Graphs

http://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon

Carlos Moreno
cmoreno@uwaterloo.ca
EIT-4103

https://ece.uwaterloo.ca/~cmoreno/ece250

These slides, the course material, and course web site are based on work by Douglas W. Harder

Graphs

Standard reminder to set phones to silent/vibrate mode, please!

Graphs

- So far, in ECE-250 ...
  - We've looked at algorithms, analysis, and some sorting algorithms.
  - Looked at data relationships and how they affect the choice of data structures.
    - Saw some of the sequential structures, for storing linearly ordered data (arrays, linked lists, queues, stacks)
    - Hash tables for unordered data (e.g., *sets*, in the strict mathematical sense of a set)
    - Trees, for hierarchical data (plus some other interesting applications deriving from their structure)

Graphs

- Today ...
  - We start with Graphs, the data structure for data featuring *adjacency* relationships.
  - We'll look into the basic notions and introductory concepts.
  - Discuss the main types of graphs (directed vs. undirected; weighted vs. unweighted, connected, complete, etc.)
  - Talk about some graph algorithms and their applications.

Graphs

- Basic definition:
  - A *graph* is defined as a set of vertices together with a set of edges representing association or adjacency between the vertices.

Graphs

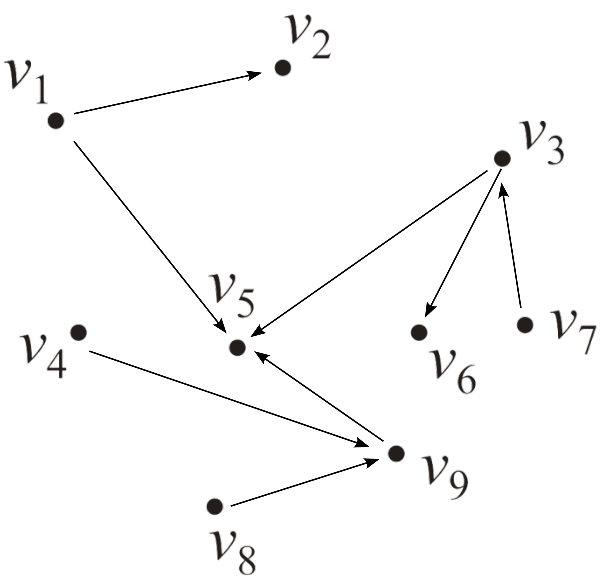
- Basic definition:
  - A *graph* is defined as a set of vertices together with a set of edges representing association or adjacency between the vertices.
- Notation:
  - A graph $G = (V, E)$ consists of the set of vertices $V = \{v_1, v_2, \cdots, v_{n-1}, v_n\}$ and the set of edges E, where each edge is a pair (v_i, v_j), with $v_i, v_j \in V$

Graphs

- Basic definition:
  - A *graph* is defined as a set of vertices together with a set of edges representing association or adjacency between the vertices.
- Notation:
  - A graph $G = (V, E)$ consists of the set of vertices $V = \{v_1, v_2, \cdots, v_{n-1}, v_n\}$ and the set of edges E, where each edge is a pair (v_i, v_j) , with $v_i, v_j \in V$
  - The number of vertices (in this case, *n*) is usually denoted $|V|$, and the number of edges $|E|$

Graphs

- Graphically, we could represent it like this (this is an example of a graph with $|V| = 9$:



Graphs

- Since the vertices usually represent some element (not unlike nodes in a tree), it is common practice to draw them as circles containing a value.

Graphs

- Since the vertices usually represent some element (not unlike nodes in a tree), it is common practice to draw them as circles containing a value.
  - So, in a sense (at least from a “visual analogy” point of view) a graph ends up being like a “generalized” or “extended” version of a tree.

Graphs

- More definitions — *directed* and *undirected* graphs:
  - A *directed* graph is one where edges are *ordered* pairs (v_i, v_j), where v_j is adjacent to v_i

Graphs

- More definitions — *directed* and *undirected* graphs:
  - A *directed* graph is one where edges are *ordered* pairs (v_i, v_j), where v_j is adjacent to v_i
  - Visually, edges are represented with arrows, denoting a direction in the adjacency relationship (arrow going from v_i to v_j).

