Introduction to Parallel Computing

George Karypis

Search Algorithms for Discrete Optimization Problems

Overview

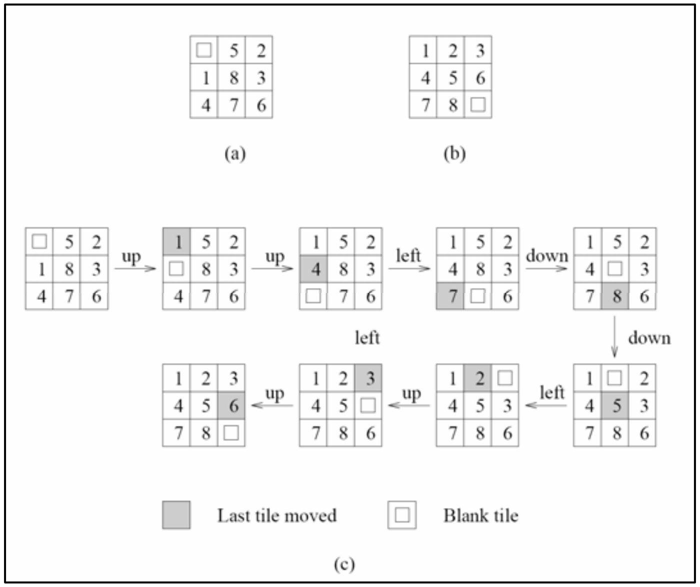
- What is a Discrete Optimization Problem
- Sequential Solution Approaches
- Parallel Solution Approaches
- Challenges

Discrete Optimization Problems

- A discrete optimization problem (DOP) is defined as a tuple of (S, f)
  - S : The set of feasible states
  - f : A cost function $f : S \to R$
- The objective is to find the optimal solution x_{opt} in S such that $f(x_{opt})$ is maximum over all solutions.
- Examples:
  - 0/1 integer linear programming problem
  - 8-puzzle problem

Examples

- 0/1 Linear integer problem:
  - Given an *mxn* matrix *A*, vectors *b* and *c*, find vector *x* such that
    - *x* contains only 0s and 1s
    - $Ax > b$
    - $f(x) = x^T c$ is maximized.
- 8-puzzle problem:
  - Given an initial configuration of an 8-puzzle find the shortest sequence of moves that will lead to the final configuration.

