Principle of Parallel Algorithm Design

Alexandre David

B2-206

Today

- Preliminaries (3.1).
- Decomposition Techniques (3.2).
- Surprise.

Overview

- Introduction to parallel algorithms.
  - Tasks and decomposition.
  - Processes and mapping.
  - Processes vs. processors.
- Decomposition techniques.
  - Recursive decomposition.
  - Exploratory decomposition.
  - Hybrid decomposition.

Introduction

- Parallel algorithms have the added dimension of *concurrency*.
- Typical tasks:
  - Identify concurrent works.
  - Map them to processors.
  - Distribute inputs, outputs, and other data.
  - Manage shared resources.
  - Synchronize the processors.

Decomposing Problems

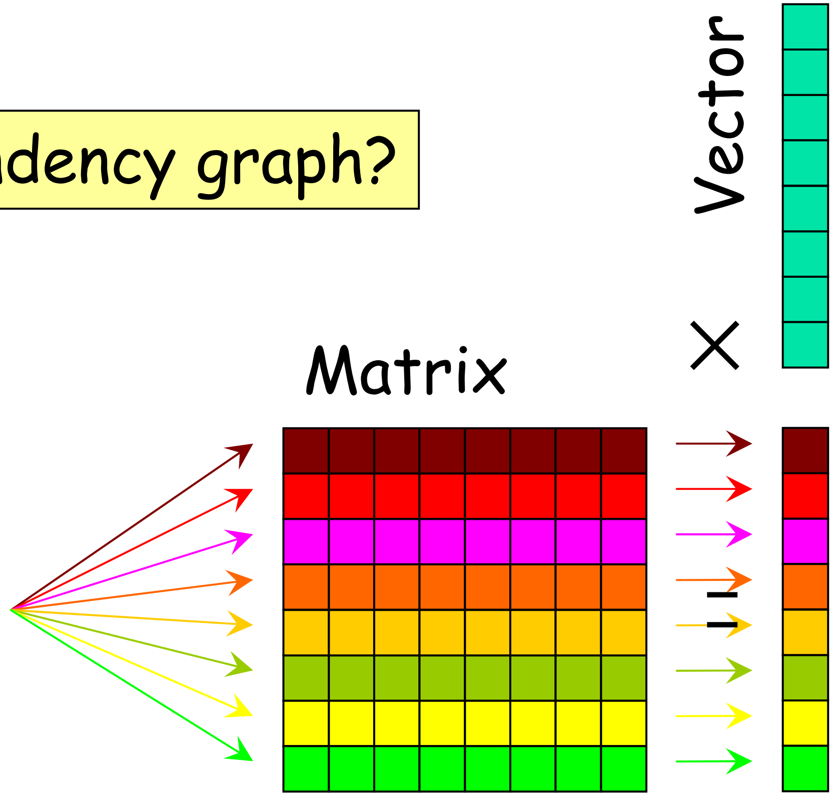
- Decomposition into *concurrent* tasks.
  - No unique solution.
  - Different sizes.
  - Decomposition illustrated as a directed graph:
    - Nodes = tasks.
    - Edges = dependency.

Task dependency graph

Example: Matrix * Vector

Task dependency graph?

N tasks, 1 task/row:



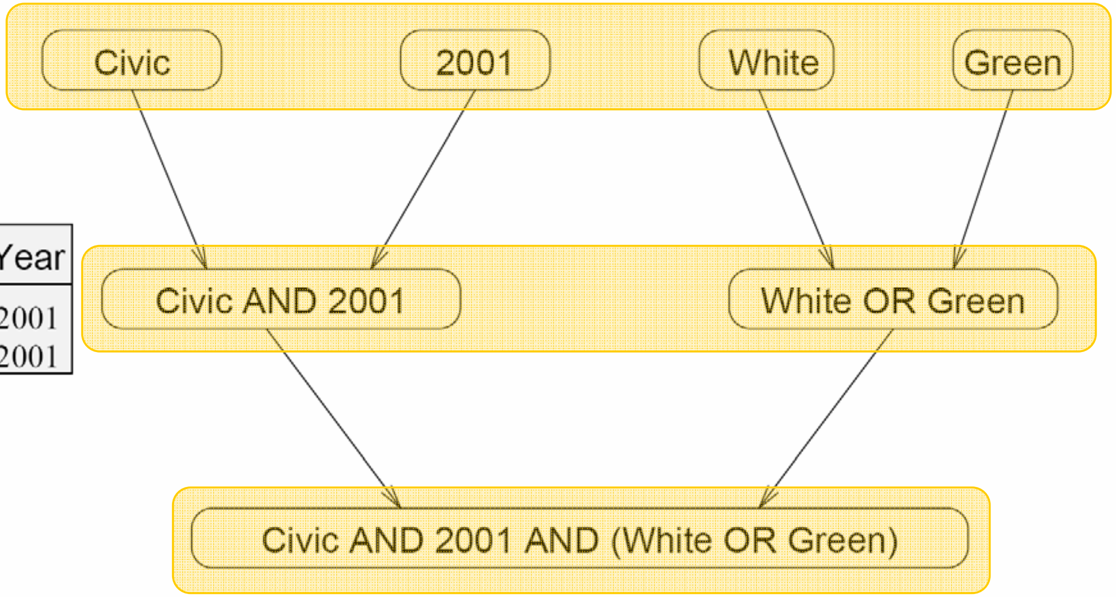
Example: Database Query Processing

MODEL = ``CIVIC'' AND YEAR = 2001 AND (COLOR = ``GREEN'' OR COLOR = ``WHITE)

ID#	Model	Year	Color	Dealer	Price
4523	Civic	2002	Blue	MN	$18,000
3476	Corolla	1999	White	IL	$15,000
7623	Camry	2001	Green	NY	$21,000
9834	Prius	2001	Green	CA	$18,000
6734	Civic	2001	White	OR	$17,000
5342	Altima	2001	Green	FL	$19,000
3845	Maxima	2001	Blue	NY	$22,000
8354	Accord	2000	Green	VT	$18,000
4395	Civic	2001	Red	CA	$17,000
7352	Civic	2002	Red	WA	$18,000

Table 3.1 A database storing information about used vehicles.

