Non-Oblivious Local Search[1]

- Usually in a local search algorithm for a certain problem, one moves from a solution to a "neighboring" solution if it improves the cost of the solution. In certain situations, moving to a neighboring solution which improves a different but related function can actually lead to a better approximation factor. This interesting idea is called non-oblivious[2] local search in the literature. We illustrate this using the Max-2SAT problem.

- In the Max-2SAT problem we are given a 2SAT formula. A 2SAT formula ϕ has m clauses on n variables where each clause consists of 2 literals. A literal is a variable or its negation. Given an assignment of the variables to $\{\mathsf{true}, \mathsf{false}\}$, a clause is satisfied if one of the literals is satisfied. The objective is to find an assignment which maximizes the number of satisfied clauses.

  For example, if
  $$\phi = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (x_2 \vee x_3) \wedge (x_1 \vee \overline{x_3})$$
  then the assignment $(x_1, x_2, x_3) = (\mathsf{false}, \mathsf{true}, \mathsf{true})$ satisfies the second and third clause out of the above 4. On the other hand $(\mathsf{true}, \mathsf{false}, \mathsf{true})$ satisfies all 4 clauses.

  It is known that there is a linear time algorithm to test whether a 2SAT formula is satisfiable or not, that is, checking if $\texttt{opt} = m$ or not. However, the Max-2SAT problem is NP-hard. Below is a (natural) local search algorithm for the same.

- **A Local Search Algorithm.**

```
procedure MAX-2SAT LOCAL SEARCH(2SAT formula φ):
    Begin with an arbitrary assignment of the variables x = (x_1, ..., x_n).
    while true do:
        If there exists a variable x_i such that swapping its value increases the number of satisfied clauses, do so.
        Otherwise break.
    return x.
```

Theorem 1. MAX-2SAT LOCAL SEARCH returns a $\frac{2}{3}$-approximation.

Proof. In fact, we will show something stronger : the local optimal assignment satisfies $\geq \frac{2m}{3}$ clauses.

A clause is satisfied if one or both its literals evaluate to true. Given a variable x_i, let A_i be the clauses c such that c currently evaluates to true only because of x_i. More precisely, if x_i is currently true, then $A_i = \{(x_i \vee \beta) : \beta \text{ eval. to } \mathsf{false}\}$. If x_i is currently false, then $A_i = \{(\overline{x}_i \vee \beta) : \beta \text{ eval. to } \mathsf{false}\}$.

[1] *Lecture notes by Deeparnab Chakrabarty. Last modified : 8th January, 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!

[2] I am not sure why this name is used. Non-obvious local search would probably be a better choice.

Note that, by definition, for two different variables x_i, x_j, we have $A_i \cap A_j = \emptyset$. Therefore, if alg denotes the number of satisfied clauses, we have

$$\mathsf{alg} \geq \sum_{i=1}^{n} |A_i| \tag{1}$$

Next, let B_i be the clauses c such that c contains x_i but c evaluates to false. That is, if x_i is currently true, then $B_i = \{(\overline{x_i} \vee \beta) : \beta \text{ eval. to } \mathsf{false}\}$. If x_i is currently false, then $B_i = \{(x_i \vee \beta) : \beta \text{ eval. to } \mathsf{false}\}$

When we flip the assignment of x_i, the clauses in A_i become unsatisfied and the clauses in B_i become satisfied. All remaining clauses retain their state. Local optimality implies

$$\forall 1 \leq i \leq n \ : \ |A_i| \geq |B_i| \tag{2}$$

Now consider a clause $c = (\alpha \vee \beta)$ which is *not* satisfied by $\mathbf{x}$. Note that c is in precisely two B_i's one corresponding to α and one corresponding to β. For instance if $c = (x_1 \vee \overline{x_2})$, then $c \in B_1$ and $c \in B_2$. Therefore, we have

$$2 \cdot (m - \mathsf{alg}) = \sum_{i=1}^{n} |B_i| \underbrace{\leq}_{(2)} \sum_{i=1}^{n} |A_i| \underbrace{\leq}_{(1)} \mathsf{alg} \quad \Rightarrow \quad \mathsf{alg} \geq \frac{2m}{3}$$

□

- **A Non-Ob(li)vious Local Search Algorithm.** Given an assignment $\mathbf{x}$ of the variables, define the following quantities. $n_0(\mathbf{x})$ counts the clauses that have both literals negated by $\mathbf{x}$. $n_1(\mathbf{x})$ counts the clauses that have exactly one literal negated by $\mathbf{x}$. $n_2(\mathbf{x})$ counts the clauses which have none of the literals negated by $\mathbf{x}$. Note that the "value" of $\mathbf{x}$, that is the number of clauses satisfied by $\mathbf{x}$ is precisely $\mathsf{val}(\mathbf{x}) = n_1(\mathbf{x}) + n_2(\mathbf{x})$. A different way of stating the local search algorithm from the previous bullet point was : flip a variable if it increases $\mathsf{val}(\mathbf{x})$.

