Local Search Algorithms[1]

- Local Search is a natural and a very commonly used heuristic in practice. It states that given any solution to an optimization problem, if certain simple "local" changes can improve the solution, then one should do it. When no such improvements via local changes are possible, the resulting solution is called a local optimum. In certain situations, it is possible to analyze the local optimum "globally". More precisely, one can prove an approximation factor for the local optimum solution using the fact that local changes do not improve it. In this lecture, we look at some examples of this phenomenon.

- **Max Cut in Undirected Graphs.** In this problem we are given an undirected graph $G = (V, E)$ with non-negative weights $w(e)$ on edges $e \in E$. Given a subset of vertices $S \subseteq V$, the subset $\partial S := \{(u, v) \in E \ : \ u \in S, v \notin S\}$ of edges is called the cut induced by S. The goal is to find a subset $S \subseteq V$ which maximizes $w(\partial S) := \sum_{e \in \partial S} w(e)$, the weight of the cut edges. Unlike the *minimum* cut version, the maximum cut problem is NP-hard. Here is a simple local search algorithm for the same.

procedure UNDIRECTED MAX-CUT LS($G = (V, E)$; $w(e)$ on edges):
  Initialize X to an *arbitrary* subset of V.
  while `true` **do**:
    if $\exists v \in S$ s.t. $w(\partial(S - v)) > w(\partial S)$ **then**:
      $S \leftarrow S - v$.
    else if $\exists v \notin S$ s.t. $w(\partial(S + v)) > w(\partial S)$ **then**:
      $S \leftarrow S + v$.
    else: ▷ *Reached local optimum and terminate loop*
      break.
  return S.

- ***Analysis.*** The following theorem shows that the above local optimal solution is a $\frac{1}{2}$-approximation since any cut contains only a subset of edges and thus $\texttt{opt} \leq w(E)$.

Theorem 1. Let S be the subset obtained at the termination of UNDIRECTED MAX-CUT LS. Then, $w(\partial S) \geq w(E)/2 \geq \texttt{opt}/2$.

Before we go to the proof, let us introduce some useful notations. For any two subsets A and B of vertices, let $E(A : B)$ denote the edges with one endpoint in A and the other in B. In fact, soon we will talk about directed graphs, in which case $E(A : B)$ will be the edges which originate from a vertex in A and land in a vertex in B. In directed graphs, $E(A : B)$ and $E(B : A)$ are different sets; in undirected graphs they are the same. When $A = \{v\}$ is a singleton vertex, we will often abuse notation and lose the curly braces, and simply write $E(v : B)$. For any subset $A \subseteq V$, we use $\overline{A}$ to denote the complement $V \setminus A$. Finally, we use the shorthand $w(A : B)$ to imply $w(E(A : B))$, the weight of the edges in $E(A : B)$.

[1] *Lecture notes by Deeparnab Chakrabarty. Last modified : 8th Jan, 2022*
These have not gone through scrutiny and may contain errors. If you find any, or have any other comments, please email me at deeparnab@dartmouth.edu. Highly appreciated!

Proof. First observe that for any subset $X \subseteq V$, we have

$$w(\partial(X - v)) - w(\partial X) = w(X : v) - w(v : \overline{X}),\ \forall v \in X$$

since in going from ∂X to $\partial(X - v)$, we lose the edges $E(v : \overline{X})$ and gain the edges $E(X : v)$[2]. Similarly,

$$w(\partial(X + v)) - w(\partial X) = w(v : \overline{X}) - w(X : v),\ \forall v \notin X;$$

Therefore, using the fact that S is a local optimum, we can assert

$$\forall s \in S: \quad w(S : s) \leq w(s : \overline{S}); \qquad \forall t \in \overline{S}: \quad w(t : \overline{S}) \leq w(S : t)$$

Adding the above inequalities for all $s \in S$ and $t \in \overline{S}$, we get

$$\sum_{s \in S} w(S : s) + \sum_{t \in \overline{S}} w(t : \overline{S}) \leq \sum_{s \in S} w(\overline{S} : s) + \sum_{t \in \overline{S}} w(S : t) \tag{1}$$