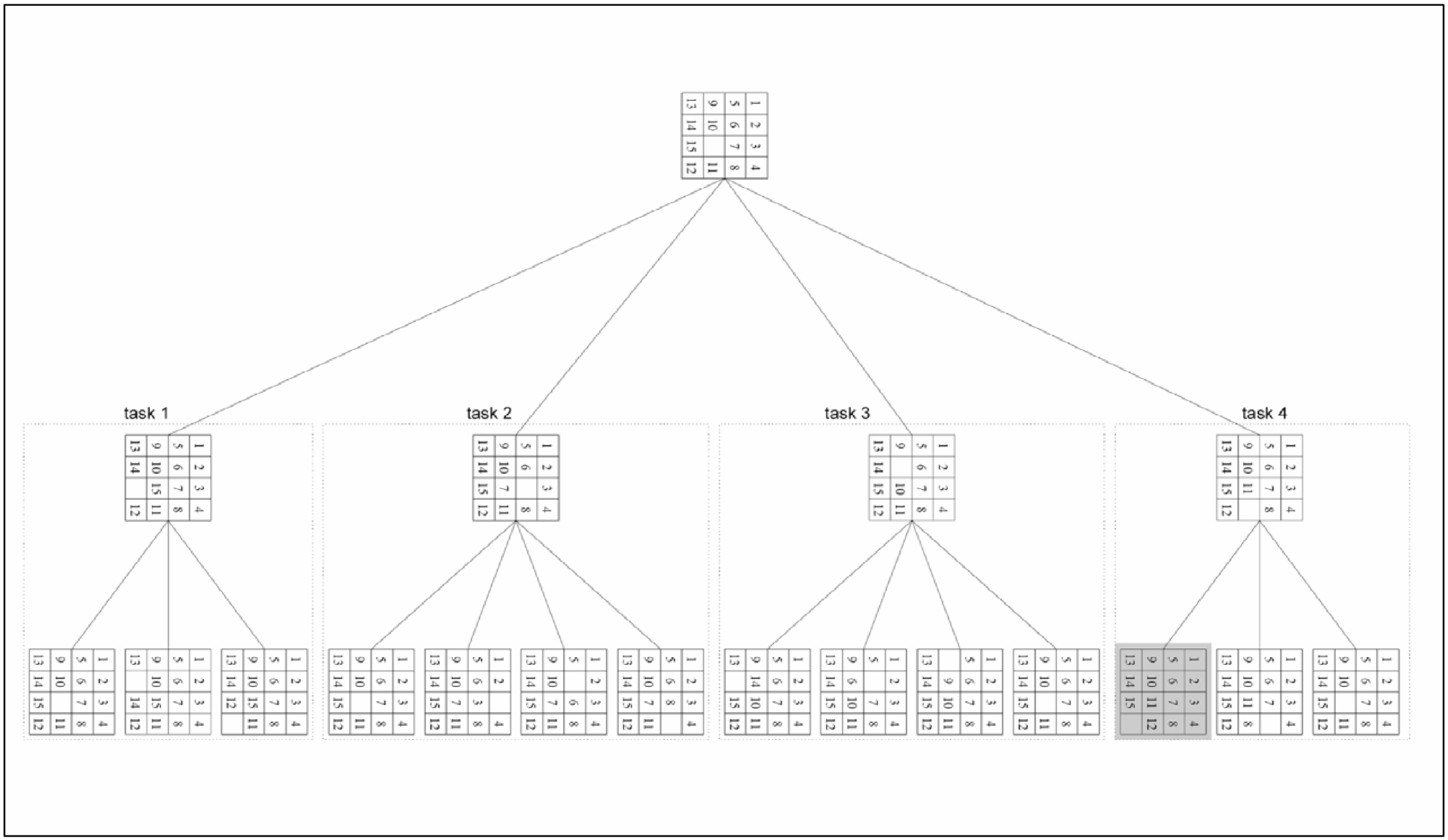


DOP & Graph Search

- Many DOP can be formulated as finding the a minimum cost path in a graph.
  - Nodes in the graph correspond to states.
  - States are classified as either
    - terminal & non-terminal
  - Some of the states correspond to feasible solutions whereas others do not.
  - Edges correspond to “costs” associated with moving from one state to the other.
- These graphs are called *state-space* graphs.

Examples of State-Space Graphs

- 15-puzzle problem:



Examples of State-Space Graphs

- 0/1 Linear integer programming problem
  - States correspond to partial assignment of values to components of the *x* vector.

Example 11.3 The 0/1 integer-linear-programming problem revisited
Consider an instance of the 0/1 integer-linear-programming problem defined in Example 11.1. Let the values of A, b, and c be given by

$$A = \begin{bmatrix} 5 & 2 & 1 & 2 \\ 1 & -1 & -1 & 2 \\ 3 & 1 & 1 & 3 \end{bmatrix}, \; b = \begin{bmatrix} 8 \\ 2 \\ 5 \end{bmatrix}, \; c = \begin{bmatrix} 2 \\ 1 \\ -1 \\ -2 \end{bmatrix}.$$

The constraints corresponding to A, b, and c are as follows:

$$\begin{aligned} 5x_1 + 2x_2 + x_3 + 2x_4 &\geq 8 \\ x_1 - x_2 - x_3 + 2x_4 &\geq 2 \\ 3x_1 + x_2 + x_3 + 3x_4 &\geq 5 \end{aligned}$$

and the function $f(x)$ to be minimized is

$$f(x) = 2x_1 + x_2 - x_3 - 2x_4.$$

$$\sum_{x_j \text{ is free}} \max\{A[i,j], 0\} + \sum_{x_j \text{ is fixed}} A[i,j]x_j \;\geq\; b_i, \quad i = 1, \ldots, m$$

