On Finding Locally Optimal Solutions
(extended abstract)

Mark W. Krentel*

Computer Science Department
Rice University, Houston, Texas 77251-1892

Abstract

We consider the problem of finding locally optimal solutions to combinatorial problems in the framework of PLS as defined in [JPY88]. We exhibit a PLS-complete problem such that the problem of verifying local optimality can be solved in LOGSPACE. For all previously known PLS-complete problems, verifying local optimality was P-complete, and it was conjectured in [JPY88] that this was necessary.

1 Introduction

A recent approach to uncovering the structure of NP-complete problems is the question of finding locally optimal solutions. Of course, if P ≠ NP then it is asking too much to find globally optimal solutions quickly, but the question of local optimality is much less clear.

To formalize this notion, Johnson, Papadimitriou and Yannakakis [JPY88] defined a class of local search problems called PLS (Polynomial-Time Local Search). A PLS problem consists of a set of instances, feasible solutions, a cost function and a neighborhood structure. A feasible solution is locally optimal if it has no neighbor with better cost. For example, consider the GRAPH PARTITIONING problem with the 2-opt neighborhood structure. Instances are undirected graphs with an even number of vertices and weights on the edges. A feasible solution is a partition of the vertices into two equal-size pieces, and the cost function (which we are trying to minimize) is the sum of the weights on the edges whose endpoints are in opposite sides of the partition. The 2-opt neighborhood structure consists of those partitions obtained by swapping one node from each side.

It is an open question if even locally optimal solutions for GRAPH PARTITIONING under 2-opt can be found in polynomial time. This problem is especially intriguing because it seems so natural that you should be able to find at least locally optimal solutions, yet it is not known. As a partial answer to this problem, we turn to completeness. A is PLS-reducible to B if we can transform instances of A into instances of B such that given a local optimum for B, we can construct a local optimum for A. [JPY88] then shows that GRAPH PARTITIONING under the (much more complicated) Kernighan-Lin neighborhood structure [KL72] is PLS-complete. This result has two surprising corollaries. First, it is NP-hard to determine the output of the Kernighan-Lin algorithm on an arbitrary instance, and secondly, there are instances of GRAPH PARTITIONING for which Kernighan-Lin takes exponential time. Naturally, the completeness result also

*Supported in part by NSF grant CCR-8809370.
implies that local optima can be found under the Kernighan-Lin neighborhood structure only if local optima can be found for all local search problems is PLS. This result is especially important because, in practice, the Kernighan-Lin algorithm produces among the best feasible solutions of any heuristic algorithm [JAMS87].

An essential feature of the construction in [JPY88] is that, given a particular feasible solution, it is P-complete to verify that this solution is locally optimal. The 2-opt neighborhood does not have this property; in fact, for 2-opt, local optimality can be verified in LOGSPACE. Moreover, it is not known if the 2-opt neighborhood is also PLS-complete, and thus there is more hope that local optima can be found. Johnson, Papadimitriou and Yannakakis conjectured that a problem could not be PLS-complete without the corresponding verification problem being P-complete. Actually, a lovely conjecture would be that local optima can be found in polynomial time if and only if the verification problem is in LOGSPACE; and, in fact, this was our original motivation, to determine under what conditions local optima can be found in polynomial time.

In this paper, we disprove the above conjecture (under the assumption that $P \neq \text{LOGSPACE}$) by exhibiting a PLS-complete problem such that the corresponding verification problem is in LOGSPACE. The problem we consider is WEIGHTED CNF SATISFIABILITY, i.e., given a boolean formula with (binary) weights on the clauses, try to maximize the sum of the weight on the true clauses, under the neighborhood structure of flipping the value of a single variable. [JPY88] show that the same problem using boolean circuits (CIRCUIT FLIP), instead of formulas in CNF, is PLS-complete and that its verification problem is P-complete. The significance of this result is that it makes it even more unlikely that local optima can be found in polynomial time. We conjecture that GRAPH PARTITIONING under 2-opt is PLS-complete, but as of now, our result doesn’t carry over. We leave it as an open problem under what conditions local optima can be found in polynomial time.