We are almost done. Observe that $\sum_{t \in \overline{S}} w(S : t)$ is precisely the $w(\partial S)$, and since the graph is undirected, so is $\sum_{s \in S} w(\overline{S} : s)$. Therefore, the RHS in (1) is $2w(\partial S)$. Next, observe that $\sum_{s \in S} w(S : s)$ is adding up the weights of edges of the type (s, s') where both end points are in S, and each such edge is counted twice. Therefore, the LHS in(1) is *two times* the weight of all edges which are **not** present in ∂S. Therefore,

$$2\left(w(E) - w(\partial S)\right) \leq 2w(\partial S) \quad \Rightarrow \quad 2w(\partial S) \geq w(E)$$

□

- **Max-Cut in Directed Graphs.** In the Max DiCut problem, we are given a directed graph $G = (V, E)$ with non-negative weights on edges. For a subset S of vertices, $\partial^+ S := \{(u, v) \in E : u \in S, v \notin S\}$ be the directed out-cut of S and $\partial^- S := \{(u, v) \in E : u \notin S, v \in S\}$ be the directed in-cut. The objective is to find $S \subseteq V$ maximizing $w(\partial^+ S) := \sum_{e \in \partial^+ S} w(e)$. The local search algorithm is almost the same with one last twist at the end.

procedure DIRECTED MAX-CUT LS($G = (V, E)$; $w(e)$ on edges):
  Initialize X to an *arbitrary* subset of V.
  while true **do**:
    if $\exists v \in S$ s.t. $w(\partial^+(S - v)) > w(\partial^+ S)$ **then**:
      $S \leftarrow S - v$.
    else if $\exists v \notin S$ s.t. $w(\partial^+(S + v)) > w(\partial^+ S)$ **then**:
      $S \leftarrow S + v$.
    else: ▷ *Reached local optimum and terminate loop*
      break.
  return S or $\overline{S}$, whichever has a larger $w(\partial^+ \cdot)$. ▷ *the twist.*

[2] Technically, we should write $E(X - v : v)$, but what we write is ok since we can assume wlog that the graph has no self-loops.

Exercise: ☕ *Show that the twist is necessary. More precisely, if the algorithm returned S, then it could be very bad solution.*

Exercise: ☕ *Generalize the proof of Theorem 1 and prove that the weight of the local optimum cut returned by* DIRECTED MAX-CUT LS *is at least $w(E)/4$. Also describing an example where this is tight, that is, the weight of the local optimum cut is very close to $w(E)/4$.*

- ***Analysis.*** The exercise above shows that the DIRECTED MAX-CUT LS is a $\frac{1}{4}$-approximation. We now show that it is in fact better; it is a $1/3$-approximation. This may seem contrary to the second part of the above exercise, but note that in that example even the optimal cut's value is "far" from $w(E)$.

Theorem 2. The cut returned by DIRECTED MAX-CUT LS is an $\frac{1}{3}$-approximation.

Proof. Let O be the optimal solution. Note that $\partial^+ O := E(O : \overline{O})$. We are going to divide these edges into four disjoint subsets. We refer the reader to Figure 1 for a useful illustration.

$$E(O:\overline{O}) = E(O\cap S:\overline{O}\cap S) \ \sqcup \ E(O\cap\overline{S}:\overline{O}\cap S) \ \sqcup \ E(O\cap S:\overline{O}\cap\overline{S}) \ \sqcup \ E(O\cap\overline{S}:\overline{O}\cap\overline{S})$$

and thus,

$$\texttt{opt} = w(O\cap S:\overline{O}\cap S) + w(O\cap\overline{S}:\overline{O}\cap S) + w(O\cap S:\overline{O}\cap\overline{S}) + w(O\cap\overline{S}:\overline{O}\cap\overline{S}) \qquad (2)$$