A Solution



ID#	Model
4523	Civic
6734	Civic
4395	Civic
7352	Civic

ID#	Year
7623	2001
6734	2001
5342	2001
3845	2001
4395	2001

ID#	Color
3476	White
6734	White

ID#	Color
7623	Green
9834	Green
5342	Green
8354	Green

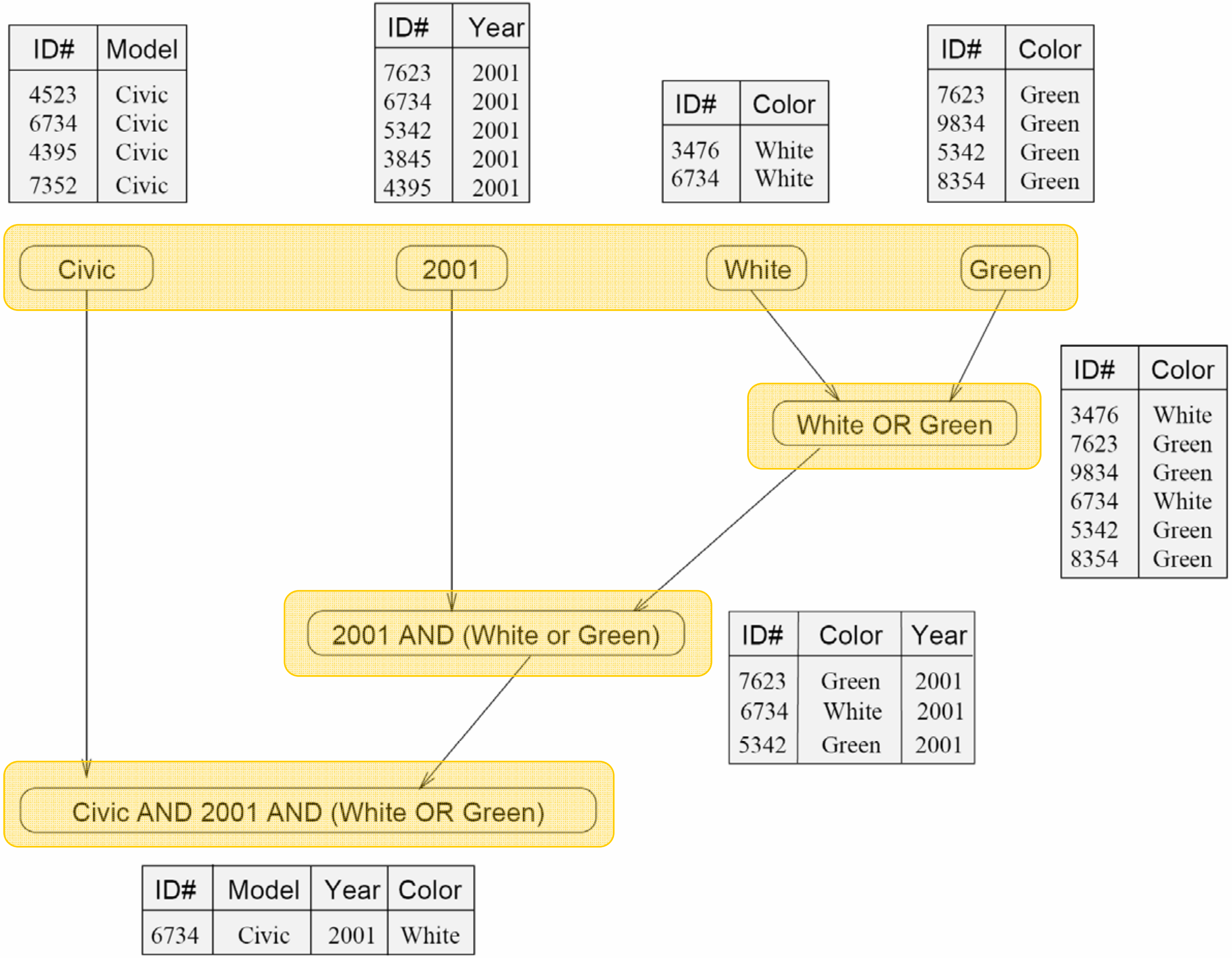
ID#	Model	Year
6734	Civic	2001
4395	Civic	2001

ID#	Color
3476	White
7623	Green
9834	Green
6734	White
5342	Green
8354	Green

ID#	Model	Year	Color
6734	Civic	2001	White

Figure 3.2 The different tables and their dependencies in a query processing operation.

Another Solution



ID#	Model
4523	Civic
6734	Civic
4395	Civic
7352	Civic

ID#	Year
7623	2001
6734	2001
5342	2001
3845	2001
4395	2001

ID#	Color
3476	White
6734	White

ID#	Color
7623	Green
9834	Green
5342	Green
8354	Green

ID#	Color
3476	White
7623	Green
9834	Green
6734	White
5342	Green
8354	Green

ID#	Color	Year
7623	Green	2001
6734	White	2001
5342	Green	2001

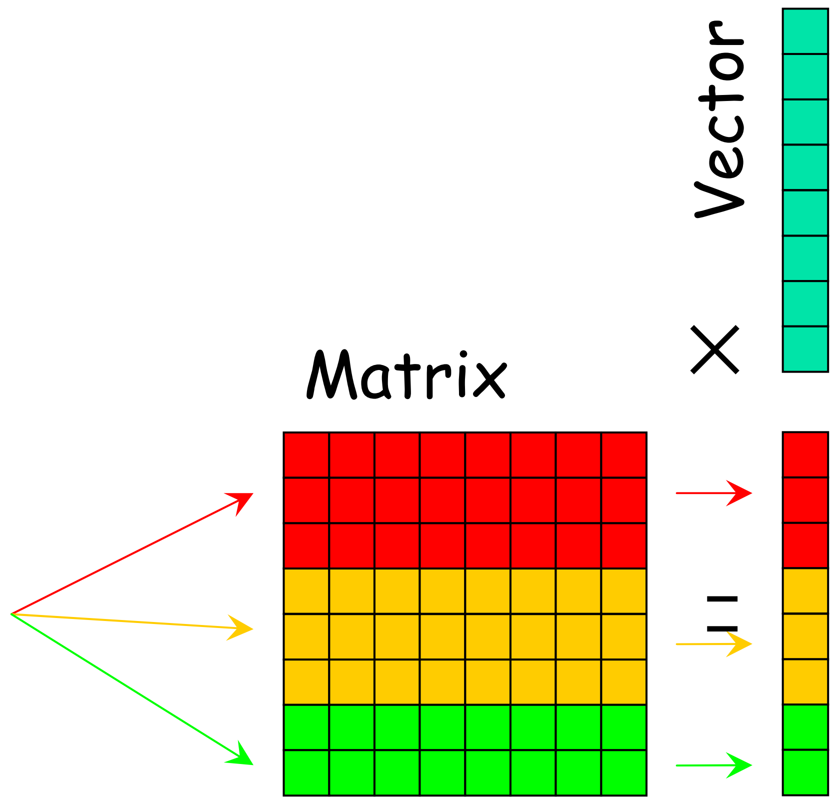
ID#	Model	Year	Color
6734	Civic	2001	White

