Breaking the O(n m) Barrier for Büchi Games and Maximal End-Component Decomposition

Krishnendu Chatterjee (IST Austria)
Monika Henzinger (University of Vienna, Austria)

LIP6, Paris, July 25, 2012
This Talk

- Two classical algorithmic problems related to graph games and verification of probabilistic systems:
  - Büchi games.
  - Maximal end-component (MEC) decomposition.
- The long-standing best known bounds for these problems have been $O(n^m)$.
- This talk we will present algorithms for these problems to break the $O(n^m)$ barrier.
Motivation

- **Roll back the clock**
  - Tom Henzinger 2002: “very important and interesting algorithmic question”.
  - Orna Kupferman 2002: “very nice theoretical problem”.
  - Moshe Vardi 2022: talk on importance of Büchi automata and synthesis.

- **Real motivation**
  - Synthesis.
  - Model checking of open systems.
  - Probabilistic verification.
Graphs vs. Games

Two interacting players in games: Player 1 (Box) vs Player 2 (Diamond).

AND-OR Graphs.
Game Graphs
Game Graphs

- A game graph $G= ((V,E), (V_1, V_2))$
  - Player 1 states (or vertices) $V_1$ and similarly player 2 states $V_2$, and $(V_1, V_2)$ partitions $V$.
  - $E$ is the set of edges.
  - $E(v)$ out-going edges from $v$, and assume $E(v)$ non-empty for all $v$.
  - Notation: $n= |V|$, $m=|E|$.

- Game played by moving tokens: when player 1 state, then player 1 chooses the out-going edge, and if player 2 state, player 2 chooses the outgoing edge.
Game Example
Game Example
Game Example
Strategies are recipe how to move tokens or how to extend plays. Formally, given a history of play (or finite sequence of states), it chooses an outgoing edge.

- $\sigma: V^* V_1 \rightarrow V.$

- $\pi: V^* V_2 \rightarrow V.$
Reachability and Buechi Objectives

- **Reachability**: there is a set of good vertices and goal is to reach them. Formally, for a set $T$ of vertices, the objective is the set of infinite paths that visit the target $T$ at least once.

- **Buechi**: there is a good set of vertices and goal is to visit them infinitely often. Formally, for a set $B$ of vertices, the objective is the set of infinite paths that visit some vertex in $B$ infinitely often. The objective is a liveness objective (like progress condition in mutual exclusion protocol).
Winning Set

- Winning set: Starting vertices such that player 1 has a strategy to ensure the objective against all strategies of player 2.

- Remark: Memoryless strategies are sufficient.

- We are interested in computing the winning set in games for player 1 for Büchi objectives.
Previous Result

- Reachability games:
  - $O(m)$ (linear time algorithm) and PTIME-complete [Immerman 81, Beeri 80].

- Büchi games:
  - Classical algorithm: $O(n \cdot m)$ [EJ91].
  - In the special case when $m = O(n)$, an $O(n^2 / \log n)$ algorithm [CJH03]
Buechi Games Algorithm
Classical Algorithm

- A simple iterative algorithm using alternating reachability.

Steps are as follows:

1. Compute player-1 alt-reach set $A$ to the current Buechi set.

2. If $A$ is the set of all vertices of current game graph, then stop and output $A$ as the winning set.

3. Else $U$ be the remaining vertices (complement of $A$). Remove player-2 alt-reach set $C$ to the set $U$ from game graph and continue.
Classical Algorithm

- Compute alt-reach for player 1 to the set B. Let us call this set A.
Let $U = V \setminus A$. Then $U$ is a trap. Clearly, $U$ is not in winning for player 1.

Hence alt-reach for player 2 to $U$ is also not winning.
Classical Algorithm

- Iterate on the remaining sub-graph.
- Every iteration what is removed is not part of winning set.
- When the iteration stops, all remaining vertices are winning for player 1.
Correctness Proof Idea

- Player 2 cannot leave.
- Player 1 can ensure to reach, and again, and again and so on.
Classical Algorithm

- Classical algorithm identifies the largest trap and removes the trap.

- At most $n$ iterations with time $O(m)$ each.

- Analysis $O(nm)$ is tight.

- Remark: Player-2 alt-reachability overall iterations is $O(m)$ (edges worked on are removed from the graph).
Our New Algorithm

- Hierarchical graph decomposition technique.

- As long as we find traps, we can remove them, need not find the largest trap.