  The non-oblivious local search flips a variable if it increases a (slightly) different function of $\mathbf{x}$. Define

$$\Phi(\mathbf{x}) = \frac{4}{3} \cdot n_2(\mathbf{x}) + n_1(\mathbf{x})$$

```
procedure MAX-2SAT NONOB LS(2SAT formula φ):
    Begin with an arbitrary assignment x of the variables
    while true do:
        If there exists a variable x_i such that swapping its value increases Φ(x)
        Otherwise break.
    return x.
```

Theorem 2. MAX-2SAT NONOB LS returns a $\frac{3}{4}$-approximation.

Proof. Indeed, we show that the final $\mathbf{x}$ satisfies $\frac{3m}{4}$ clauses.

Fix a variable x_i and partition the clauses containing x_i into four sets.

- A_i are the clauses where both x_i and the other literal evaluate to true.
- B_i are the clauses where x_i evaluates to true but the other literal evaluates to false.
- C_i are the clauses where x_i evaluates to false but the other literal evaluates to true.
- D_i are the clauses where both x_i and the other literal evaluate to false.

Now, when we flip the assignment of x_i to get the assignment $\mathbf{x}'$, then the change in potential can be described by the four sets above

$$\Phi(\mathbf{x}) - \Phi(\mathbf{x}') = \left(\frac{4}{3} \cdot |A_i| - |A_i|\right) + |B_i| + \left(|C_i| - \frac{4}{3} \cdot |C_i|\right) - |D_i|$$

Since $\mathbf{x}$, is locally optimal, we get that the above RHS is ≥ 0 for all i. Therefore,

$$\forall 1 \leq i \leq n, \quad \frac{|A_i| - |C_i|}{3} + |B_i| \geq |D_i| \tag{3}$$

We now make three more observations, and then the rest would be arithemetic. Let alg be the value of $\mathbf{x}$. Note, $\mathsf{alg} = n_1(\mathbf{x}) + n_2(\mathbf{x})$.

- $\sum_{i=1}^n |D_i| = 2(m - \mathsf{alg})$, since every unsatisfied clause lies in precisely 2 different D_i's.
- $\sum_{i=1}^n |B_i| = \sum_{i=1}^n |C_i| = n_1(\mathbf{x})$.
- $\sum_{i=1}^n |A_i| = 2n_2(\mathbf{x})$.

Therefore, if we add (3) for all $1 \leq i \leq n$, we get

$$\underbrace{\frac{2}{3} \cdot n_2(\mathbf{x}) + \frac{2}{3} n_1(\mathbf{x})}_{= \frac{2\mathsf{alg}}{3}} \geq 2 \cdot (m - \mathsf{alg}) \quad \Rightarrow \quad \mathsf{alg} \geq \frac{3m}{4}$$ □

Notes

The non-oblivious local search algorithm above (and the term itself) is from the paper [4] by Khanna, Motwani, Sudan, and Vazirani. This paper considers other examples where choosing a different function to move locally can help. There are not too many examples in the literature of non-oblivious local search algorithms. A paper [1] studies this in the context of graph and hypergraph coloring, although the paper itself is not easy to locate. A more recent famous example is that of maximizing a monotone submodular function $f(S)$ subject to the constraint that S is an independent set in a matroid. The paper [3] by Filmus and Ward give a non-oblivious local search algorithm whose locality ratio is $(1 - 1/e)$. We point the reader to Ward's thesis [5] for more details. Finally, we mention a very new result [2] by Cohen-Addad, Gupta, Hu, Oh, and Saulpic giving a non-oblivious local search for k-median.

References

[1] P. Alimonti. Non-oblivious local search for graph and hypergraph coloring problems. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 167–180, 1995.

[2] V. Cohen-Addad, A. Gupta, L. Hu, H. Oh, and D. Saulpic. An improved local search algorithm for k-median. *arXiv preprint arXiv:2111.04589*, 2021. To appear in SODA 2022.

[3] Y. Filmus and J. Ward. Monotone submodular maximization over a matroid via non-oblivious local search. *SIAM Journal on Computing*, 43(2):514–542, 2014.

[4] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing (SICOMP)*, 28(1):164–191, 1998.

[5] J. Ward. *Oblivious and non-oblivious local search for combinatorial optimization*. University of Toronto (Canada), 2012.