Graphs

- More definitions — *directed* and *undirected* graphs:
  - An un*directed* graph is one where edges are un*ordered* pairs (v_i, v_j), where both v_i is adjacent to v_j, and v_j is adjacent to v_i

Graphs

- More definitions — *directed* and *undirected* graphs:
  - An un*directed* graph is one where edges are un*ordered* pairs (v_i, v_j), where both v_i is adjacent to v_j, and v_j is adjacent to v_i
  - In this case, edges are represented as lines connecting the two vertices.

Graphs

- Standard assumption:
  - A vertex is never adjacent to itself — that is, the set of edges shall never contain (v_k, v_k)

Graphs

- Maximum number of edges in an undirected graph with *n* vertices:

Graphs

- Maximum number of edges in an undirected graph with *n* vertices:

Graphs

- Maximum number of edges in an undirected graph with *n* vertices:

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

- Thus, maximum number of edges in a directed graph is:

Graphs

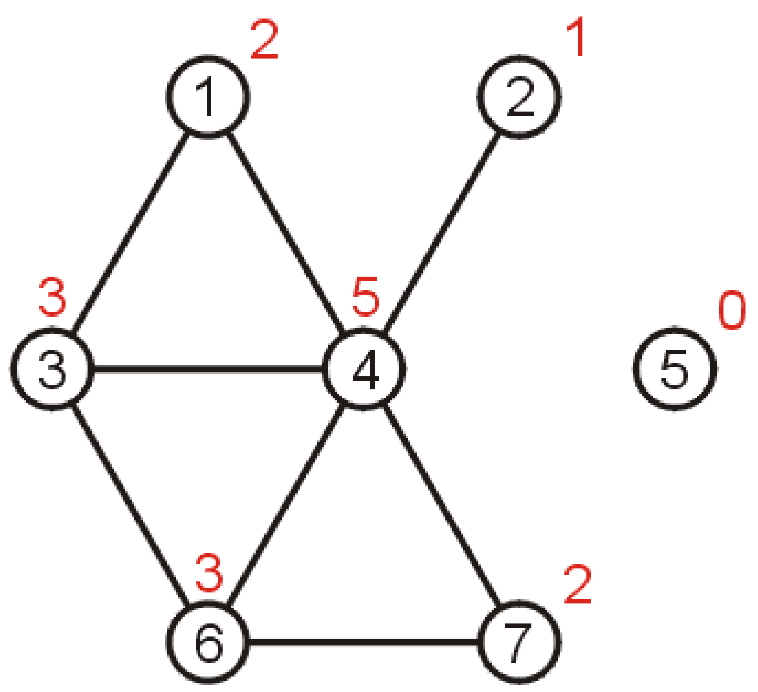
- Maximum number of edges in an undirected graph with *n* vertices:

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

- Thus, maximum number of edges in a directed graph is: twice as many — $n(n-1)$

Graphs

- The degree of a vertex (in an undirected graph) is defined as the number of adjacent vertices.
  - In the example below, the degree of each vertex is shown in red, next to the vertex:



Graphs

- What about for directed graphs? How do we define this “degree” notion for a directed graph?

Graphs

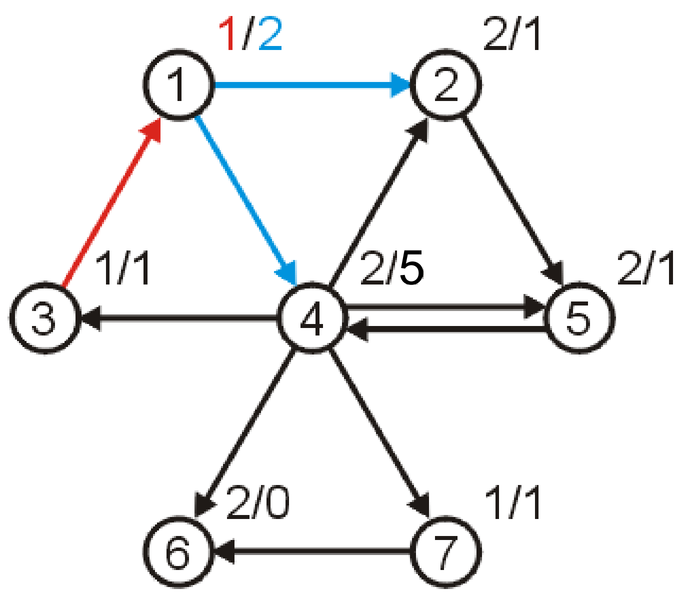
- What about for directed graphs? How do we define this "degree" notion for a directed graph?
  - There are the notions of *in degree* and *out degree*; visually, the in degree of a vertex corresponds to the number of arrows pointing to that vertex, and the out degree corresponds to the number of arrows starting at the vertex.

