Parallel Graph Algorithms

1. Introduction

Parallel processing is commonly used to solve computationally large and data-intensive tasks on a number of computational nodes.

the main goal in using this method is to obtain results much faster than would be acquired by sequential processing and hence improve the system performance.

2. Concepts and Terminology (概念和术语)

1.parallelism versus concurrency:

Parallelism indicates at least two tasks are running physically at the same time on different processors. Concurrency is achieved when two or more tasks run at the same time frame but they may not run at the same physical time. Concurrency is more general than parallelism and encompasses parallelism.

2.fine-grain versus coarse-grain parallelism

In fine-grain parallelism,tasks communicate and synchronize frequently and coarse-grain parallelism involves tasks with larger computation times that communicate and synchronize much less frequently

3. embarrassingly parallel:

The parallel computation consists of independent tasks that have no inter-dependencies in this mode

there are no precedence relationships or data communications among them.

4.multi-core computing

A multi-core processor contains more than one of the processing elements called cores. Most contemporary processors are multi-core and multi-core computing is running on programs in parallel on multi-core processors.

5. symmetric multiprocessing:

A symmetric multiprocessor contains a number of identical processors that communicate via a shared memory .Note that SMPs are organized on a coarser scale than multi-core processors which contain cores in a single integrated circuit package

6. multiprocessors:

A multi-processor consists of a number of processors which communicate through a shared memory.

7. multicomputer:

Each processor has private memory and typically communicates with other microcomputers by sending and receiving messages. There is no global memory in general.

8. cluster computing

A cluster is a set of connected computers that communicate and synchronize using messages over a network to finish a common task. A cluster is envisioned as a single computer by the user and it acts as a single computer.

9. Grid computing

A grid is a large number of geographically distributed computers that work and cooperate to achieve a common goal.

10. Could computing

Could computing enables sharing of networked computing resources

11.parallel processing with GPUs

3.Parallel Architectures

1. Shared Memory Architectures

each processor has some local memory,and inter process communication and synchronization are performed using a shared memory that provides access to all processors. Data is read from and written to the shared memory locations;

we need to provide some form of control on access to this memory to prevent race conditions.

advantage

fast data access to memory.

disadvantages

the shared memory should be protected against concurrent accesses by the parallel tasks and this process should be controlled by the programmer in many case.

due to the bottleneck over the bus while accessing the shared memory. the number of processors is limited.

2.Distributed Memory Architectures

the communication and synchronization are performed exclusively by sending and receiving of message.

these system are also called **message passing systems**

in these systems, each processor has its own local memory and address space is not shared among processes.

access to local memory is faster than global memory access.

advantage

scalability and use of the off-the-self computers

no overheads in memory management as in shared memory

disadvantage

the task of the algorithm designer to manage data communication using messages and the communication over the network is commonly serial which is slower than the parallel communication of the shared memory system.

3.Flynn's Taxonomy

- Single-instruction Single-data (SISD)

  execute one instruction per unit time on one data item.

- Single-instruction Multiple-data (SIMD)

