Wege finden in Tournament-Graphen

Sommerakademie Rot an der Rot — AG 1
Wieviel Platz brauchen Algorithmen wirklich?

Dmitrijs Dmitrenko

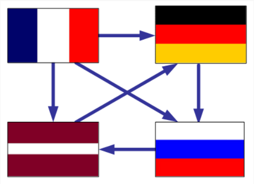
Institut für Informatik
Universität Rostock

9. August 2010

Outline

Tournament-Graphs and Their Paths

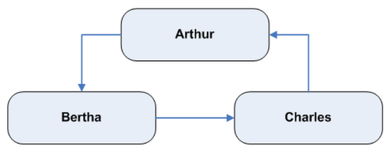
Properties of the Tournament-Graph



- Directed graph obtained by assigning a direction for each edge in an undirected complete graph
- Directed graph in which every pair of vertices is connected by a single directed edge
- Emerges e.g. when knights fight with each other, showing who has won against whom

Problem Description - Knights' Tournament Example

- Knights are nodes, wins are edges
- Direction of the edge defines the victory
- Indirect wins:
  - Arthur beats Bertha
  - Bertha beats Charles
  - Charles beats Arthur



How Difficult Is It to Tell Whether a Path Exists?

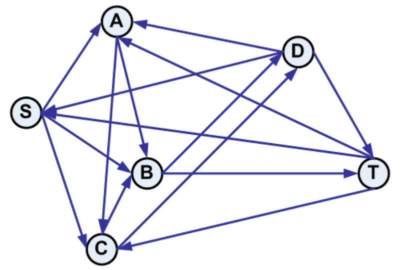
- The reachability problem (REACH) for finite directed graphs is **NL-complete**
- If the independence number of finite directed graphs is bounded by a constant k, the REACH complexity is much lower - first order definable for all k
  - Formally, for each k the language $REACH_{a \leq k} := REACH \cap \{\langle G, s, t\rangle | \alpha(G) \leq k\}$ is first order definable, where $\langle\rangle$ denotes a standard binary encoding
  - Such languages can be decided by AC^0-circuits, in constant parallel time on concurrent-read, concurrent-write parallel random access machines (CRCW-PRAMs), and in **logarithmic space**
- **Succinctly represented graphs** are given indirectly via a program or a circuit that decides the edge relation of the graph
- SUCCINT-REACH is **PSPACE-complete**

How Difficult Is It To Construct a Path?

- A simple algorithm can be applied:
  - Start at source
  - Check, whether the target can be reached from the successor
  - Make a suitable successor a current vertex
  - Repeat, until the target is reached

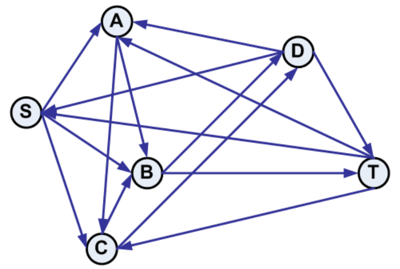
How Difficult Is It To Construct a Path?

- A simple algorithm can be applied:
  - Start at source
  - Check, whether the target can be reached from the successor
  - Make a suitable successor a current vertex
  - Repeat, until the target is reached



How Difficult Is It To Construct a Path?

- A simple algorithm can be applied:
  - Start at source
  - Check, whether the target can be reached from the successor
  - Make a suitable successor a current vertex
  - Repeat, until the target is reached



- How about tournament-graphs with 10 000 nodes?

Path Construction Issues

- A correct algorithm does not move to any successor, but to the successor that is **nearest to the target**, searching the shortest path
- A corrected algorithm does not only produce some path, but the **shortest** one
- A path between any two connected vertices can be constructed in **logarithmic space** in graphs with bounded independence number (**logspace approximation scheme**)

How Difficult Is It To Construct a Shortest Path?

- Constructing the shortest path in the tournament graph is as difficult as performing the same task in the arbitrary graph
- The complexity of constructing the shortest path depends on the complexity of the **distance problem:**
  $DISTANCE_{tourn} := \{\langle G, s, t, d\rangle | G$ is a tournament in which there is a path from s to t of length at most d$\}$
- This problem is **NL-complete**
- The **succinct version** of DISTANCE$_{tourn}$ is **PSPACE-complete**

Graph-Theoretic Terminology and Known Results

Definitions (1)

- A **(directed) graph** is a nonempty set V of vertices together with a set $E \subseteq V \times V$ of directed edges
- A graph is **undirected** if its edge relation is symmetric
- A **forest** is an undirected, acyclic graph
- A **tree** is a connected forest
- A **path of length** l in a graph $G = (V, E)$ is a sequence $(v_0, ..., v_l)$ of distinct vertices with $(v_i, v_{i+1}) \in E$ for $i \in \{0, ..., l-1\}$
- A vertex t is **reachable** from a vertex s if there is a path from s to t
- The **distance** $d(s, t)$ of two vertices is the length of the shortest path between them or ∞, if no path exists