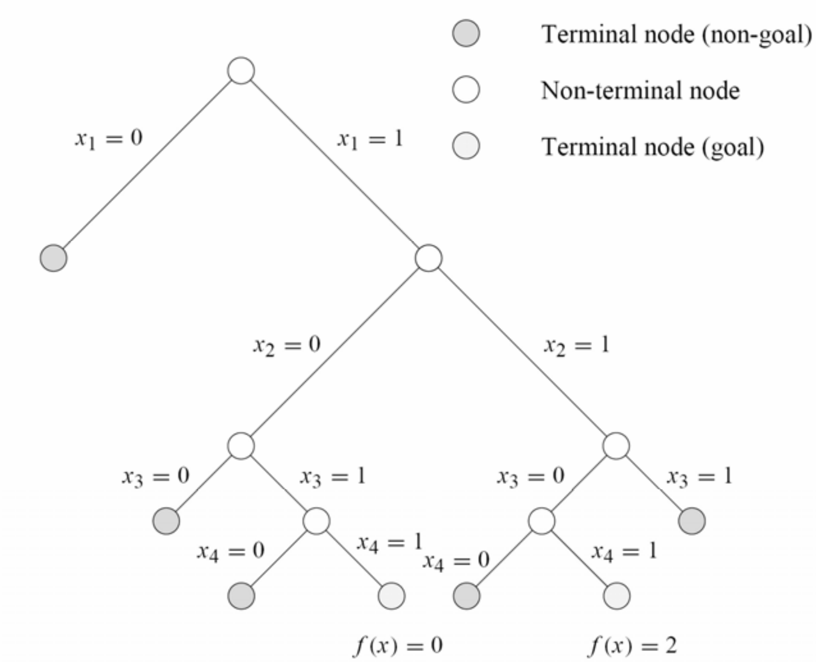


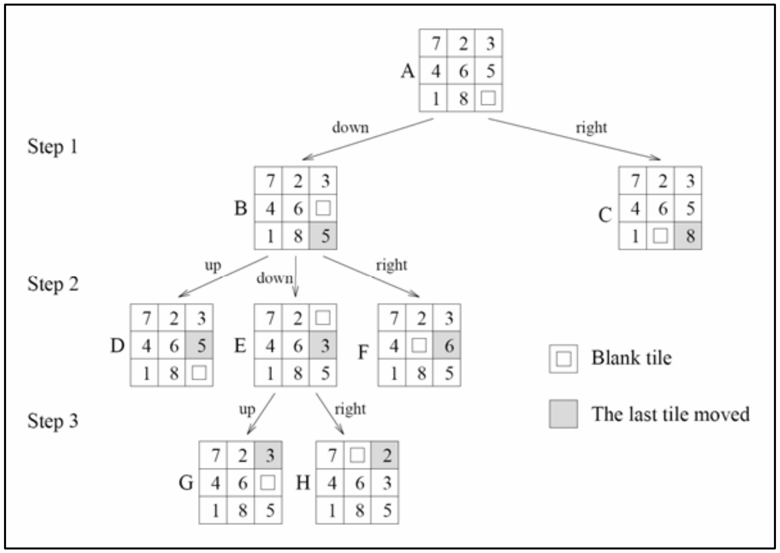
Figure 11.2 The graph corresponding to the 0/1 integer-linear-programming problem.

Exploring the State-Space Search

- The solution is discovered by exploring the state-space search.
  - Exponentially large
    - Heuristic estimates of the solution cost are used.
      - Cost of reaching to a feasible solution from current state x is
        - $l(x) = g(x) + h(x)$
- *Admissible* heuristics are the heuristics that correspond to lower bounds on the actual cost.
  - Manhattan distance is an admissible heuristic for the 8-puzzle problem.
- Idea is to explore the state-space graph using heuristic cost estimates to guide the search.
  - Do not spend any time exploring "bad" or "unpromising" states.

Exploration Strategies

- Depth-First
  - Simple & Ordered Backtracking
  - Depth-First Branch-and-Bound
    - Partial solutions that are inferior to the current best solutions are discarded.
  - Iterative Deepening A*
    - Tree is expanded up to certain depth.
    - If no feasible solution is found, the depth is increased and the entire process is repeated.
  - Memory complexity linear on the depth of the tree.
  - Suitable primarily for state-graphs that are trees.