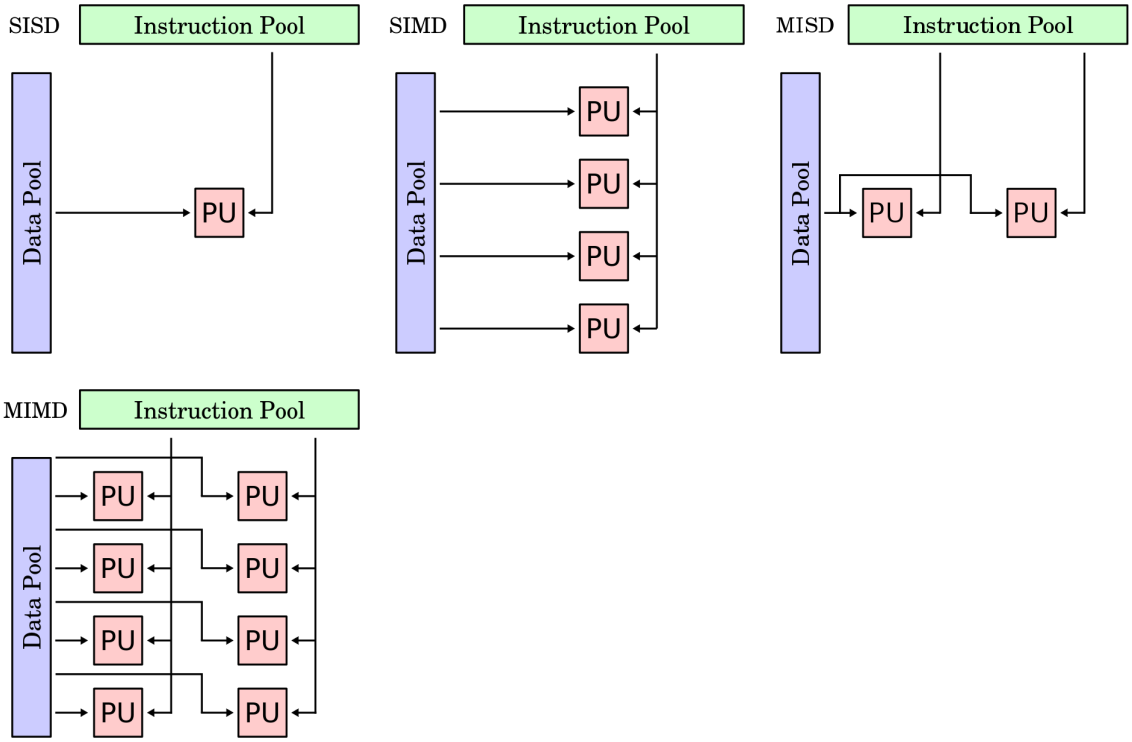
  we have synchronous parallel processors executing the same instruction on different data in this architecture.

- Multiple-instruction Single-data (MISD)

  execute different instruction on the same data

- Multiple-instruction Multiple-data (MIMD)

  we have processors running different instructions on different data in this case.



Models

1.The parallel random access memory (PRAM) Model

The processor is identical with some local memory, and there is a global shared memory used for communication and synchronization.

Processors work synchronously using a global clock and at each time unit, a processor can perform a read from a global or local memory location; execute a single RAM operation and write to one global or local memory location.

PRAM models are classified according to the read or write access rights to the global memory as follows:

- The exclusive-read-exclusive-write (EREW)

  this model required each processor to read from or to write to global memory locations exclusively, one processor at a time.

- The concurrent-read-exclusive-write (CREW)

  this model allows concurrent reading of the same global memory location by more than one processor;

  writing to a global memory location has to be done exclusively.

- The concurrent-read-concurrent-write (CRCW)

  this model allows both concurrent reads and concurrent writes

  in case of concurrent writes,a mechanism is needed to decide what is to be written as last to the memory location.

the PARM model is idealistic as it assumes unbounded number of processors and unbounded size of shared memory. However,it simplifies parallel algorithm design greatly by abstracting communication and synchronization details.

2. Message Passing Model

Message Passing model is based on the distributed memory architecture. There is no shared memory, and the parallel tasks that run on different processors communicate and synchronize using two basic primitives: send and receive.

同步/异步

同步操作：当一个操作发出之后，再没有得到结果之前，调用就不返回。

异步操作：当一个操作发出之后，调用者不能立刻得到结果，世界处理的组件完成之后通过状态，通知，和回调来通知调用者。

阻塞/非阻塞

阻塞：再调用结果返回之前，调用方会被挂起，直至调用返回

非阻塞：不能立即得到结果之前，该函数不会阻塞当前线程，会立刻返回。

5.Analysis of Parallel Algorithms

How to evaluate a parallel algorithm is good or bad?