- Running time: $O(n^2)$
  - Better worst case for dense graphs.
  - Along with previous [CJH03] algorithm breaks $O(n \cdot m)$ for all cases.
New Algorithm

- Create log n graphs hierarchically.

- Game graph $G_i = (V, E_i)$ ensuring $|E_i|$ is at most $O(n \cdot 2^i)$. Some special way to select edges according to ordering.
Construction of $G_i$

- The graph $G_{i-1}$ is a sub-graph of $G_i$.
- In $G_i$, for every vertex add at most $2^i$ out-edges.
Construction of $G_i$

- The graph $G_{i-1}$ is a sub-graph of $G_i$.
- In $G_i$, for every vertex add at most $2^i$ out-edges.
- Then for every vertex add at most $2^i$ more in-edges with preference to edges from player-2 non-Buechi vertices.

At most $2^i$
New Algorithm

- Search for traps in $G_1$, $G_2$, and so on.
  - Call a player-1 vertex with edges more than $2^i$ as blue in $G_i$.
  - Compute alt-1 reachability to the set of Buechi or blue vertices.
  - If the complement is non-empty, then that is a trap (some sense largest trap (without Buechi and blue) in $G_i$).
  - In the trap all player-1 edges are retained (any player-1 vertex where edges are not retained are blue and not in trap).
  - Correctness follows.
Correctness of New Algorithm

- Correctness is as follows:
  - When identify a trap, then all player-1 edges of the original game graph in the trap.
  - When algorithm stops no trap as in the final graph all edges are retained.
  - Correctness follows from classical algorithm.

- Challenge is running time analysis
  - We are working on more edges possibly (edge belong to several graphs).
  - Not clear we gain anything.
  - Challenge is to analyze the size of the trap we discover.
Running Time Analysis

- Analysis of the size of the trap.
  - Trap U identified in $G_i$ but not in $G_{i-1}$.
  - We analyze the size of the trap we identify.
Running Time Analysis

- Analysis of the size of the trap.
  - Trap U identified in $G_i$ but not in $G_{i-1}$.
  - Case 1: U contains a player-1 vertex $v$ that was blue in $G_{i-1}$.
  - Then $v$ has at least $2^{i-1}$ out-edges, otherwise would not have been blue.
  - Since a trap all out-going edges from $v$ in trap. Size of trap at least $2^{i-1}$. 

\[ G_{i-1} \quad \text{Trap U} \quad G_i \]
Running Time Analysis

- Analysis of the size of the trap.
  - Trap U identified in $G_i$ but not in $G_{i-1}$.
  - Case 1: Done. U does not contain a player-1 vertex v that was blue in $G_{i-1}$. All player-1 edges in $G_i$ and $G_{i-1}$ identical.
  - Case 2: Two sub-cases to analyze.
Running Time Analysis

- Analysis of the size of the trap.
  - Case 1: All player-1 edges in $G_i$ and $G_{i-1}$ identical.
  - Case 2 (a): All player-2 edges in $G_i$ and $G_{i-1}$ are identical. Then $U$ is a trap in $G_{i-1}$ and this a contradiction.
  - Case 2(b): One new player-2 edge in the trap.
Running Time Analysis

- Analysis of the size of the trap.
  - Case 1: All player-1 edges in $G_i$ and $G_{i-1}$ identical.
  - Case 2 (a): All player-2 edges in $G_i$ and $G_{i-1}$ are identical.
  - Case 2(b): One new player-2 edge $(u,v)$ in the trap.
  - Vertex $v$ has at least $2^{i-1}$ in edges from player-2 non-Buechi vertices as they have the priority.
Running Time Analysis

- Analysis of the size of the trap.
  - Case 1: All player-1 edges in $G_i$ and $G_{i-1}$ identical.
  - Case 2 (a): All player-2 edges in $G_i$ and $G_{i-1}$ are identical.
  - Case 2(b): One new player-2 edge $(u,v)$ in the trap.
  - Vertex $v$ has at least $2^{i-1}$ in edges from player-2 non-Buechi vertices as they have the priority.
Running Time Analysis

- Analysis of the size of the trap.
  - Case 1: All player-1 edges in $G_i$ and $G_{i-1}$ identical.
  - Case 2 (a): All player-2 edges in $G_i$ and $G_{i-1}$ are identical.
  - Case 2(b): One new player-2 edge $(u,v)$ in the trap.
  - Vertex $v$ has at least $2^{i-1}$ in edges from player-2 non-Buechi vertices as they have the priority.
  - All in-edges in the trap.
  - Size of trap at least $2^{i-1}$. 