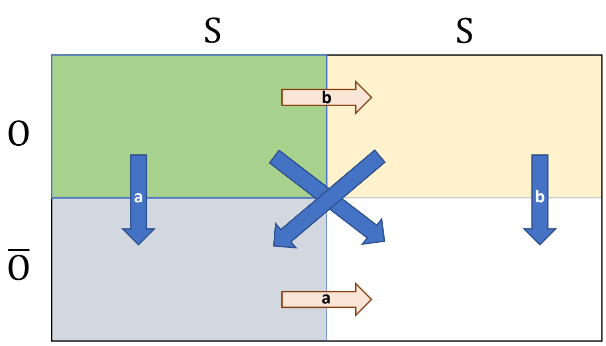

Figure 1: *The rectangle denotes V. The set S is what is vertically below the letter marked in blue, and the set O is what is horizontally to the right of the letter marked in yellow. $O\cap S$ is the green (= blue + yellow) region. The blue edges are those in $\partial^+ O$ divided into four parts. Part marked with "a" is charged to the salmon colored "a" edges in $\partial^+ S$. Similarly, the blue "b" edges are charged to the salmon "b" edges. The cross blue edges are in $\partial^+ S$ or $\partial^+ \overline{S}$.*

Now we use the local optimality conditions. Recall,

$$\forall s \in S: \ \ w(S:s) \le w(s:\overline{S}); \qquad \forall t \in \overline{S}: \ \ w(t:\overline{S}) \le w(S:t)$$

In particular, this is used for the first and fourth sets in (2) as follows

$$\forall s \in \overline{O}\cap S \ : \ w(O\cap S : s) \underbrace{\le}_{\text{since } w\ge 0} w(S:s) \le w(s:\overline{S}) \ \Rightarrow \ w(O\cap S:\overline{O}\cap S) \le w(\overline{O}\cap S:\overline{S}) \underbrace{\le}_{\text{since } w\ge 0} w(S:\overline{S}) \qquad (3)$$

$$\forall t \in O \cap \overline{S} \,:\, w(t : \overline{O} \cap \overline{S}) \underbrace{\leq}_{\text{since } w \geq 0} w(t : \overline{S}) \leq w(S : t) \;\Rightarrow\; w(O \cap \overline{S} : \overline{O} \cap \overline{S}) \leq w(S : O \cap \overline{S}) \quad (4)$$

The second and third term in (2) are upper bounded more simply as

$$w(O \cap \overline{S} : \overline{O} \cap S) \underbrace{\leq}_{\text{since } w \geq 0} w(\overline{S} : S) \quad \text{and} \quad w(O \cap S : \overline{O} \cap \overline{S}) \underbrace{\leq}_{\text{since } w \geq 0} w(S : \overline{O} \cap \overline{S}) \qquad (5)$$

Adding Equations (3) to (5) and substituting in Equation (2), we get

$$\texttt{opt} \leq w(S : \overline{S}) + w(\overline{S} : S) + \underbrace{w(S : \overline{O} \cap \overline{S}) + w(S : O \cap \overline{S})}_{= w(S : \overline{S})} \leq 3\mathsf{alg}$$ □

Exercise: ☕ *Describe an example showing that the analysis above is tight.*

- **Running Time?** Above, we argued that the cut returned by the above algorithms are $\frac{1}{2}$ and $\frac{1}{3}$ approximations, respectively. However, do these algorithms terminate? Well, every time the while loop runs, the weight of the cut strictly goes up. Does that imply termination? Yes. Because, it means the same cut can't be seen again. Therefore, the algorithm terminates in at most 2^n loops. But that is ridiculous for in that time we can in fact find the global optimum. Can we do a better analysis of the running time? The answer is no! In fact, there are weights where the number of loops can be exponential in n.

  What then? Firstly, note that when the weights on the edges are small integers (eg, an unweighted graph), the algorithm does terminate in mW-loops, where $W = \max_{e \in E} w(e)$; this is because the weight of the cut goes up by 1 in each loop. Another thought is that perhaps we can find a locally optimum solution in a *different* way. One doesn't *need* to do local search to find a local optimum. Indeed, this is an *open* question. However, this may not be possible (see Notes for a little more discussion).

  This brings us to a standard idea present in many local search algorithms which face such problems. Note that both the algorithms above perform a "local step" if that step strictly increases the solution. The modification of this idea is : perform the local step only if it leads to a *significant* increase in the solution. More precisely, in the UNDIRECTED MAX-CUT LS algorithm, we replace the if- and else-if conditions with $w(\partial(S - v)) > (1 + \varepsilon)w(\partial S)$ and $w(\partial(S + v)) > (1 + \varepsilon)w(\partial S)$, respectively, for some fixed parameter $\varepsilon > 0$. Then one can show the following, which is left as an exercise for the reader.

