Maximum Clique
Maximum $k$-Clique
Maximum $k$-Club
Existing Work

- Many computational papers on the $k$-club problem.
  - Not a hereditary property, and modelling it is hard, so lots of scope for being clever.
- “Unlike the maximum clique problem, the maximum $k$-clique problem has not been the subject of extensive research and we are not aware of any computational results for this problem to date.” (Shahinpour and Butenko, 2013)
Reducing $k$-Clique to Clique
But Is This Practical?

- We probably want to work with large sparse graphs as inputs.
- There are many good maximum clique algorithms for large sparse graphs.
- But $G^k$ is large and dense!
- There are also good maximum clique algorithms for small dense graphs, but small dense graphs can be very hard.
  - $\omega(G(500, 0.9))$ can’t be solved in a CPU-decade.
- It’s well known in the CP community that reduction to clique is a bad way of solving things.
  - (This is not necessarily true, but it’s well known... )
The Good News

- Start with a state-of-the-art maximum clique algorithm for small dense graphs.
- Replace cubic preprocessing and inference with cheaper quadratic algorithms.
- Spend some time making the $G^k$ reduction fast.
- Buy a hefty amount of RAM.
The Results ($k \in \{2, 3, 4\}$)

- **Erdős collaboration graphs:**
  - $|V| \leq 6,927$, $|E| \leq 11,850$, $D(G^k) \leq 0.57$ (or $= 1$).
  - 16 take $< 1s$, 2 take $< 12s$, 3 $> 1h$.

- **DIMACS 2 Clique Graphs with diameter $> 2$:**
  - $|V| \leq 500$, $|E| \leq 46,627$, $D(G^k) \leq 0.87$ (or $= 1$)
  - 22 take $< 1s$, 2 take $> 1h$.

- **DIMACS 10 Partitioning Graphs (smallest 20):**
  - $|V| \leq 36,519$, $|E| \leq 1,007,284$, $D(G^k) \leq 0.67$.
  - 2 take $> 1h$, most take well under a minute.

- **DIMACS 10 Clustering Graphs (smallest 20):**
  - $|V| \leq 40,421$, $|E| \leq 175,691$, $D(G^k) \leq 0.99$.
  - 7 take $> 1h$, most take well under a minute.
But We Can Do Better
Better Results!

- Erdős collaboration graphs:
  - The three open instances now take 3.8s, 69.4s, 207.3s.
- DIMACS 2 Clique Graphs:
  - The two open instances now take 0.1s.
- DIMACS 10 Partitioning Graphs:
  - One open instance now takes 16.9s, the other still takes > 1h.
- DIMACS 10 Clustering Graphs:
  - Two open instances closed.
- Not a universal success, though:
  - Factor of 10 slowdown on a few fairly easy instances.
  - Without laziness, the results are terrible...
What About Parallel Search?

- Active research topic for maximum clique.
  - Work stealing strategies really make a huge difference (often more so than load balance)!
- In this paper:
  - Dynamic work splitting, starting at the top, and working downwards.
  - Use parallelism to steal early, where value ordering heuristics are weakest—a bit like discrepancy search or restarts.
Even Betterer Results!

- Two open instances closed.
- Improved bounds on four remaining instances.
- Also super-linear speedups in (at least) two cases.
- But only if we get it right:
  - Again, work stealing strategies matter.
  - Load balancing is harder: must use a much deeper splitting limit than for typical maximum clique graphs.
This technique is the only known practical way of getting:
- Reproducible runtimes.
- Guarantees of no (exponential) slowdown over sequential, or when adding cores.
- Decent work distribution.
- Respectable speedups in practice.

I personally think these properties are all critical for practical multi-core parallel search.
What About Random Graphs?

Average Search Nodes

Edge Probability

$k = 1$

$k = 2$

$k = 3$

$k = 4$
Conspicuously Missing From This Paper

- Parallel $G^k$ construction.
- Predicting $G(n, p)^k$ on random graphs.
- So just why are $G^k$ graphs so easy to solve, anyway?
Conclusion

- There are many clique relaxations (density-based, degree-based, distance-based, ...). It’s often not clear which you’d want in practice.
- $k$-Clique tends to be easy to calculate, at least.
- Usually the $k$-Clique and $k$-Club numbers are the same, too.
- The domination rule is probably useful in other settings.
- Work-stealing strategies matter when doing parallel search.