Graphs

- What about for directed graphs? How do we define this "degree" notion for a directed graph?
  - There are the notions of *in degree* and *out degree*; visually, the in degree of a vertex corresponds to the number of arrows pointing to that vertex, and the out degree corresponds to the number of arrows starting at the vertex.
  - Formally:
    - Out-degree of a vertex: number of vertices which are adjacent to the given vertex.
    - In-degree of a vertex: number of vertices which the given vertex is adjacent to.

Graphs

- Example: in/out degrees shown:



Graphs

- Definition: A *path* is an ordered sequence of vertices $(v_{0,} v_{1,} v_2, \cdots v_{k-1}, v_k)$

  where $(v_{i-1}, v_i) \in E \quad \forall\, i, \quad 1 \leqslant i \leqslant k$

Graphs

- Definition: A *path* is an ordered sequence of vertices $(v_{0,} v_{1,} v_2, \cdots v_{k-1}, v_k)$

  where $(v_{i-1}, v_i) \in E \quad \forall\, i, \quad 1 \leqslant i \leqslant k$

- This is a path from v_0 to v_k
- The length of this path is k

Graphs

- Definition: A *path* is an ordered sequence of vertices $(v_{0,} v_{1,} v_2, \cdots v_{k-1}, v_k)$

  where $(v_{i-1}, v_i) \in E \quad \forall i, \quad 1 \leqslant i \leqslant k$

- This is a path from v_0 to v_k
- The length of this path is k
  - Notion similar than with a path in a tree.
  - Important distinction: in this case, we're unrestricted in direction and the sequence of vertices.

Graphs

- Example:
  - Consider a graph where vertices represent an actor/actress, and edges represent associations between actors/actresses that have shared roles in the same movie.

Graphs

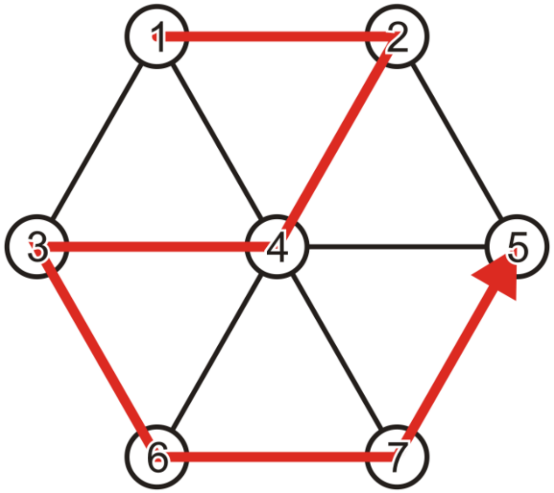
- Example:
  - Consider a graph where vertices represent an actor/actress, and edges represent associations between actors/actresses that have shared roles in the same movie.
  - Then, the length of the path from Kevin Bacon to any other vertex is claimed to be less than 6 !!

Graphs

- Example:
  - Here's Kevin Bacon himself explaining this important concept from graph theory:

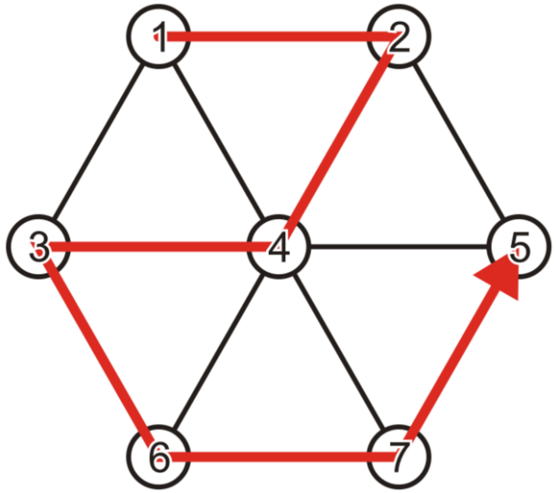
Graphs

- Examples of paths:
  - (1, 2, 3, 3, 6, 7, 5)