Granularity

- Number and size of tasks.
  - Fine-grained: many small tasks.
  - Coarse-grained: few large tasks.
- Related: *degree of concurrency*.
  - Maximal degree of concurrency.
  - Average degree of concurrency.

Coarser Matrix * Vector

N tasks, 3 task/row:



Granularity

- Average degree of concurrency if we take into account varying *amount of work?*
- Critical path = longest directed path between any start & finish nodes.
- Critical path length = sum of the weights of nodes along this path.
- Average degree of concurrency = total amount of work / critical path length.

Database Example

Critical path (3).

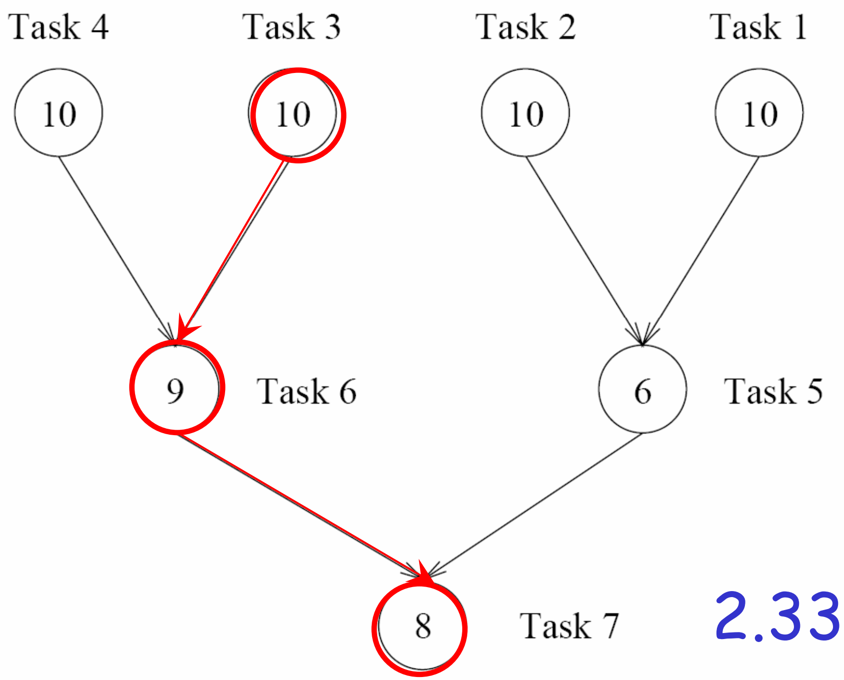
Critical path length = 27.

Av. deg. of concurrency = 63/27.

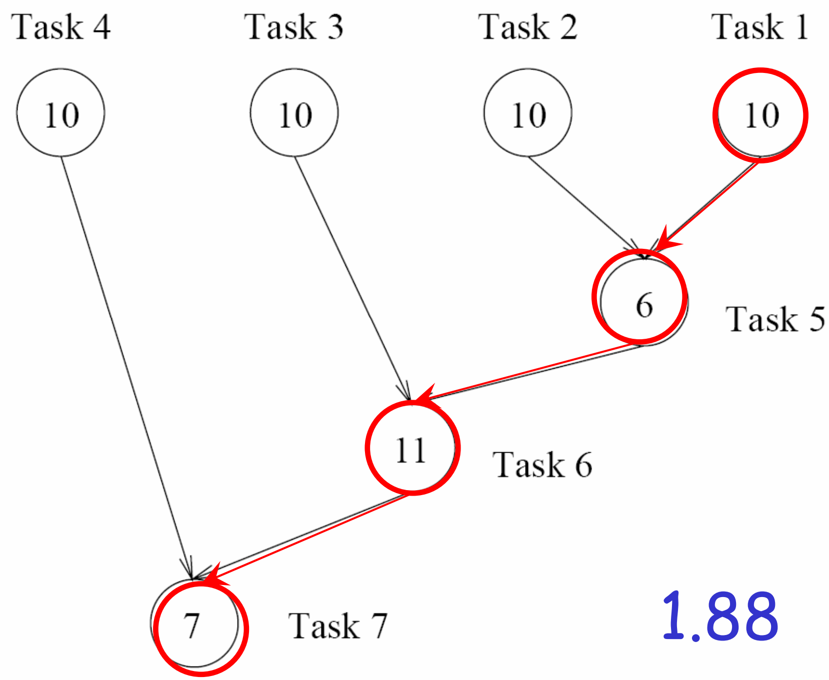
Critical path (4).

Critical path length = 34.

Av. deg. of conc. = 64/34.



2.33



1.88

Interaction Between Tasks

- Tasks often share data.
- Task interaction graph:
  - Nodes = tasks.
  - Edges = interaction.
  - Optional weights.
- Task dependency graph is a sub-graph of the task interaction graph.

