Local Search for Max-Cut

CS 181 — Advanced Algorithms

The Max-Cut Problem

Max-Cut (Unweighted). Given an undirected graph $G = (V, E)$, find a partition of V into two sets S and $\bar{S} = V \setminus S$ that maximizes the number of *crossing edges*:

$$\text{cut}(S) \;=\; \big|\{\, \{u, v\} \in E \mid u \in S,\; v \in \bar{S} \,\}\big|.$$

Max-Cut is **NP-hard**, so we seek an efficient approximation algorithm. We saw a *local search* algorithm: start with any partition, and repeatedly improve it by moving a single vertex from one side to the other.

The Local Search Algorithm

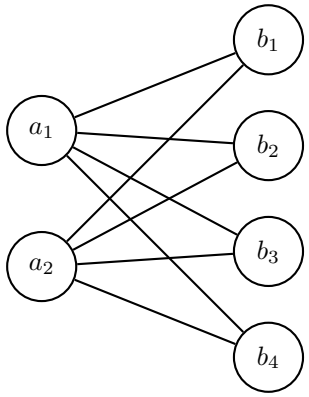
Algorithm LocalSearch(G):

1. Start with an arbitrary partition $(S, \bar{S})$ of V.
2. **While** there exists a vertex $v \in V$ such that moving v to the other side increases $\text{cut}(S)$:
   - Move v (i.e., if $v \in S$ set $S \leftarrow S \setminus \{v\}$; if $v \in \bar{S}$ set $S \leftarrow S \cup \{v\}$).
3. Return the partition $(S, \bar{S})$.

A partition is called **locally optimal** if no single-vertex move increases the cut value.

1. **Runtime.** Argue that LocalSearch always terminates and give a bound on the number of iterations. (what happens to the value of $\text{cut}(S)$ after each iteration? What is its maximum possible value?) Give a bound on the overall runtime.

2. **Example** Consider the graph with vertices $A = \{a_1, a_2\}$ and $B = \{b_1, b_2, b_3, b_4\}$. Suppose we run Local Search with the starting partition $S = \{a_1, b_1, b_2\}$ and $\bar{S} = \{a_2, b_3, b_4\}$. What is the cut value returned? What if we start with $S = \{a_1, a_2\}$?



3. **Tradeoffs**

   (a) Explain why there is always a starting solution which causes Local Search to return the optimal solution.

   (b) What if our local move space considered swapping pairs of nodes instead of singletons? What about at most k nodes each time?