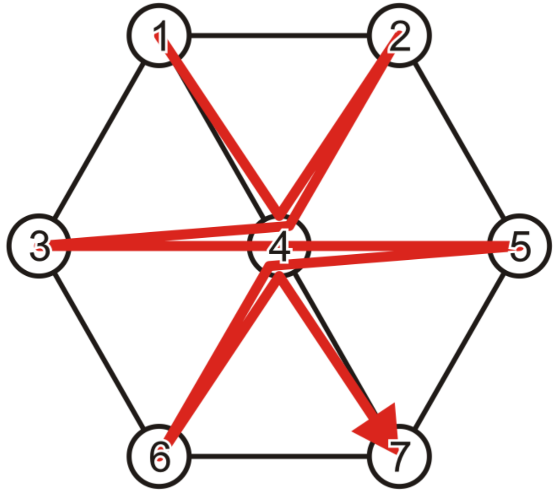


Graphs

- Examples of paths:
  - (1, 2, 3, 3, 6, 7, 5)

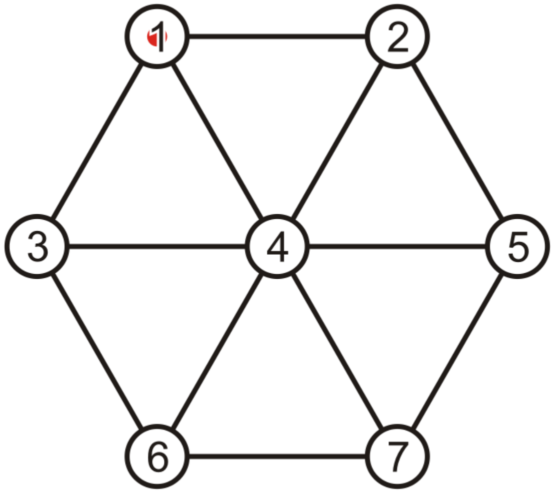


  - (1, 4, 2, 4, 3, 4, 5, 4, 6, 4, 7)



Graphs

- A perhaps curious example of a path:
  - (1)



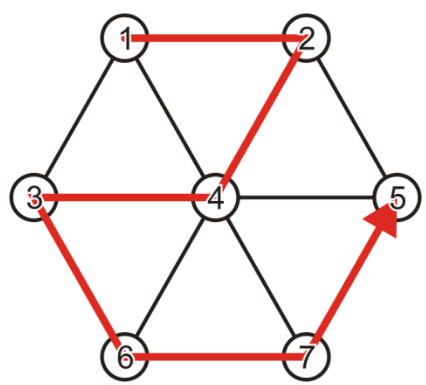
Graphs

- A *simple path* is one that has no repeated vertices other than possibly the first and last.

Graphs

- A *simple path* is one that has no repeated vertices other than possibly the first and last.

- Example of a simple path:



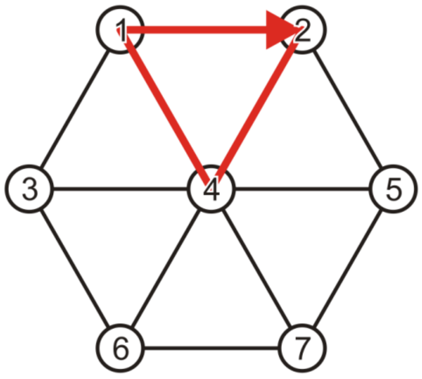
Graphs

- A *cycle* is a non-trivial simple path where the first and last vertices are the same vertex.

Graphs

- A *cycle* is a non-trivial simple path where the first and last vertices are the same vertex.

- Example of a cycle:
  - (2, 4, 1, 2)



Graphs

- There are some fascinating theoretical aspects related to this. For example:
  - A Hamiltonian path is a path that visits each vertex exactly once.
  - A Hamiltonian cycle is a Hamiltonian path that is a cycle.
  - An Eulerian path is a path that visits each edge exactly once.