  Exercise: ☕☕ *Prove that the modified* UNDIRECTED MAX-CUT LS *(a) terminates in* $O\left(\frac{\log nW}{\varepsilon}\right)$*-while loops, and (b) returns a solution which is* $\left(\frac{1}{2} - \varepsilon\right)$*-approximate. Here* W *is the maximum weight of an edge when all weights are integer valued.*

- **Maximum Colorful Coverage.** For the final example of this lecture, we consider a twist on the maximum coverage problem which we have seen greedy algorithms for. The input is like the set-cover problem. A universe U of n elements, and m subsets $(S_1, \ldots, S_m)$. However, each subset S_i now has a *color* χ_i which is one of k colors. Without loss of generality we assume $\chi_i \in \{1, 2, \ldots, k\}$. We need to pick k sets each of a different color so that the union is as large as possible.

  Here is a local search algorithm.

procedure COLORFUL COVERAGE LS(U, $S_1, \ldots, S_m$, Colors on sets):
  Initialize $\mathcal{S} := (S_1, \ldots, S_k)$ arbitrarily picking one set from each color class.
  ▷ *We use* $\mathsf{val}(\mathcal{S})$ *to denote* $|\cup_{S \in \mathcal{S}} S|$.
  ▷ *We will use* S_c *to denote the set of color* c *in* $\mathcal{S}$.
  while true **do**:
    if $\exists S'_c$ of color c such that $\mathsf{val}(\mathcal{S} - S_c + S'_c) > \mathsf{val}(\mathcal{S})$ **then**:
      $\mathcal{S} \leftarrow \mathcal{S} - S_c + S'_c$
    else:
      break.
  return $\mathcal{S}$.

Theorem 3. COLORFUL COVERAGE LS returns a $\frac{1}{2}$-approximation.

Proof. Let $(S_1, \ldots, S_k)$ be the solution returned by the local search algorithm, and let $(O_1, \ldots, O_k)$ be the optimal solution. We have renamed the sets such that S_i and O_i are of color i. Let $A = \bigcup_{i=1}^{k} S_i$ and let $O = \bigcup_{i=1}^{k} O_i$. Note that $\texttt{opt} = |O|$ and $\texttt{alg} = |A|$.

Let $X = O \setminus A$ be the set of elements that the optimal solution covers but the local search algorithm misses. Let $\theta_i = |O_i \cap X|$ denote the number of these elements that are present in O_i. Note, $|X| \le \sum_{i=1}^{k} \theta_i$. Let $n_i := \{e \in S_i \ : \ e \notin S_j, j \neq i\}$ be the number of elements in A which are *uniquely* covered by S_i. Note that $|A| \ge \sum_{i=1}^{k} n_i$.

Now, $\mathcal{S}' := \mathcal{S} - S_i + O_i$ is a feasible solution of value $\mathsf{val}(\mathcal{S}') = \mathsf{val}(\mathcal{S}) - n_i + \theta_i$. Since $\mathcal{S}$ is a local optimum, we get that $n_i \ge \theta_i$ for $1 \le i \le k$. And so

$$|A| \ge \sum_{i=1}^{k} n_i \ge \sum_{i=1}^{k} \theta_i \ge |X| \ge |O| - |A| \quad \Rightarrow \quad |A| \ge \frac{|O|}{2}$$

□

Notes

The complexity of local search was initiated in the paper [1] by Johnson, Papadimitriou, and Yannakakis. They defined a complexity class called PLS which captures the complexity of finding locally optimum solutions for optimization problems. Soon after, Schaffer and Yannakakis [2] proved that the max-cut problem is PLS-complete. A repercussion is that if one finds a polynomial time algorithm for the problem of finding a local optimum cut, then there exists efficient algorithms for a host of problems (all problems in this class). We refer the reader to the papers for more details.

References

[1] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of computer and system sciences*, 37(1):79–100, 1988.

[2] A. A. Schäffer. Simple local search problems that are hard to solve. *SIAM journal on Computing*, 20(1):56–87, 1991.