- the running time of an algorithm
- the number of processors it uses
- its cost

the running time of a parallel algorithm T_p can be specified as $T_p = t_{fin} - t_{st}$

where t_{st} is the start time of the algorithm on the first processor and t_{fin} is the finishing time of the algorithm in the last processor. The Depth D_p of a parallel algorithm is the largest number of dependent steps performed by the algorithm. The dependency in this context means a step cannot be performed before a previous one finishes since it needs the output from this previous step.

If T_P is the worst-case running time of a particular algorithm A for a problem Q using p identical processors and T_s is the worst-case running time of the fastest known sequential algorithm to solve Q, the *speedup* S_p is defined as below

$S_p = \frac{T_s}{T_p}$

Efficiency of a parallel algorithm is defined as

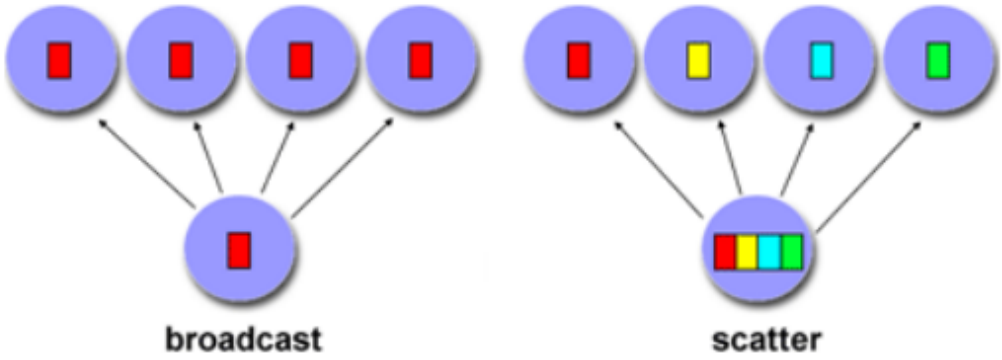
$E_p = \frac{S_p}{p}$

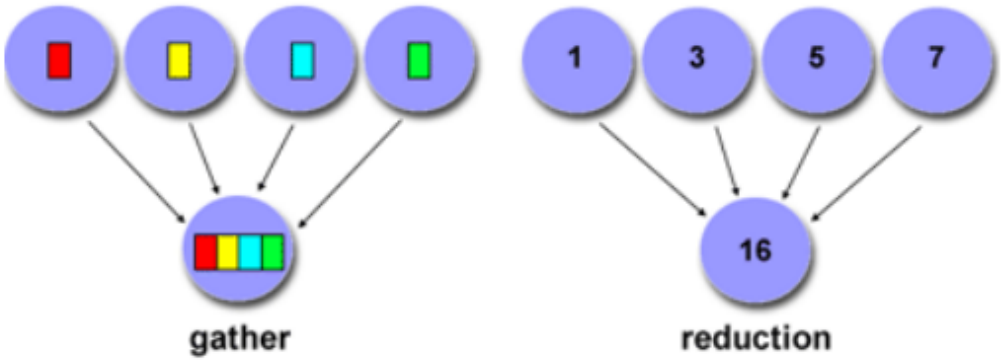
A parallel algorithm is said to be scalable if its efficiency remains almost constant when both the number of processors and the size of the problem are increased.

6. Basic Communication Modes

Processes residing on different computing nodes of the parallel system need to communicate to finish an overall task.

- One-to-all Broadcast: A process p_i broadcast a message *m* to all *n*-1 other processes
- All-to-one Reduction: The data of each process of the parallel system is combined using an associative operator and the result is stored in a single memory location of a specific process.
- All-to-all Broadcast and All-to-all Reduction: We have each process p_i , $0 \leq i < n$ of n processes sending message m_i to all other processes simultaneously. Each process stores messages m_i $0 \leq i < n$ at the end of this communication. In the all-to-all reduction mode is the reverse operation of all-to-all broadcast in which each process p_i stores n distinct messages sent by other processes in the end.
- Scatter and Gather Operation: A personalized unique message for each process is sent from a process in the scatter operation. the source process p_i has message m_j $0 \leq j < n$,$j \neq i$ for each of the processes and each process p_j $j \neq i$,receives its message m_j in the end. The reverse procedure is performed in the gather operation with each process sending its unique message to be gathered at one specific process.
- All-to-all Personalized Communications: In this mode,each process p_i has a distinct message m_i and sends this message to all other processes. All processes in the system receive messages m_i $0 \leq i < n$ at the end of this communication.