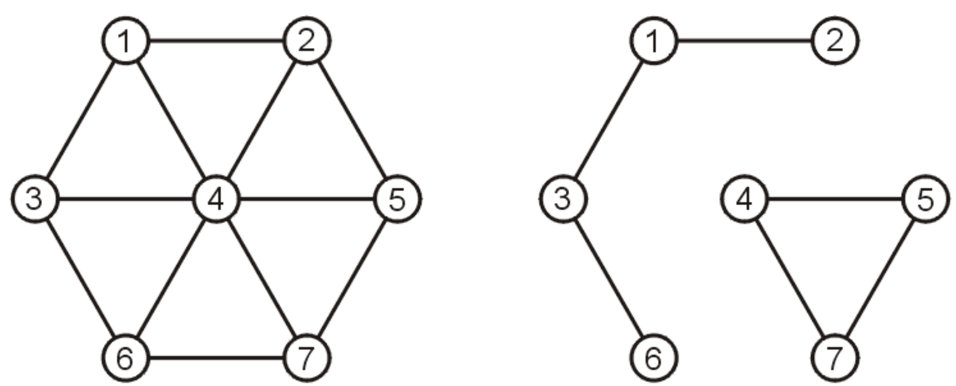
Graphs

- The problems of determining whether a graph contains a Hamiltonian path, a Hamiltonian cycle, or an Eulerian path are quite interesting from a theoretical point of view:
  - As much as they seem almost identical in terms of difficulty, testing for an Eulerian path can be done very efficiently.
  - Testing for Hamiltonian path or cycle has been proved to be *NP-complete* — a class of problems that we'll see, are as close as we get to proving that no efficient solution exists.

Graphs

- Definition: Two vertices *v*, *w* are connected if there exists a path from *v* to *w*.

- A graph is connected if there exists a path between any two vertices.



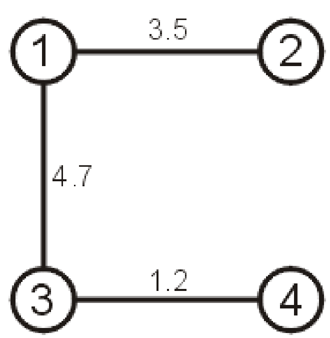
Connected Unconnected

Graphs

- Weighted graphs:
  - A numeric value (denoted *weight*, or *cost*) may be associated with each edge in a graph.
    - This could represent a distance, time, energy consumption associated with going from one vertex to the other, etc.

Graphs

- Weighted graphs:
  - A numeric value (denoted *weight*, or *cost*) may be associated with each edge in a graph.
    - This could represent a distance, time, energy consumption associated with going from one vertex to the other, etc.
  - Example:



Graphs

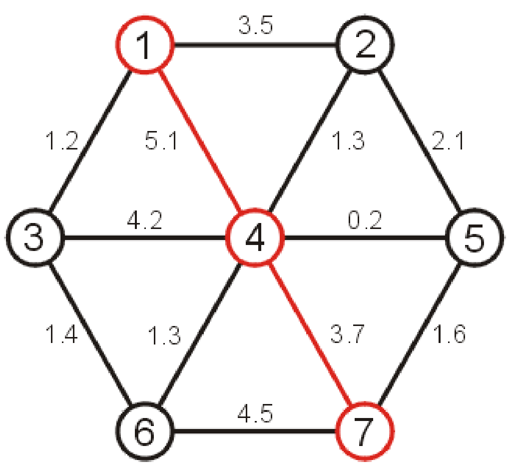
- Weighted graphs:
  - Observation: an unweighted graph could be seen as a weighted graph where every edge has weight 1.

Graphs

- Weighted graphs:
  - For weighted graphs, the length of a path is the sum of the weights of all edges in the path.

Graphs

- Weighted graphs:
  - For weighted graphs, the length of a path is the sum of the weights of all edges in the path.
  - Example: the length of the path (1, 4, 7) in the graph below is 5.1 + 3.7 = 8.8



Graphs

- Weighted graphs:
  - One typical application is finding directions — a graph where vertices represent intersections and edges represent streets.

