Finding Triangles or Independent Sets

Adrian Dumitrescu

Algoresearch L.L.C., Milwaukee, WI, USA

Second part of the presentation is based on joint work with

Andrzej Lingas

Lund University, Sweden

Abstract

We revisit the algorithmic problem of finding a triangle in a graph (Triangle Detection), and examine its relation to other problems such as 3Sum and Independent set. We discuss several new algorithms:

(I) A randomized algorithm which given a graph $G = (V, E)$ with n vertices and m edges, computes a $(1 \pm \varepsilon)$-approximation of the number of triangles in G and finds a triangle with high probability. We use this algorithm in relation to a question of Pătraşcu (2010) regarding the triangle detection problem.

(II) An algorithm which given a graph $G = (V, E)$ performs one of the following tasks in $O(m+n)$ (i.e., linear) time: (i) compute a $\Omega(1/\sqrt{n})$-approximation of Maximum Independent Set in G or (ii) find a triangle in G.

(III) An algorithm which given a graph $G = (V, E)$ performs one of the following tasks in $O(m + n^{3/2})$ time: (i) compute a $\sqrt{n}$-approximation for Graph Coloring of G or (ii) find a triangle in G. The run-time is faster than that for any previous method for each of these tasks on dense graphs, with $m = \omega(n^{9/8})$.

(IV) We revisit the algorithmic problem of finding all triangles in a graph $G = (V, E)$ with n vertices and m edges. Chiba and Nishizeki (1985) gave a combinatorial algorithm running in $O(m\alpha) = O(m^{3/2})$ time, where $\alpha = \alpha(G)$ is the graph arboricity. We provide a new very simple combinatorial algorithm for finding all triangles in a graph that runs in the same time.

(V) We give improved arboricity-sensitive running times for counting and/or detection of copies of K_ℓ, for small $\ell \geq 4$. Our new algorithms are faster than all previous algorithms in certain high-range arboricity intervals for every $\ell \geq 7$.