- 

7. Parallel Algorithm Design Methods

We can employ one of the following strategies when designing a parallel algorithm: modify an existing sequential algorithm by detecting subtasks that can be performed in parallel which is by far one of the most commonly used approaches

1. Partitioning: Data,the overall task or both,can be partitioned into a number of processors. Partitioning of data is called *data* or *domain decomposition* and partitioning of code is termed *functional decomposition*
2. Communication: The amount of data and the sending and receiving parallel subtasks are determined in this step.

3. Agglomeration(集聚): The subtasks determined in the first two steps are arranged into larger groups with the aim of reducing communication among them.
4. Mapping: The formed groups are allocated to the processors if the parallel system. When the task graph that depicts subtasks and their communication is constructed.

1. Data Parallelism

Data parallelism or *data decomposition* method comprises simultaneous execution of the same function on the elements of a data set. Dividing data to *k* processors $p_1, \ldots, p_k$ each of which working on its partition can be applied to both PRAM and message passing models.

Data is decomposed onto processors

processors performance similar tasks on data

2. Functional Parallelism

Functional parallelism or *functional decomposition* approach involves applying different operations to possibly different data simultaneously. Also called *task parallelism*, this model involves simultaneous execution of various functions on multiple processors on the same or different data sets.

8. Parallel Algorithm Methods for Graphs

Partitioning of data for parallel graph algorithm means balanced partitioning of graph among the processors which is an NP-hard problem. We need to radically different methods for parallel graph algorithm, and graph contraction,pointer jumping,and randomization are the three fundamental approaches for this purpose.

1. Randomization and Symmetry Breaking

Symmetry breaking in a parallel graph algorithm involves selection of a subset from a large set of independent operations using some property of the graph.

A randomized algorithm makes certain decisions based on the result of coin flips during the execution of the algorithm，These algorithms assume any input combination is possible. Two main classes of randomized algorithms are Las Vegas and Monte Carlo algorithms

symmetry breaking may be employed to correct the output when independent parallel operations on the vertices or edges of a graph produce a large and possibly incorrect result.

2. Graph Partitioning

graph partitioning task is dividing the vertex set V into disjoint vertex sets $V_1, \ldots, V_k$ such that the number of vertices in each partition is approximately equal and the number of edges between the subgraphs induced by the vertices in the partition is minimal

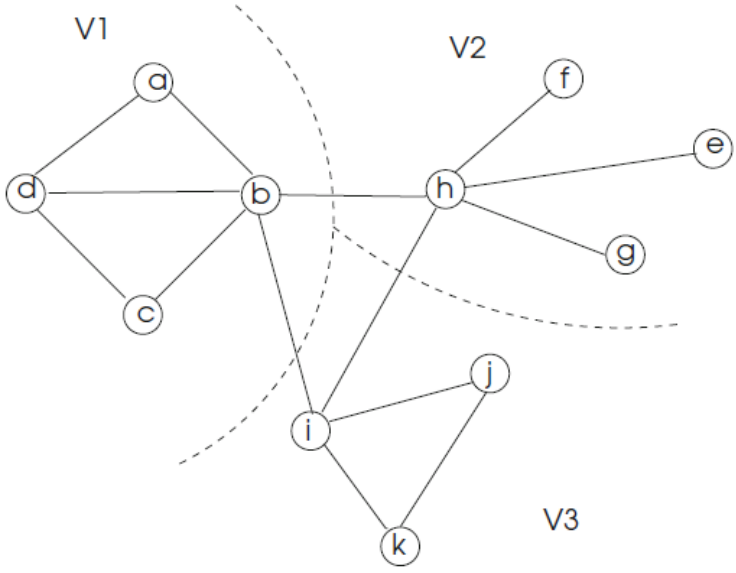


Fig. 4.8 Partitioning of a sample graph into three disjoint vertex sets; $V_1 = \{a, b, c, d\}$, $V_2 = \{f, g, h, e\}$ and $V_3 = \{j, k, l\}$. A number of vertices in these partitions are 4, 4, and 3, respectively, and there is a total of three edges between the partitions

