CHALMERS

Multiprocessor scheduling

How are tasks assigned to processors?

- Static assignment
  - The processor(s) used for executing a task are determined before system is put in mission ("off-line")
  - Algorithms: partitioned scheduling, guided search, non-guided search, ...
- Dynamic assignment
  - The processor(s) used for executing a task are determined during system operation "on-line"
  - Algorithms: global scheduling

CHALMERS

Multiprocessor scheduling

How are tasks allowed to migrate?

- Partitioned scheduling
  - Each instance of a task must execute on the same processor
  - Equivalent to multiple uniprocessor systems!
- Guided search & non-guided techniques
  - Depending on migration constraints, a task may or may not execute on more than one processor
- Global scheduling
  - A task is allowed to execute on an arbitrary processor (sometimes even after being preempted)

CHALMERS

Partitioned scheduling

General characteristics:

- Each processor has its own queue for ready tasks
- Tasks are organized in groups, and each task group is assigned to a specific processor
- When selected for execution, a task can only be dispatched to its assigned processor

CHALMERS

Partitioned scheduling

Advantages:

- Mature scheduling framework
  - Most scheduling theory pertaining to uniprocessor scheduling are also applicable here
  - Uniprocessor resource-management protocols can be used
- Partitioning of tasks can be automated
  - For example, using a bin-packing algorithm

Disadvantages:

- Cannot exploit all unused execution time
  - Surplus capacity cannot be shared among processors
  - Will suffer from overly-pessimistic WCET derivation

CHALMERS

Partitioned scheduling

Complexity of schedulability analysis for partitioned scheduling: (Leung & Whitehead, 1982)

The problem of deciding whether a task set is schedulable on m processors with respect to partitioned scheduling is NP-complete in the strong sense.

Consequence:

There cannot be any pseudo-polynomial time algorithm for finding an optimal partition of a set of tasks unless P = NP.

CHALMERS

Partitioned scheduling

Bin-packing algorithms:

- Basic idea:
  - The problem concerns packing objects of varying sizes in boxes ("bins") with the objective of minimizing number of used boxes.
- Application to multiprocessor systems:
  - Bins are represented by processors and objects by tasks.
  - The decision whether a processor is "full" or not is derived from a utilization-based feasibility test.
- Assumptions:
  - Independent, periodic tasks
  - Preemptive, uniprocessor scheduling (RM)

CHALMERS

Partitioned scheduling

Bin-packing algorithms:

Rate-Monotonic-First-Fit (RMFF): (Dhall and Liu, 1978)

- Let the processors be indexed as $\mu_1, \mu_2, \ldots$
- Assign the tasks in the order of increasing periods (that is, RM order).
- For each task τ_i, choose the lowest previously-used j such that τ_i, together with all tasks that have already been assigned to processor μ_j, can be feasibly scheduled according to the utilization-based RM-feasibility test.
- Processors are added if needed for RM-schedulability.

CHALMERS

Partitioned scheduling

Bin-packing algorithms:

The utilization guarantee bound U_{RMFF} for a system with m processors using the RMFF scheduling policy (with arbitrary task-assignment order) is

$$m\left(2^{1/2}-1\right) \le U_{RMFF} \le (m+1)/\left(1+2^{1/(m+1)}\right)$$ (Oh & Baker, 1998)

Note: $\left(2^{1/2}-1\right) \approx 0.41$

Thus: task sets whose utilization do not exceed ≈ 41% of the total processor capacity is always RMFF-schedulable.

CHALMERS

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 measure).

CHALMERS

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.

CHALMERS

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.

CHALMERS

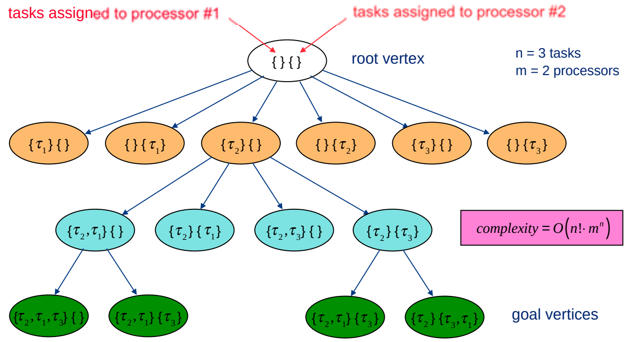
Guided search

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.

CHALMERS

An example search tree



CHALMERS

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

optimality guaranteed

near-optimality guaranteed

optimality not guaranteed

CHALMERS

Guided search

How do we avoid an exhaustive search?