Exploration Strategies

- Best-First Search
  - OPEN/CLOSED lists
  - A* algorithm
    - Heuristic estimate is used to order the nodes in the open list.
  - Large memory complexity.
    - Proportional to the number of states visited.
  - Suitable for state-space graphs that are either trees or graphs.

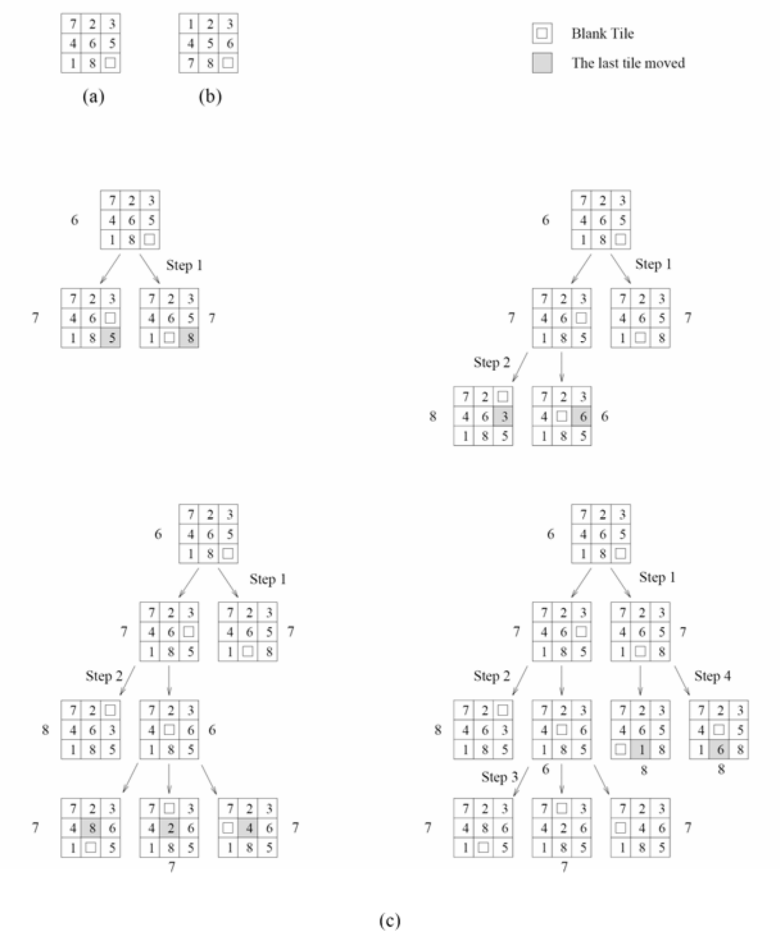


Figure 11.6 Applying best-first search to the 8-puzzle: (a) initial configuration; (b) final configuration; and (c) states resulting from the first four steps of best-first search. Each state is labeled with its h-value (that is, the Manhattan distance from the state to the final state).