3. Graph Contraction

Graph contraction method involves obtaining smaller graphs by shrinking the original graph at each step.

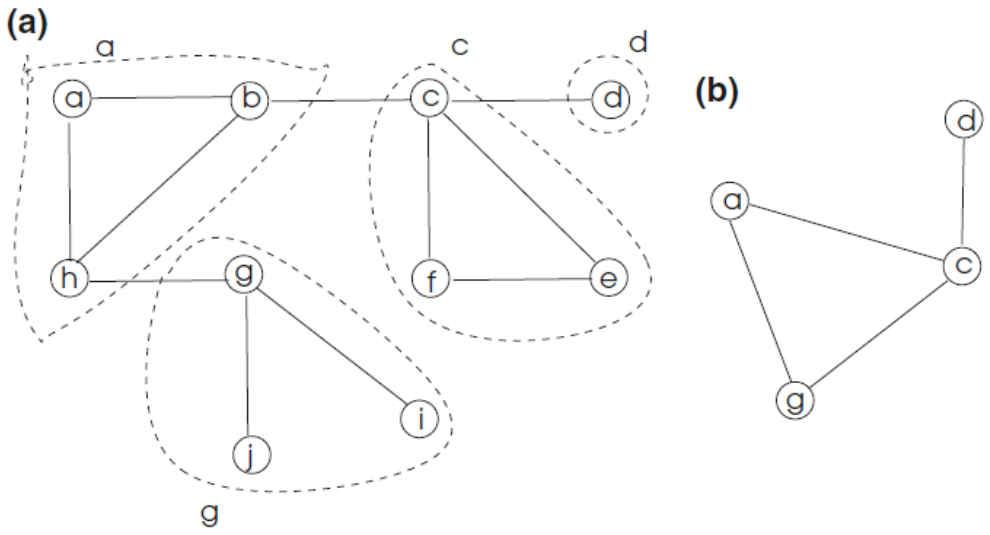
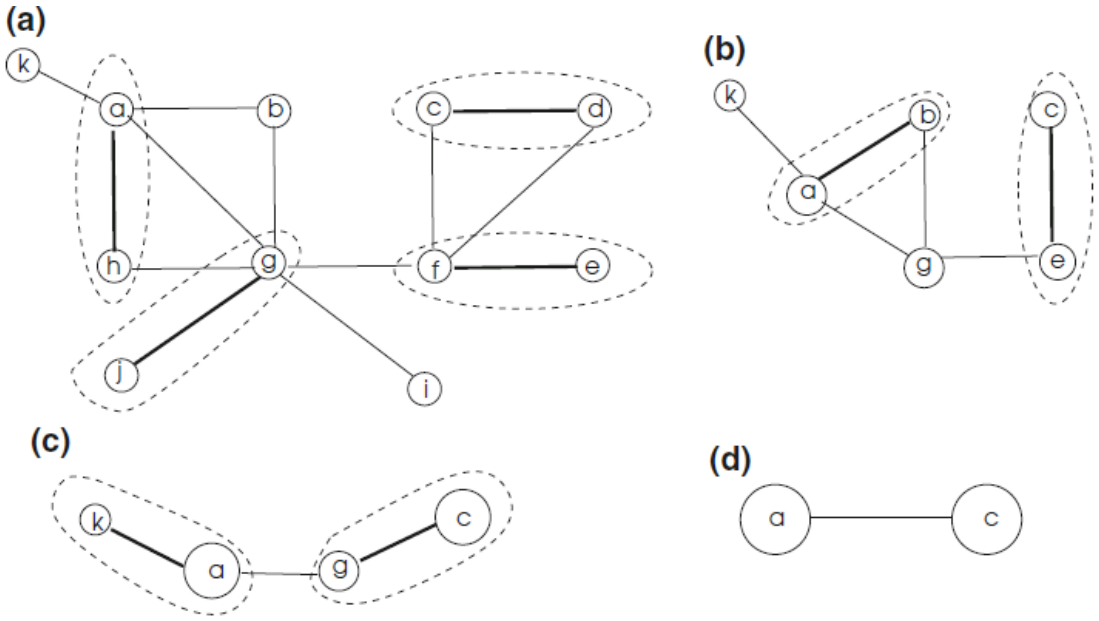


