability reasons, certain critical tasks must have N_{repl} replicas. The value of N_{repl} is common for all critical tasks.

The replicas can be created in one of two ways:

- **R1:** 1 primary and N_{repl} - 1 active backups on separate processors
- **R2:** 1 primary and N_{repl} - 1 active backups on one processor

Task deadlines decide whether R1 or R2 is used for replication:

a) if task deadline is loose enough to allow for execution of both the primary and the N_{repl} - 1 backups before the deadline, R2 is chosen

b) otherwise, R1 is chosen.

Guided search

Replication-constrained allocation:

A B&B algorithm is applied whose objective is to maximize the probability of no dynamic failure, P_{ND}, which is the probability that all tasks within one LCM period meet their deadlines even in the presence of processor or communication-link failures.

Note: When the degree of replication is increased, reliability of the system is increased, whereas the schedulability is decreased. The probability of no dynamic failure reflects both reliability and schedulability with a bias towards schedulability.

Generating schedules

Approaches for searching for a feasible schedule:

- Guided search
  - Organize the set of possible solutions in a search tree.
  - Traverse the search tree in a deterministic fashion.
  - Capable of finding an optimal solution (if one exists), at the price of exponential (worst case) time complexity.
- Non-guided search
  - View the set of possible solutions as a search landscape.
  - Traverse the solution landscape using heuristics with elements of randomness.
  - Reasonable time complexity, but no guarantee of optimality.

Non-guided search

General characteristics:

- Each non-guided search is given an initial task-to-processor assignment from which the search starts.
- Within each iteration step during search, different derivable alternatives of changing the current assignment are examined.
- To check whether an alternative is feasible or not, a run-time efficient feasibility test has to be used.
- In order to help the search find better assignments, the number of deadline misses is included as a penalty into the function calculating the goodness of the assignment.

Non-guided search

Examples:

- Simulated annealing
- Genetic optimization

- Tabu search
- Neighbourhood search
- ...

These techniques all have in common that it is sufficient to state what makes a good solution, not how to get one!

Non-guided search

Simulated annealing: (Kirkpatrick, Gelatt and Vecchi, 1983)

- Basic idea:
  - Simulated annealing is a global optimization technique which borrows ideas from statistical physics. The technique is derived from observations of how slowly-cooled molten metal can result in a regular crystalline structure.
  - The salient property of the technique is the incorporation of random jumps from local minima to potential new solutions. As the algorithm progresses, this ability is lessened, by reducing a temperature factor, which makes larger jumps less likely.
  - The main objective of the technique is to find the lowest point in an energy landscape.

Non-guided search

Simulated annealing:

- Application to multiprocessor scheduling:
  - The set of all task-to-processor assignments for a given set of task and processors is called the problem space. A point in the problem space is an assignment of tasks to processors.
  - The neighbor space of a point is the set of points that are reachable by moving any single task to any other processor.
  - The energy of a point in problem space is a measure of the goodness of the task assignment represented by that point.
  - The energy function determines the shape of the problem space. It can be visualized as a rugged landscape, with deep valleys representing good solutions, and high peaks representing poor or infeasible ones.

Non-guided search

Simulated annealing:

- Algorithm:

  A random starting point is chosen, and its energy E_s is evaluated. A random point in the neighbor space is then chosen, and its energy E_n is evaluated. This point becomes the new starting point if either $E_n \leq E_s$, or if $E_n > E_s$ and

  $$e^x \geq \text{random}(0,1) \quad \text{where } x = -(E_n - E_s)/C$$

  The control variable C is analogous to the temperature factor in a thermodynamic system. During the annealing process, C is slowly reduced (cooling the system), making higher energy jumps less likely. Eventually, the system freezes into a low energy state.

Non-guided search

Simulated annealing:

- Implementation: (Tindell, Burns & Wellings, 1992)

  Neighbor function: Choose a random task and move it to a randomly-chosen processor.

  Energy function: The weighted sum of the following characteristics of the assignment:

  - Number of tasks assigned to the wrong processor
  - Number of replicas assigned to the same processor
  - Number of processors with too high a memory utilization
  - Number of tasks which do not meet their deadlines
  - Total communication bus utilization

Non-guided search

Genetic optimization: (Goldberg, 1989)

- Basic idea:
  - Based on Darwin's evolution theory: "Survival of the Fittest"
  - Solutions to a problem is viewed as individuals forming a population.
  - Pair of individuals can create children (new individuals)
  - New individuals are created by applying a crossover operator to the genes of the parents
  - Genes of a new individual may mutate

Non-guided search

Genetic optimization:

- Application to multiprocessor scheduling:
  - Tasks assignments and orderings are viewed as "chromosomes"
  - Tasks represent "genes"
  - Mutation means that a task is moved to another processor