Definitions (2)

- For $i \in \mathbb{N}$, a vertex $u \in V$ is said to **i-dominate** a vertex $v \in V$ if there is a path from u to v of length at most i
- A set $U \subseteq V$ is an **i-dominating** set for G if every vertex $v \in V$ is i-dominated by some vertex $u \in U$
- The **i-domination number** $\beta_i(G)$ is the minimal size of an i-dominating set for G
- A set $U \subseteq V$ is an **independent set** if there is no edge in E connecting vertices in U
- The maximal size of independent sets in G is its **independence number** $\alpha(G)$

Tournament Graphs (1)

- A **tournament** is a graph with exactly one edge between any two different vertices and $(v, v) \notin E$ for all $v \in V$
- The name tournament originates from such a graph's interpretation as the outcome of a **round-robin tournament** in which every player encounters every other player exactly once, and in which **no draws occur**
- Any tournament on a finite number n of vertices contains a **Hamiltonian path**, i.e., directed path on all n vertices
- A tournament in which $((a \rightarrow b)$ and $(b \rightarrow c)) \Rightarrow (a \rightarrow c)$ is called **transitive**. The following statements are equivalent for a tournament T on n vertices:
  1. T is transitive
  2. T is acyclic
  3. T does not contain a cycle of length 3
  4. The score sequence (set of outdegrees) of T is $\{0, 1, 2, ..., n-1\}$
  5. T has exactly one Hamiltonian path

Tournament Graphs (2)

- A tournament for which every player loses at least one game is called a **1-paradoxical** tournament, **k-paradoxical** if for every k-element subset S of V there is a vertex v_0 in $V \setminus S$ such that $v_0 \longrightarrow v$ for all $v \in S$
- The **score sequence** of a tournament is the nondecreasing sequence of outdegrees of the vertices of a tournament
- The **score set** of a tournament is the set of integers that are the outdegrees of vertices in that tournament

Fact:

Let $G = (V, E)$ be a finite graph with at least two vertices, $n := |V|$, $\alpha := \alpha(G)$, and $c := (\alpha^2 + \alpha)/(\alpha^2 + \alpha - 1)$. Then

1. $\beta_1(G) \leq \lceil \log_c n \rceil$ and
2. $\beta_2(G) \leq \alpha$

Complexity of the Approximation Problem

Problem Description

- Both finding, whether a **path between two vertices** exists and the **construction of such a path** in graphs with bounded independence number can be done in **logarithmic space**
- The shortest path can be constructed only if $\mathsf{L} = \mathsf{NL}$
- It is possible to find a path that is approximately as long as the shortest path
- There is a **logspace approximation scheme** for constructing paths whose length is as close to the length of the shortest path as one would like

Theorem:

For all k there exists a deterministic Turing machine M with read-only access to the input tape and write-only access to the output tape such that:

1. On input $\langle G, s, t, m \rangle$ with $\langle G, s, t \rangle \in \text{REACH}_{\alpha \leq k}$ and $m \geq 1$, it outputs a path from s to t of length at most $(1 + 1/m)d(s, t)$
2. On input $\langle G, s, t, m \rangle$ with $\langle G, s, t \rangle \notin \text{REACH}_{\alpha \leq k}$ it outputs 'no path exists'
3. It uses space $O(\log m \log n)$ on the work tapes, where n is the number of vertices in G

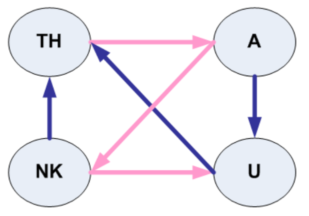
Complexity of the Distance Problem

Problem Description

- Decision of whether the distance of two vertices in a graph is smaller than a given input number
- This problem is NL-complete even for tournaments
- The succint version of this problem is PSPACE-complete
- We can easily solve the distance problem, if we have oracle access to an algorithm that constructs shortest paths
- The logspace algorithm for constructing shortest path in tournaments is impossible, unless L = NL

Conclusion

- Checking whether a path exists in a given graph can be done using AC^0-circuits
- Constructing a path between two vertices can be done in logarithmic space
- Constructing the shortest path in logarithmic space is impossible, unless $L = NL$
- The problem of shortest paths in graphs with bounded independence number cannot be solved exactly in logarithmic space (unless $L = NL$), but it can be approximated well: there exists a logspace approximation scheme for it
- The distance problem for directed graphs is just as hard as the reachability problem for directed graphs


TH
A
NK
U