Counting Paths in Graphs

1 Homomorphisms for Counting Paths

Lovász was one of the first to recognize the close relationship between the number of subgraphs $\#\text{Sub}(H, G)$ and the number of homomorphisms $\#\text{Hom}(H, G)$. Radu Curticapean, Holger Dell, and Dániel Marx formalized these insights under the name 'Graph Motif Parameters' [1]. Furthermore, they showed that

$$\#\text{Sub}(H,G) = \sum_{\rho} \frac{(-1)^{|V(H)|-|V(H_\rho)|} \cdot \prod_{B\in\rho}(|B|-1)!}{\#\text{Aut}(H)} \cdot \#\text{Hom}(H_\rho, G) \quad (1)$$

holds. Here, we sum over all partitions ρ of the vertex set of H, such that the quotient graph H_ρ is from the set of homomorphic images of H. Essentially one can think of the homomorphic images as the graphs we obtain by contracting non neighboring vertices. Regardless of the partitions, the number of automorphisms of a path remains 2. In our implementation, we generate the set of homomorphic images for $k+1$ by considering all possibilities of extending the graphs in the homomorphic images for k. One can prove that the number of such homomorphic images for P_k is exactly the Bell number $B(k-1)$, which is the number of ways we can partition $k-1$ distinguishable objects. Moreover, one can prove that for any other connected graph the number of homomorphic images is $\leq B(k-1)$, so in some sense, the problem to count paths is more challenging compared to other subgraphs. We use the library 'nauty' to check for graph isomorphisms and the library 'homlib' to compute graph homomorphisms. It turned out that homlib didn't work correctly for large graphs. We fixed some issues in the dynamic programming routine of homlib, but further testing is needed to verify homlib for larger graphs. We achieve a runtime of $\mathcal{O}((\frac{0.792(k-1)}{ln(k)})^{k-1} \cdot n^{0.174k+\mathcal{O}(1)})$.

2 Paths for fixed Terminals

Let s and t be the fixed terminal vertices, and let k be the upper bound on the number of vertices in an s-t path we want to count. We apply the following reduction rules:

- delete all vertices v where $distS[v] + distT[v] \geq k$
- delete all vertices which are not in the same connected component as s and t

Now all remaining vertices are contained in some s-t path on $\leq k$ vertices. Compute dp[mask][v] where mask is the set of vertices used in the path so far ending with vertex v. Afterwards, we need to calculate the transition of our dynamic programming approach by considering all possibilities to extend the mask. This leads to a runtime of $\mathcal{O}(2^n \cdot n^2)$.

References

[1] Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 210–223, New York, NY, USA, 2017. ACM.