Example: Sparse Matrix Multiplication

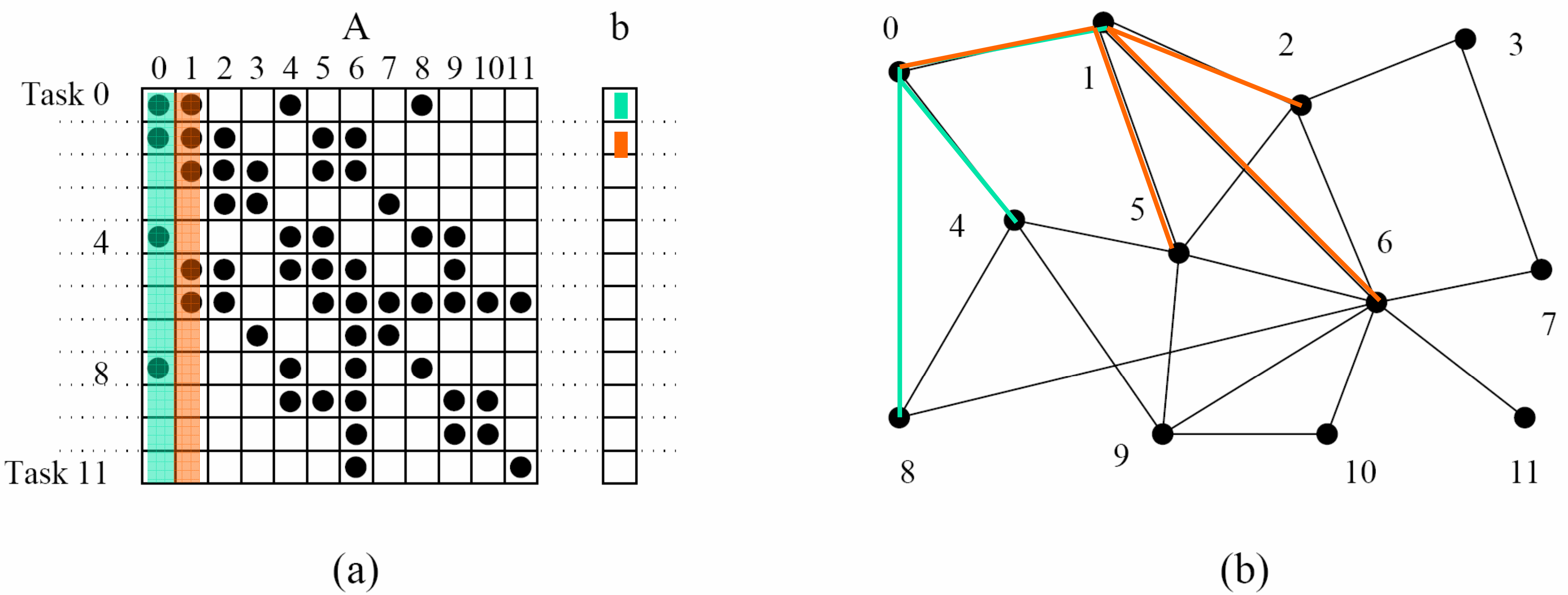


Figure 3.6 A decomposition for sparse matrix-vector multiplication and the corresponding task-interaction graph. In the decomposition Task i computes $\sum_{0 \le j \le 11, A[i,j] \ne 0} A[i, j].b[j]$.

Processes and Mapping

- Tasks run on processors.
- Process: processing agent executing the tasks. Not exactly like in your OS course.
- Mapping = assignment of tasks to processes.

- API expose processes and binding to processors not always controlled.

Mapping Example

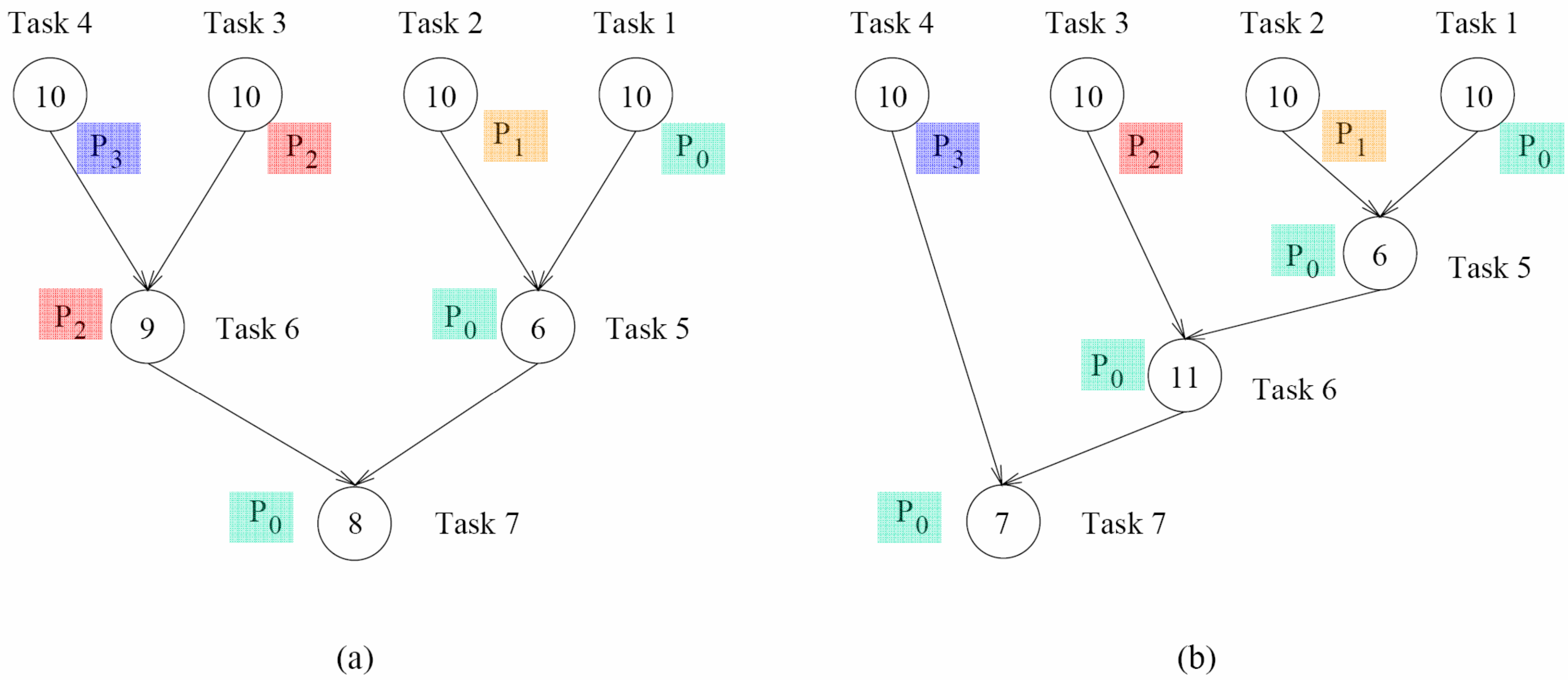


Figure 3.7 Mappings of the task graphs of Figure 3.5 onto four processes.