Fig. 4.9 Contraction of a sample graph. The vertex set in **a** is partitioned into four subsets and each subset is represented by one of the vertices in the subset to get the contracted graph in **b**

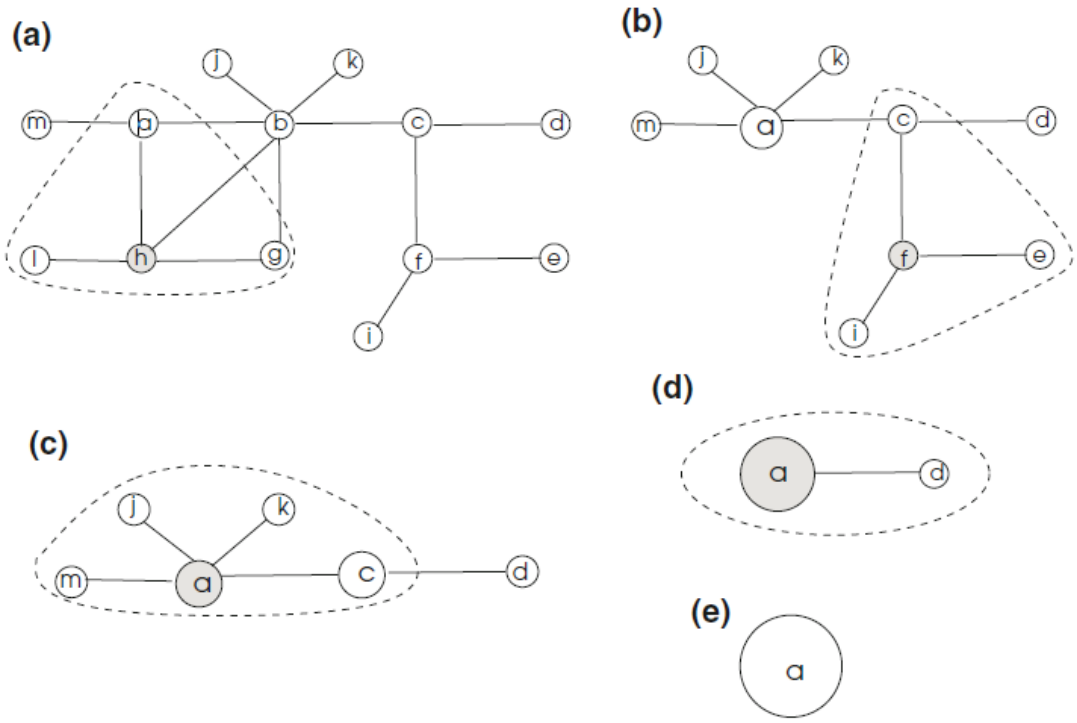
Edge Contraction

The edge partitioning of a graph involves selecting distinct edges or isolated vertices which will form the partitions.We then contract vertices pairwise that are incident to these selected edges in this method. Since two vertices and the edge between them are contracted to have a single vertex,