$G_{i-1}$  

$G_i$ 

Trap U
New Algorithm

- Search for traps in $G_1$, $G_2$, and so on.
- If we find a trap, then remove it from all graphs.
- Key argument: if we find a trap in $G_i$, then size of trap is at least $2^{i-1}$.
- Work done $O(n \cdot 2^{i+1})$ and charge to vertices removed.
Result

- New algorithm: $O(n^2)$.
  - Time spent to identify trap is $O(n \cdot 2^{i+1})$ and charge to the trap of size $2^{i-1}$.

- Strikingly simple algorithm breaks the long standing $O(n \cdot m)$ barrier.

- Along with $O(n^2 / \log n)$ algorithm for $m=O(n)$ of [CJH03] we break $O(n \cdot m)$ barrier for all cases.
Maximal End-component Decomposition
### Maximal End-component Decomposition

- An end-component $U$ is a set of vertices such that
  - Graph induced by $U$ is strongly connected.
  - For all player-2 vertices in $U$ all out-going edges end in $U$.
  - Typically used in MDPs (where player 2 is the probabilistic player).

- We keep the notations uniform as our goal is to present algorithm for the problem.

- **Maximal end-component (MEC) decomposition:**
  - Classical algorithm: $O(n \cdot m)$ [CY95, deAlfaro97]
Classical Algorithm

- A simple iterative algorithm using scc decomposition and alternating reachability.

Steps are as follows:
1. Compute the bottom scc’s. They are all mec’s and let their union be U.
2. Remove player-2 alt-reach set C to the set U from game graph and continue.
3. Stop when all vertices are removed.

Remark: Player-2 alt-reach over all iterations is O(m). Main work is repeated scc decomposition.
New Algorithm

- Same as for Buechi games using hierarchical graph decomposition technique.
- Instead of traps search for bottom scc’s.
- $O(n^2)$ time algorithm.
- We also present a different improved algorithm.
Different Sub-quadratic Algorithm

- The different sub-quadratic algorithm splits the classical algorithm as follows:

  - If more than $m^{0.5}$ edges were lost, then use classical scc decomposition algorithm.

  - If less than $m^{0.5}$ edges were lost, then use Tarjan scc algorithm with vertices having lost edges as starting point in lock-step.
Different Sub-quadratic Algorithm
Different Sub-quadratic Algorithm
Different Sub-quadratic Algorithm

Stop when a bottom scc $C$ is found. The bottom scc $C$ is removed and work done is at most edges in $C$ times $m^{0.5}$. 
Different Sub-quadratic Algorithm

- The different sub-quadratic algorithm splits the classical algorithm as follows:

  - If more than $m^{0.5}$ edges were lost, then use classical SCC decomposition algorithm.
    - Total work: $O(m^{1.5})$ since at most $m^{0.5}$ iterations of $O(m)$ time each.
  - If less than $m^{0.5}$ edges were lost, then use Tarjan SCC algorithm with vertices having lost edges as starting point in lock-step.
    - Total work: $O(m^{1.5})$ since every removed edge is charged at most $O(m^{0.5})$. 
MEC Decomposition

- Two algorithms:
  - $O(n^2)$.
  - $O(m^{1.5})$.

- Clearly can do the min of the above two.
Summary

- Buechi games: a simple $O(n^2)$ time algorithm improving long-standing $O(n \cdot m)$ bound.

- MEC decomposition in time $O(\min(m^{1.5},n^2))$ (worst case $O(m \cdot n^{2/3})$).
Conclusion

- **Buechi games and MEC decomposition:**
  - A core algorithmic problem in verification with long-standing $O(nm)$ barrier.
  - We present a simple $O(n^2)$ time algorithm for the problem, also for mec decomposition.
  - For mec decomposition also $O(m^{1.5})$ algorithm that gives a worst case $O(mn^{2/3})$ algorithm.

- **Open questions:**
  - $O(mn^{1-\varepsilon})$ or $O(nm^{1-\varepsilon})$ for Buechi games, for some $\varepsilon > 0$.
  - $O(mn^{1/2})$ algorithm for mec decomposition.
Thank you!

Questions?