Processes vs. Processors

- Processes = logical computing agent.
- Processor = hardware computational unit.
- In general 1-1 correspondence but this model gives better abstraction.
- Useful for hardware supporting multiple programming paradigms.

Now remains the question:
How do you decompose?

Decomposition Techniques

- Recursive decomposition.
  - Divide-and-conquer.
- Data decomposition.
  - Large data structure.
- Exploratory decomposition.
  - Search algorithms.
- Speculative decomposition.
  - Dependent choices in computations.

Recursive Decomposition

- Problem solvable by divide-and-conquer:
  - Decompose into sub-problems.
    - Do it recursively.
  - Combine the sub-solutions.
    - Do it recursively.
- Concurrency: The sub-problems are solved in parallel.

Quicksort Example

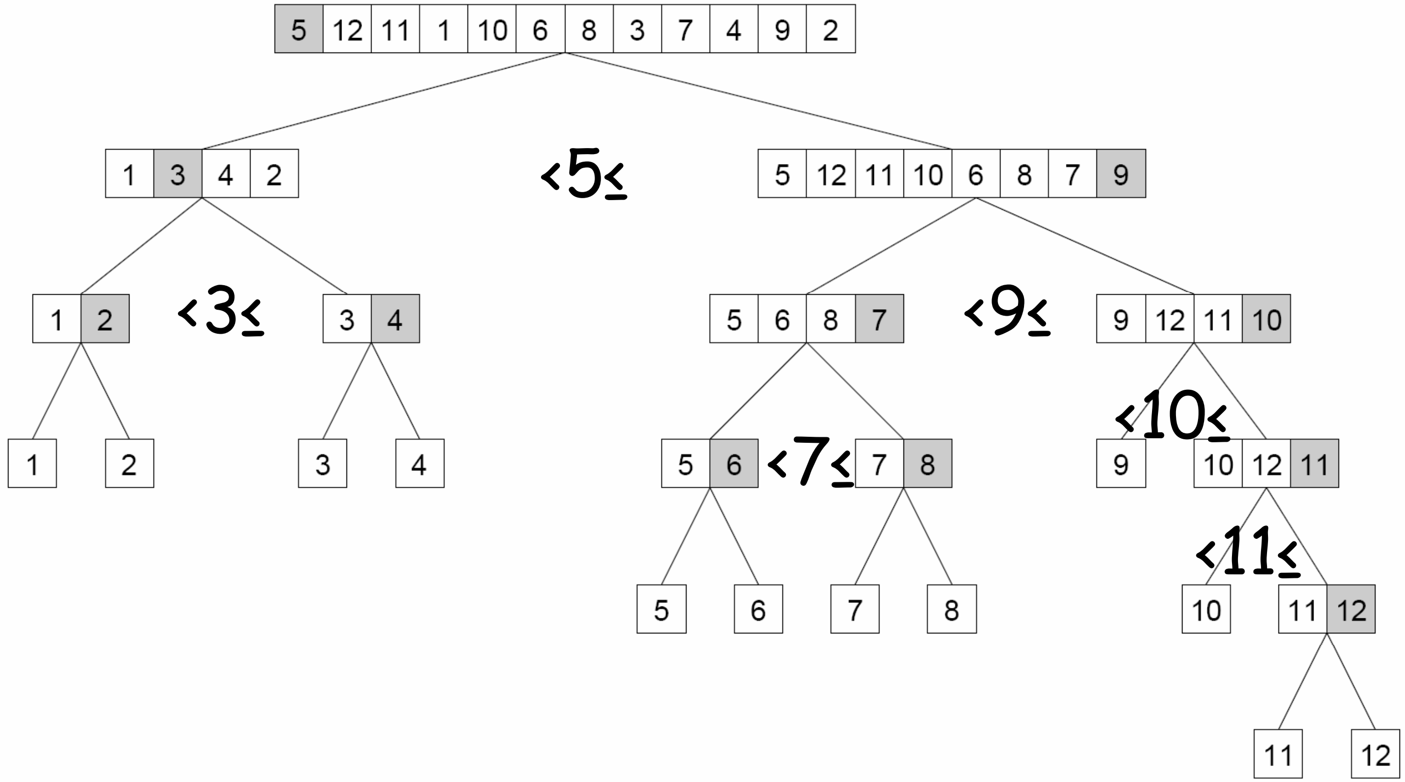


Figure 3.8 The quicksort task-dependency graph based on recursive decomposition for sorting a sequence of 12 numbers.