Trees vs Graphs

- Exploring a graph as if it was a tree.
  - Can be a problem…

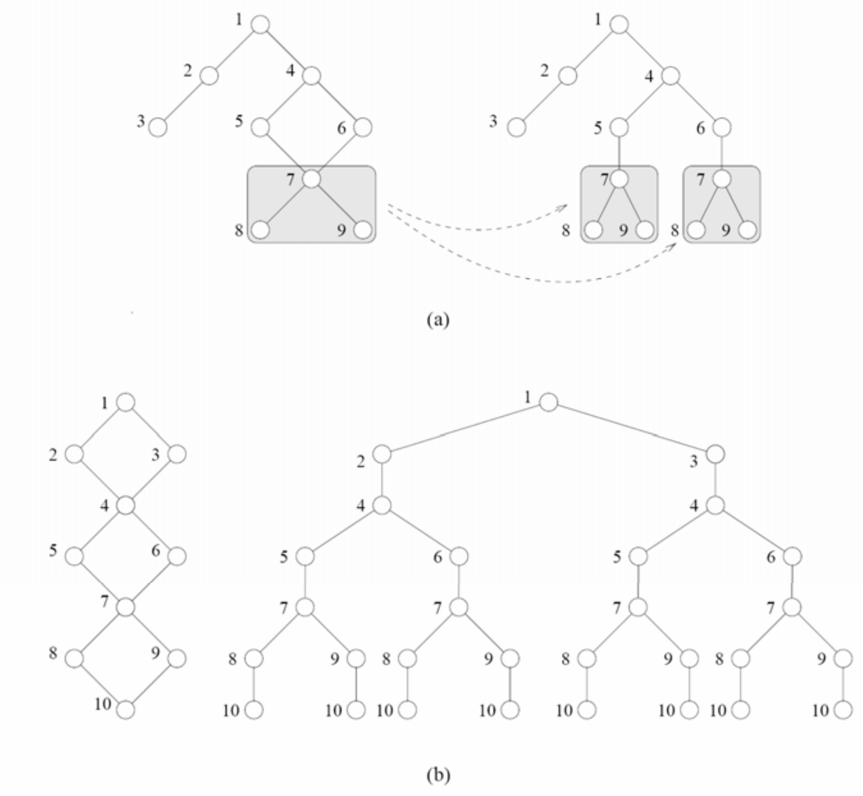
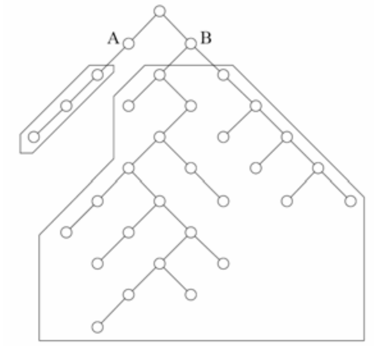
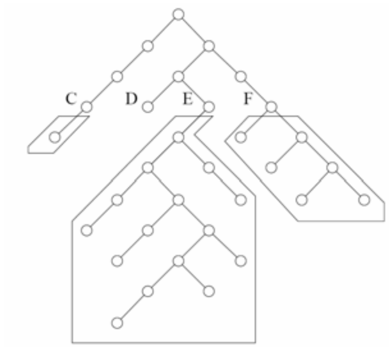


Figure 11.3 Two examples of unfolding a graph into a tree.