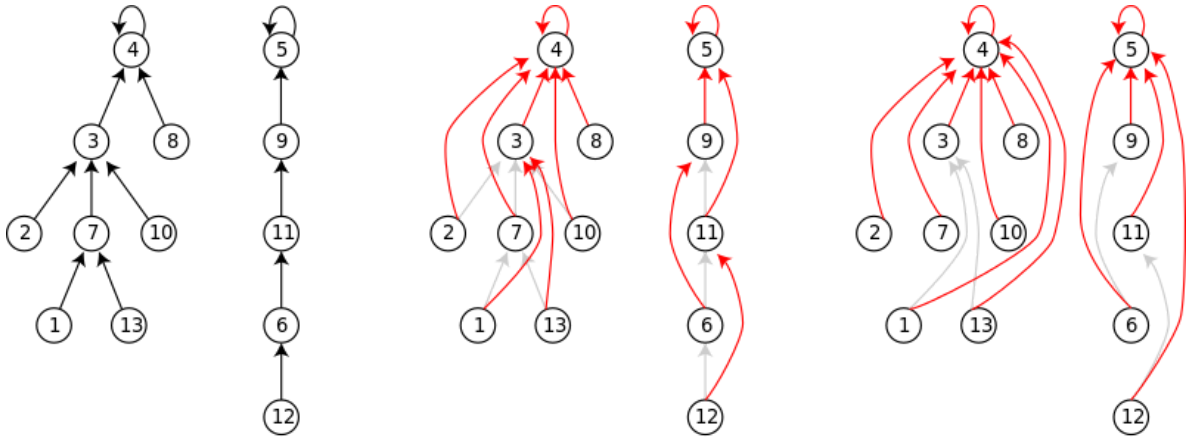
Star Contraction

In star contraction, we select a vertex in each component as the center of the star and all other vertices connected directly to this center called satellites are contracted to form a supervertex.



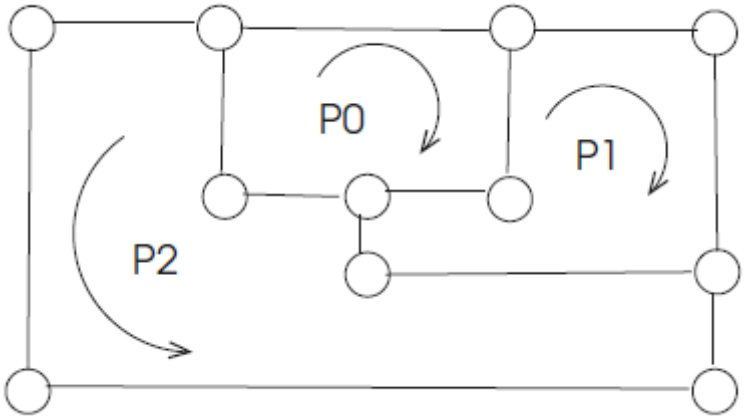
Pointer jumping

Pointer jumping or path doubling is a design technique for parallel algorithms that operate on pointer structures such as linked list and directed graphs. It can be used to find the roots of a forest of rooted trees



ear decomposition

An ear decomposition of a graph G = (V, E) is the union of paths P0, P1, ..., Pn with P0 being a simple cycle and Pi (1 ≤ i ≤ n) is a path with both endpoints in P0 ∪ P1 ∪ ... ∪ Pi−1.



9. Processor Allocation

1. **Static Allocation**

   Static allocation of tasks to processors is performed before the execution of the algorithm. We need to know the execution time of tasks,their interaction,and the size of data transferred between tasks to be able to perform this allocation

2. **Dynamic Load Balancing**

   In dynamic load balancing,we allocate the tasks to the processor during the execution of the algorithm.

   Wemay have a centralized load balancing scheme in which a central process commonly called the **supervisor** manages load distribution. Whenever a processor becomes idle, supervisor is informed which provides a new subtask/data to the idle worker. Fully distributed approaches have no supervisor processes and monitoring of load at each processor is performed by equal workers which exchange
   their states during execution of the parallel program。