Minimal Number

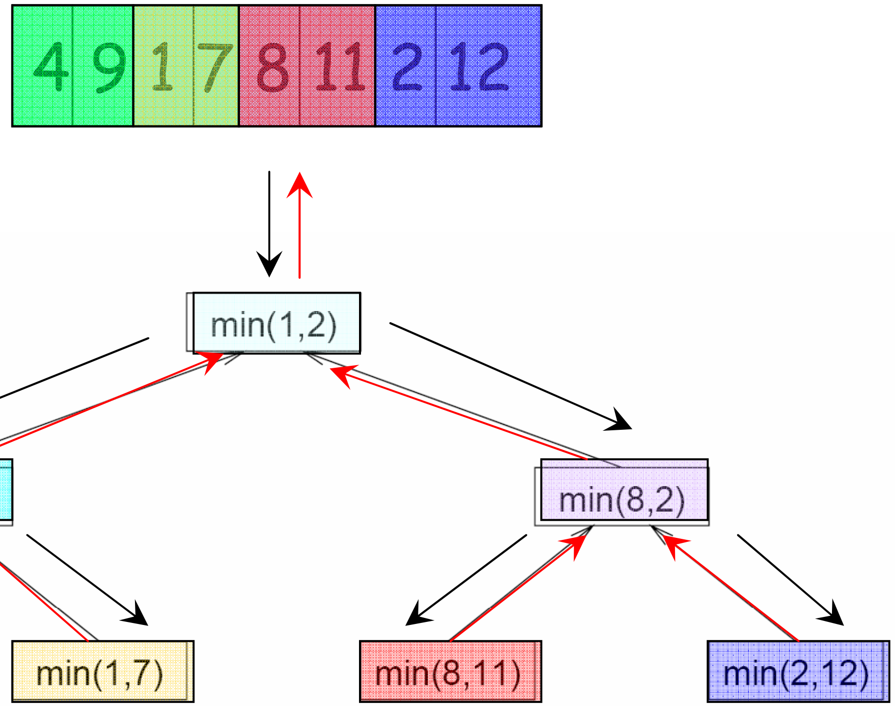


Figure 3.9 The task-dependency graph for finding the minimum number in the sequence {4, 9, 1, 7, 8, 11, 2, 12}. Each node in the tree represents the task of finding the minimum of a pair of numbers.

Data Decomposition

- 2 steps:
  - Partition the data.
  - Induce partition into tasks.
- How to partition data?
- Partition output data:
  - Independent “sub-outputs”.
- Partition input data:
  - Local computations, followed by combination.

Matrix Multiplication

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \rightarrow \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

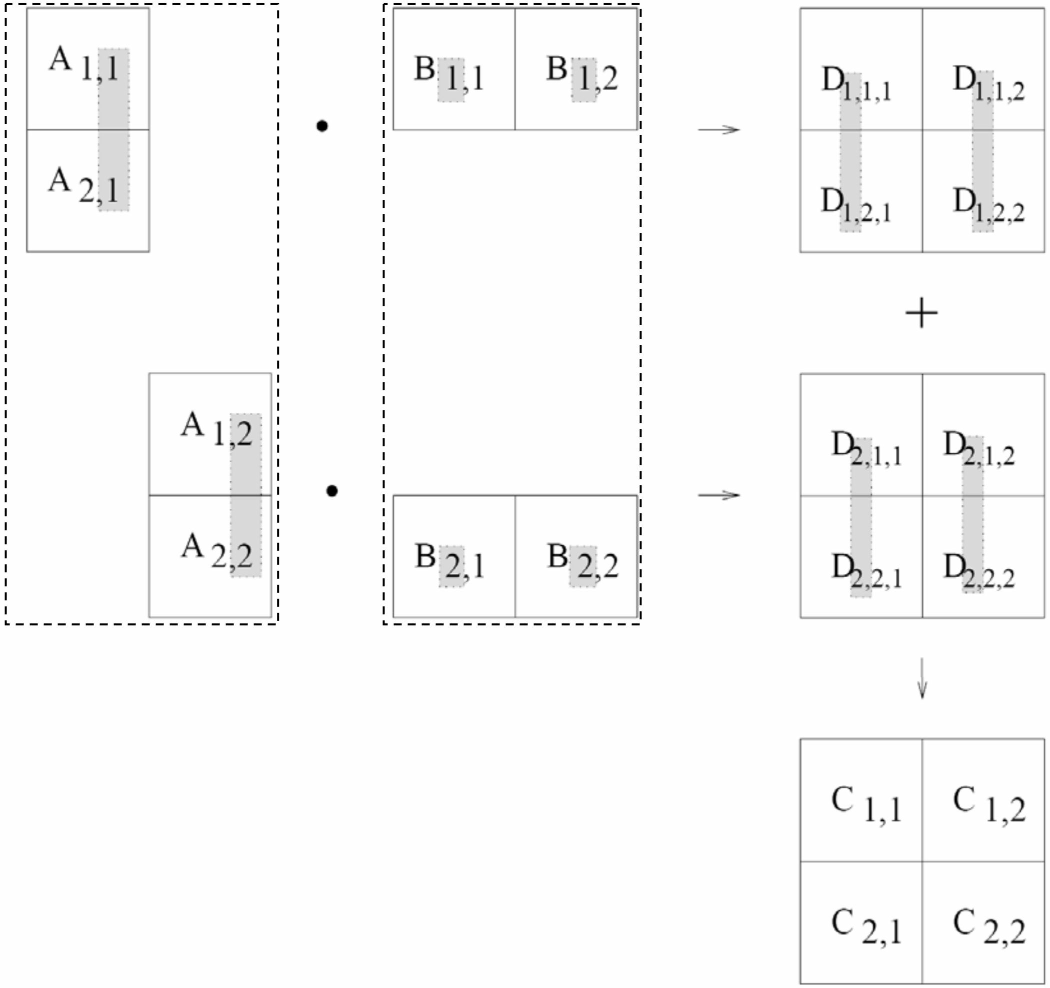
(a)

$$\begin{array}{ll} \text{Task 1:} & C_{1,1} = A_{1,1}B_{1,1} + A_{1,2}B_{2,1} \\ \text{Task 2:} & C_{1,2} = A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ \text{Task 3:} & C_{2,1} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1} \\ \text{Task 4:} & C_{2,2} = A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{array}$$

(b)

Figure 3.10 (a) Partitioning of input and output matrices into 2×2 submatrices. (b) A decomposition of matrix multiplication into four tasks based on the partitioning of the matrices in (a).

Intermediate Data Partitioning



Linear combination of the intermediate results.

Owner Compute Rule

- Process assigned to some data
  - is responsible for all computations associated with it.
- Input data decomposition:
  - All computations done on the (partitioned) input data are done by the process.
- Output data decomposition:
  - All computations for the (partitioned) output data are done by the process.