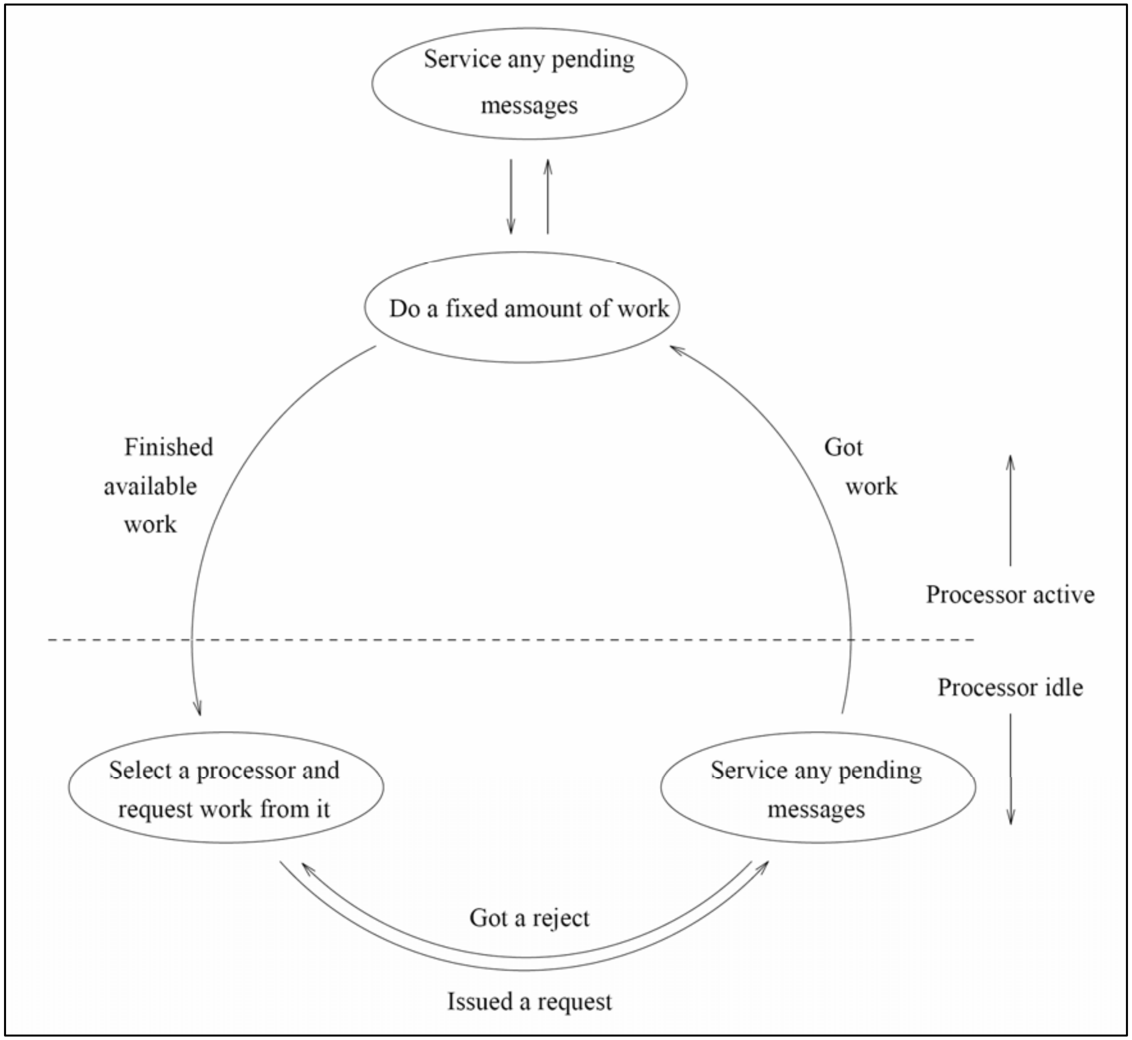
Parallel Depth-First Challenges

- Computation is dynamic and unstructured
  - Why dynamic?
  - Why unstructured?
- Decomposition approaches?
  - Do we do the same work as the sequential algorithm?
- Mapping approaches?
  - How do we ensure load balance?





Overall load-balancing strategy



Some more details

- Load balancing strategies
  - Which processor should I ask for work?
    - Global round-robin
    - Asynchronous (local) round-robin
    - Random
- Work splitting strategies
  - Which states from my stack should I give away?
    - top/bottom/one/many

Analysis

- How can we analyze these algorithms?
- Focus on worst-case complexity.
- Assumptions/Definitions:
  - a-splitting:
    - A work transfer request between two processors results in each processor having at least *aW* work for 0<a<=5 and *W* the original work available to one processor.
  - V(p) the number of work-transfer requests that are required to ensure that each processor has been requested for work at least once.
- Then…

$$T_o = t_{comm} V(p) \log W$$

Analysis

- Different load balancing schemes have different $V(p)$
  - Global round-robin: $V(p)=O(p)$.
  - Asynchronous round-robin: $V(p) = O(p^2)$
  - Random: $V(p) = O(p\log(p))$

Termination Detection

- How do we know that the total work has finished?
  - Dijkstra's algorithm
  - Tree-based termination