Graphs

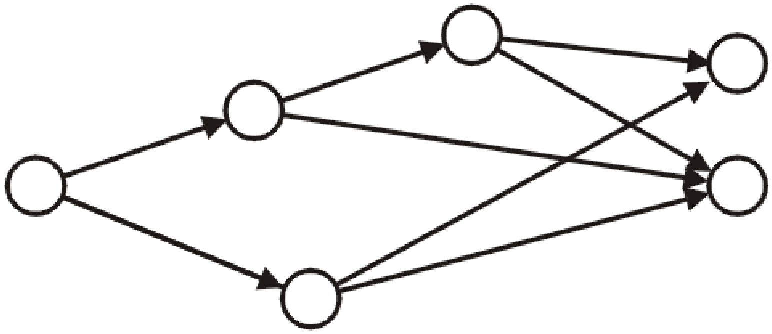
- Weighted graphs:
  - One typical application is finding directions — a graph where vertices represent intersections and edges represent streets.
  - Is this a directed or undirected graph?

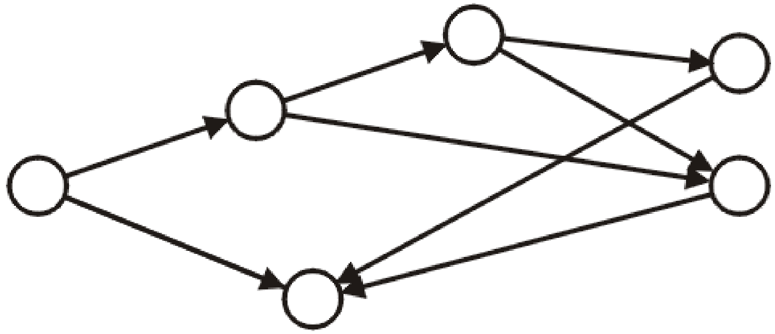
Graphs

- Weighted graphs:
  - One typical application is finding directions — a graph where vertices represent intersections and edges represent streets.
  - Is this a directed or undirected graph?
    - If we must account for one-way streets, then we have to make it a directed graph.
  - Weights could represent the estimated time (perhaps based on traffic statistics, combined with speed limit, etc.)

Graphs

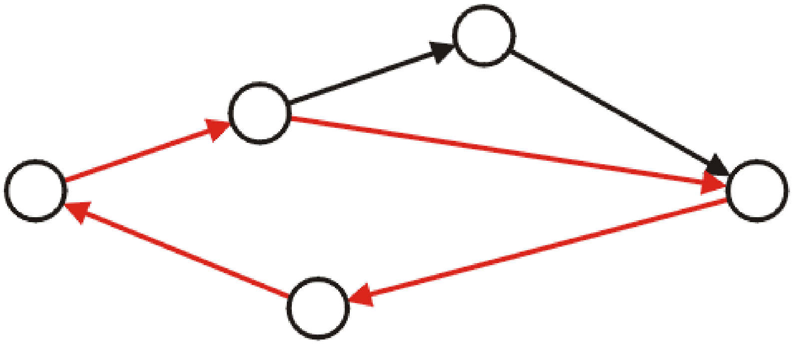
- Directed Acyclic Graphs (DAGs):
  - A directed graph that has no cycles.
  - Examples:





Graphs

- Directed Acyclic Graphs (DAGs):
  - A directed graph that has no cycles.
  - Example of something that is *not* a DAG:



Graphs

- Directed Acyclic Graphs (DAGs):
  - A directed graph that has no cycles.
  - Hmm ... What does this sound like? (what type of relationship could be represented here?)

Graphs

- Directed Acyclic Graphs (DAGs):
  - A directed graph that has no cycles.
  - Hmm ... What does this sound like? (what type of relationship could be represented here?)
    - In general, we're talking about a *partial ordering*.
    - Specific examples include course prerequisite diagrams, compiler optimization diagrams for code dependencies.

Graphs

- In the next few lessons, we'll look at some of the algorithms to solve particular problems with graphs (notably, shortest path, minimum spanning tree, topological sort).

Graphs

- In the next few lessons, we'll look at some of the algorithms to solve particular problems with graphs (notably, shortest path, minimum spanning tree, topological sort).

Summary

- During today's lesson, we:
  - Introduced Graphs — data structure for storing data featuring *adjacency* relationships.
  - Saw some of the basic notions.
  - Discussed the main types of graphs (directed vs. undirected; weighted vs. unweighted, connected, complete, etc.)
  - Mentioned some graph algorithms and their applications.