Exploratory Decomposition

15-puzzle example

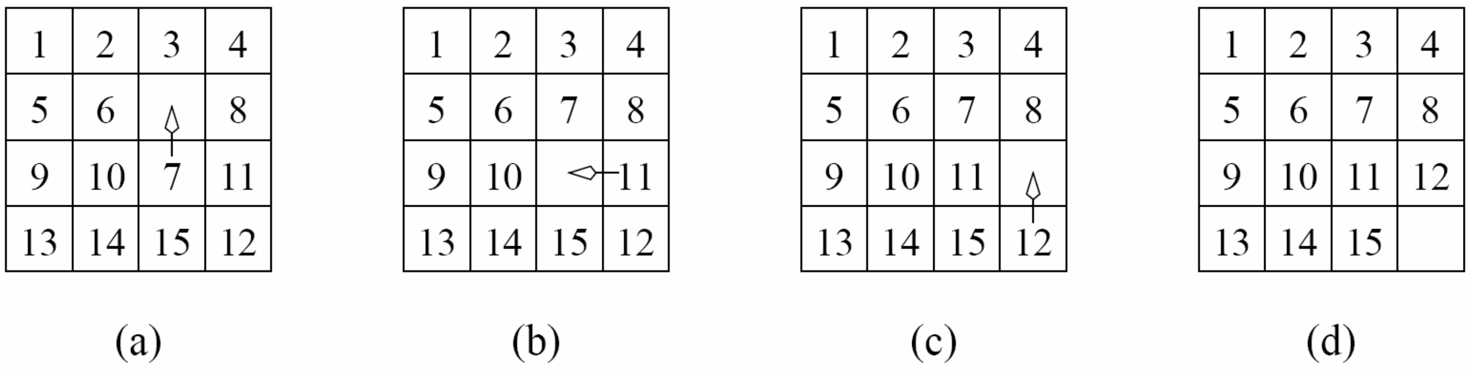
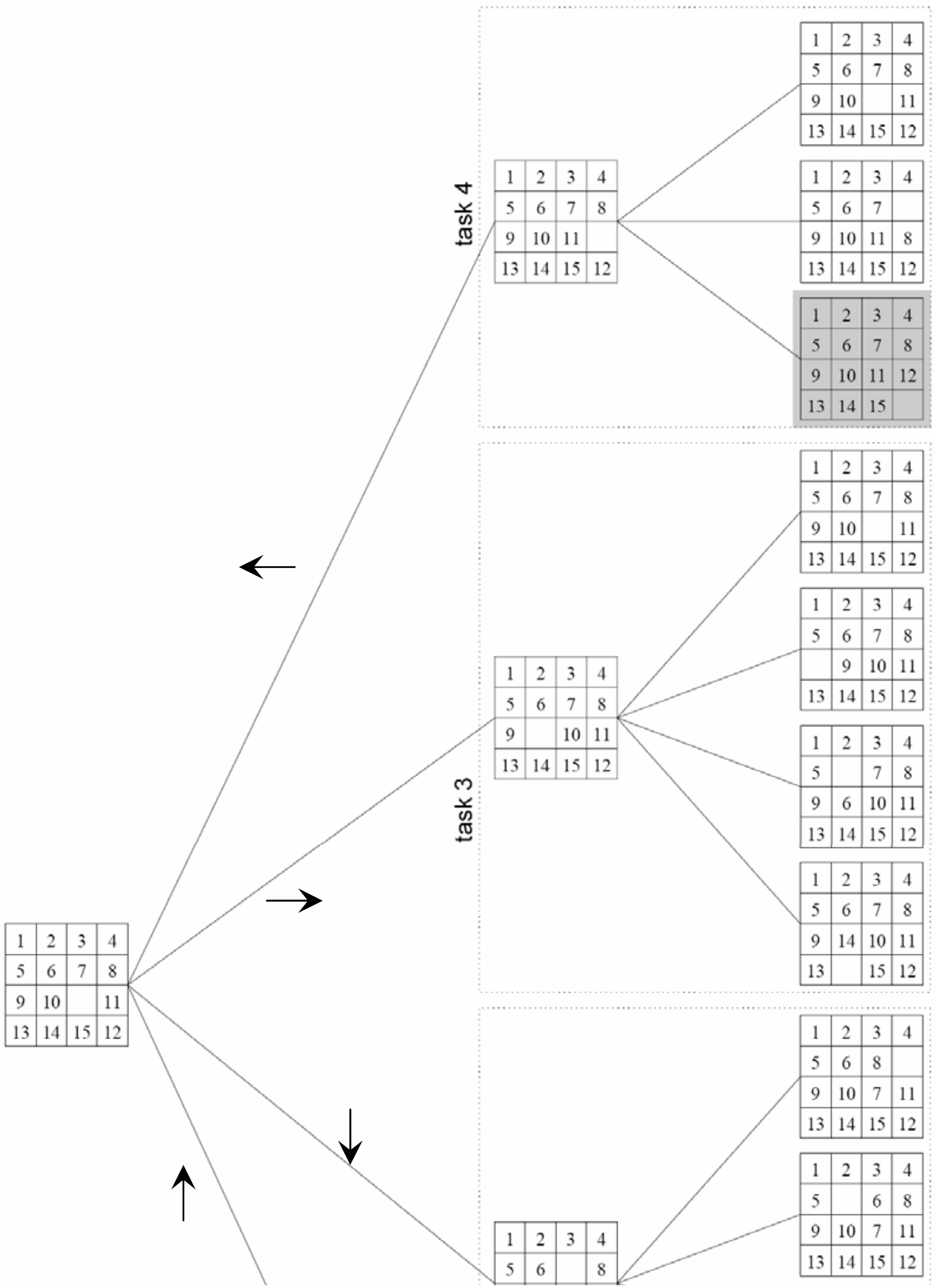


Figure 3.17 A 15-puzzle problem instance showing the initial configuration (a), the final configuration (d), and a sequence of moves leading from the initial to the final configuration.

Search



Anomalous Behavior Possible

Work depends on the order of the search!