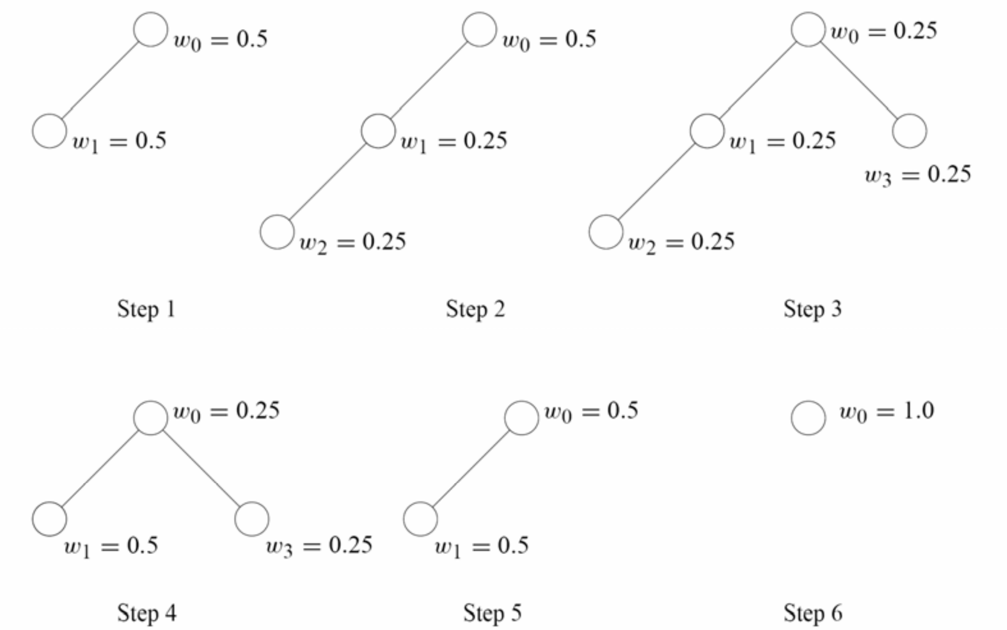


Figure 11.10 Tree-based termination detection. Steps 1–6 illustrate the weights at various processors after each work transfer.

Parallel Best-First Challenges

- Who maintains the Open & Closed lists
- How do you search a graph?

Open/Closed List Maintenance

- Centralized scheme
  - contention
- Distributed scheme
  - non-essential computations.
    - periodic information exchange.

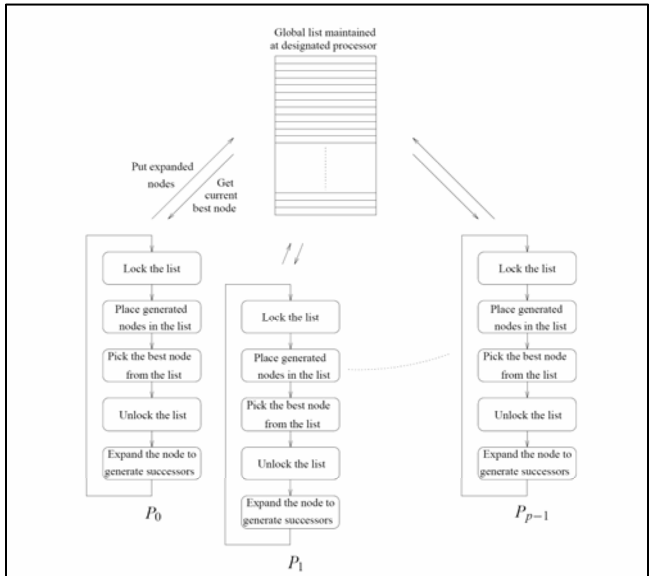


Figure 11.14 A general schematic for parallel best-first search using a centralized strategy. The locking operation is used here to serialize queue access by various processors.

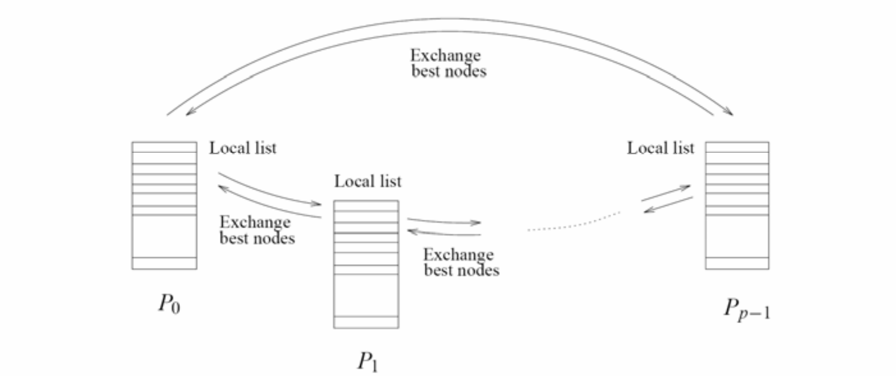


Figure 11.15 A message-passing implementation of parallel best-first search using the ring communication strategy.