- Bound pruning
  - use optimistic lower bounds

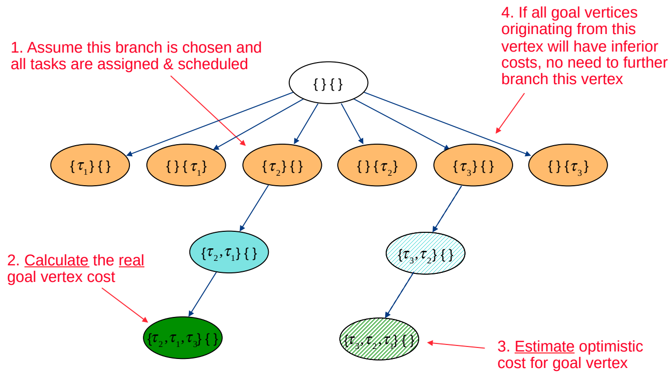
Additional reading:
Read the paper by Jonsson and Shin (ICPP'97)
Study how different vertex selection rules and estimated bounds affect the performance of the search algorithm

- Performance guarantees
  - solution is within guaranteed bound from optimum
- Local optimization
  - only a subset of child vertices are retained

optimality not guaranteed

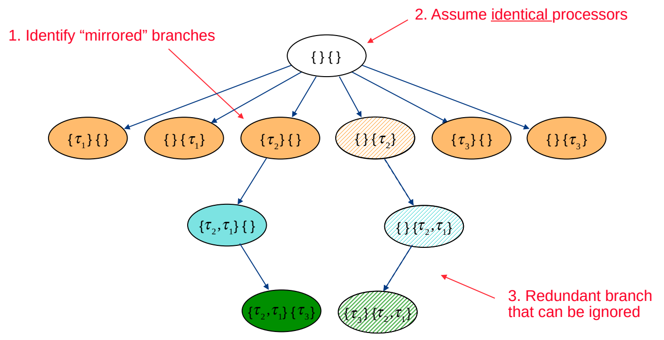
CHALMERS

An example of bound pruning



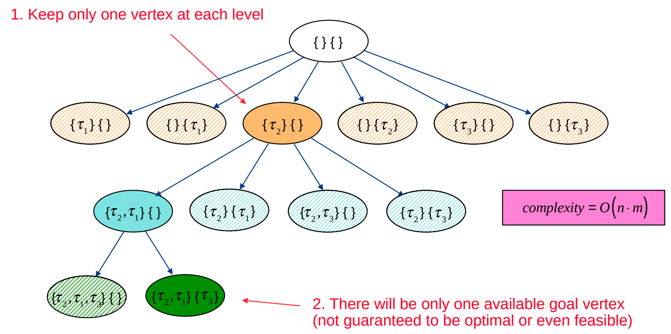
CHALMERS

An example of redundancy pruning



CHALMERS

An example of local optimization



CHALMERS

Guided search

Some (optimal) branch-and-bound algorithms:

- Distributed real-time systems: (Peng and Shin, 1989)
  - Minimizes system hazard (maximum normalized task response time)
  - Starts with an empty schedule
- Fault-tolerant real-time systems: (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 schedule
  - May change degree of replication and restart the algorithm

CHALMERS

Guided search

Some (optimal) branch-and-bound algorithms:

- Uniprocessor real-time systems: (Xu and Parnas, 1990)
  - Minimizes maximum task lateness
  - Starts with an initial (complete) schedule
  - Modifies preemption, precedence and exclusion constraints
- Multiprocessor real-time systems: (Xu, 1993)
  - Minimizes maximum task lateness
  - Starts with an initial (complete) schedule
  - Modifies preemption, precedence and exclusion constraints

CHALMERS

Guided search

Some good local-optimization algorithms:

- 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.
  - Lower-bound function determines for the current vertex whether it is strongly feasible, that is, whether a feasible schedule can be obtained by expanding any of its child vertices.
  - Reduces search complexity by only investigating the *k* child vertices with closest deadline in the check for strong feasibility.
  - Reduces search complexity by limiting the number of allowed backtracks (to vertices at lower search-tree levels)

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

CHALMERS

Guided search

Some good local-optimization algorithms:

- Pair-wise clustering: (Ramamritham, 1995)
  - Promising vertices are explored in the order of decreasing search-tree level; within each level, exploration is made in the order of increasing task LFT (latest finishing time).
  - Lower-bound function determines for the current vertex whether it is feasible using simple heuristics that keep track of latest start time and available time resources.
  - LFT is derived from task set end-to-end deadlines.
  - Pairs of communicating tasks are clustered based on the communication volume ratio. If the ratio between the task pair's execution times and communication volume is below a certain bound, the two tasks are assigned to the same processor.

CHALMERS

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.

CHALMERS

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!

CHALMERS

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.

CHALMERS

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.

CHALMERS

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

  $e^x \geq \text{random}(0,1)$ 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.

CHALMERS

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

CHALMERS

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

CHALMERS

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