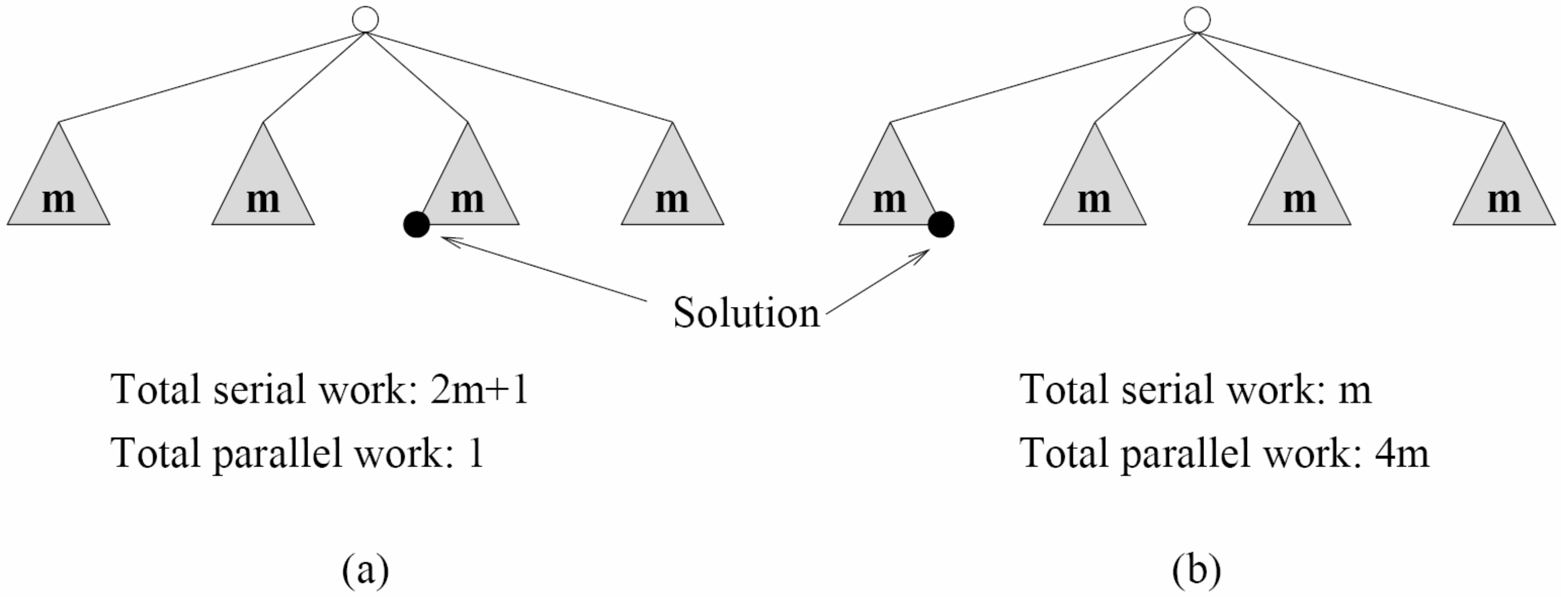


Figure 3.19 An illustration of anomalous speedups resulting from exploratory decomposition.

Speculative Decomposition

- Dependencies between tasks are not known a-priori.
  - How to identify independent tasks?
  - Conservative approach: identify tasks that are *guaranteed* to be independent.
  - Optimistic approach: schedule tasks even if we are not sure – may roll-back later.

Speculative Decomposition Example

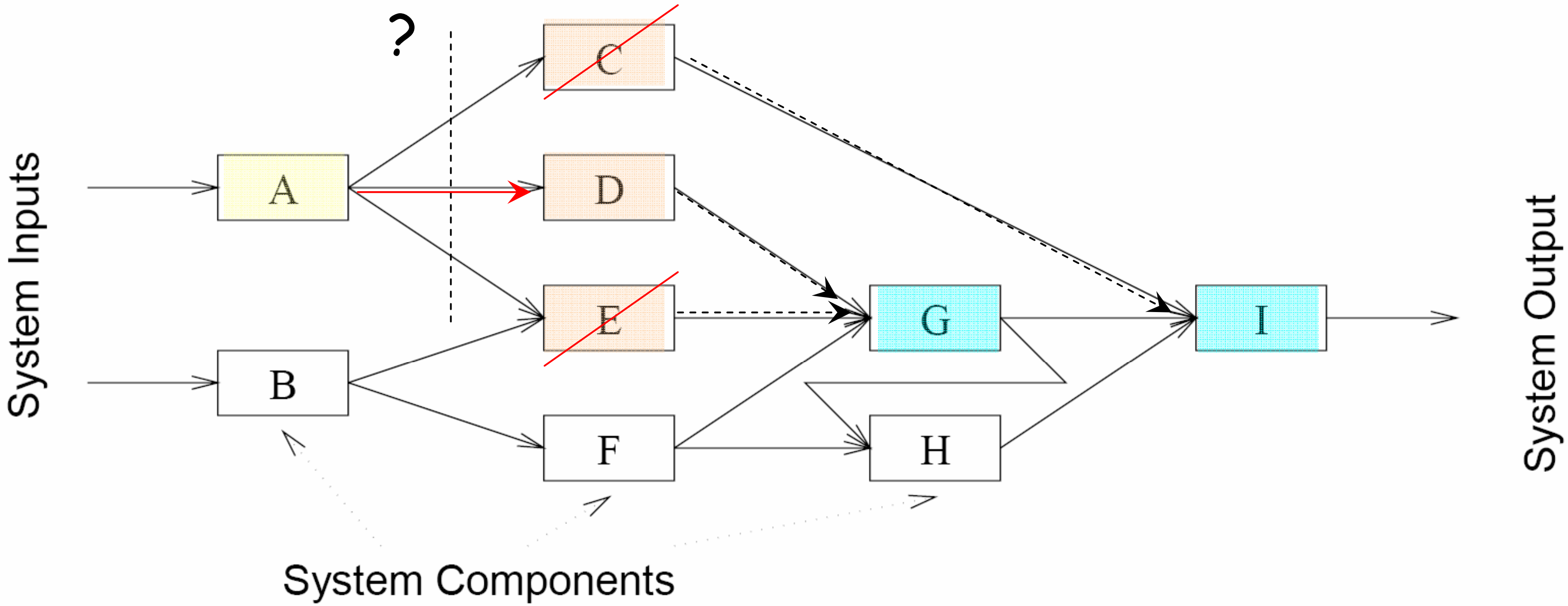


Figure 3.20 A simple network for discrete event simulation.