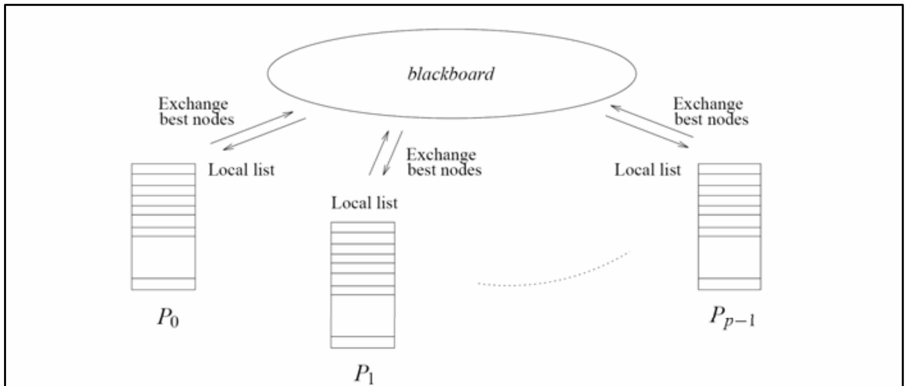


Figure 11.16 An implementation of parallel best-first search using the blackboard communication strategy.

Searching graphs

- Associate a processor with each individual node
  - Every time a node is generated is sent to this processor to check if it has been generated before.
    - Random hash-function that ensures load balancing.
  - High communication cost.

Speedup Anomalies

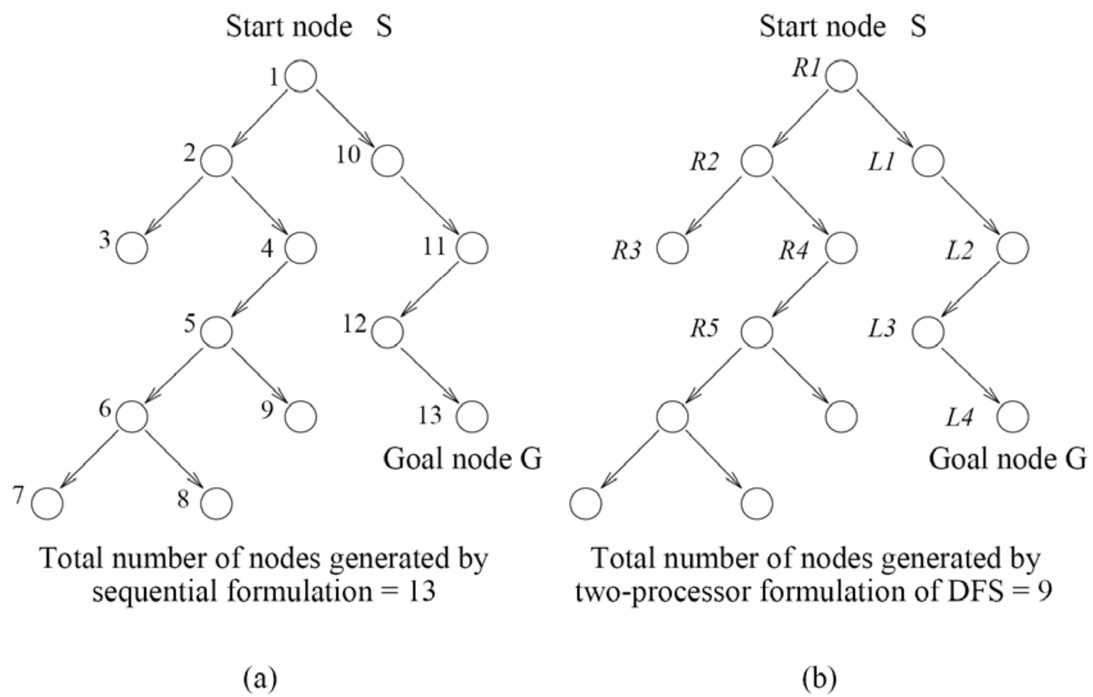


Figure 11.17 The difference in number of nodes searched by sequential and parallel formulations of DFS. For this example, parallel DFS reaches a goal node after searching fewer nodes than sequential DFS.