2 Main Result

The definitions below of PLS, and later PLS-reducible, are designed to formalize the question of when it is possible to find locally optimal solutions in polynomial time. Notice that there are no constraints on the number of neighbors, or on the diameter (the minimum distance between two feasible solutions, maximized over all pairs of feasible solutions) of the neighborhood structure, or even that the neighborhood relation is connected or symmetric. It would be valid to have every feasible solution a neighbor of every other feasible solution, but then algorithm $C^\Pi$ would imply that verifying global optimality is in polynomial time. Also notice that $C^\Pi$ is not required to compute the best neighbor.

**Definition** A PLS (Polynomial-Time Local Search) problem, $\Pi$, consists of the following.

1. A set of instances $D^\Pi \subseteq \Sigma^*$.
2. A set of feasible solutions $FS^\Pi(x) \subseteq \Sigma^{p(|x|)}$ for every $x \in D^\Pi$, for some polynomial $p$.
3. A set of neighbors $N^\Pi_x(s) \subseteq FS^\Pi(x)$ for every $x \in D^\Pi$ and $s \in FS^\Pi(x)$.
4. A measure function $m^\Pi_x : FS^\Pi(x) \rightarrow N$ for every $x \in D^\Pi$, where $N = \{0, 1, 2, \ldots\}$, the set of natural numbers.
We require $D^\Pi$, $FS^\Pi(x)$, $N^\Pi_i(s)$ and $m^\Pi_i$ to be polynomial-time computable. In addition, we require the following three algorithms to be computable in polynomial time.

1. Algorithm $A^\Pi$, on input $x \in D^\Pi$, produces an initial feasible solution in $FS^\Pi(x)$.

2. Algorithm $B^\Pi$, on input $x \in D^\Pi$ and $s \in FS^\Pi(x)$, computes $m^\Pi_i(s)$.

3. Algorithm $C^\Pi$, on input $x \in D^\Pi$ and $s \in FS^\Pi(x)$, determines if $s$ is locally optimal, and if not produces another solution $s' \in N^\Pi_i(s)$ with better cost.

**Definition** Circuit Flip. Instances are boolean circuits with $n$ inputs and $n$ outputs. A feasible solution is an assignment to the inputs, and the measure function (which we are trying to maximize) is the output sequence viewed as a binary number. The neighborhood of an assignment contains all other assignments obtained by flipping the value of a single input.

**Definition** Weighted CNF Satisfiability. Instances are CNF boolean formulas with (binary) weights on the clauses. A feasible solution is an assignment to the variables, and the measure function (again, maximization) is the sum of the weights on the true clauses. The neighborhood is again, all other assignments obtained by flipping the value of a single input.

**Definition** $A$ is PLS-reducible to $B$ if there are polynomial-time computable functions $f$ and $g$ such that $f$ maps instances of $A$ to instances of $B$, and $g$ given $x \in D^A$ and a locally optimal $y \in FS^B(f(x))$, produces a locally optimal $g(x, y) \in FS^A(x)$.

**Theorem 2.1** [JPY88] Circuit Flip is PLS-complete, and its corresponding verification problem is P-complete.

**Theorem 2.2** The problem of verifying local optimality for Weighted CNF Satisfiability is in LOGSPACE.

**Proof:** Addition of $n$-bit numbers can be computed in LOGSPACE by recomputing the result bit by bit. The rest is straightforward. \( \square \)

**Theorem 2.3** Weighted CNF Satisfiability is PLS-complete.

**Proof:** We reduce the Circuit Flip problem to Weighted CNF Satisfiability. Let $C$ be a circuit with inputs $X_1, \ldots, X_n$, outputs $Y_1, \ldots, Y_n$ and gates $G_1, \ldots, G_M$. Assume for convenience that $M \geq n$ and that the first $n$ gates, $G_1, \ldots, G_n$, just copy the input and that these are the only gates that $X_1, \ldots, X_n$ appear in. We also assume that the gates are topologically ordered, so that if $G_i$ is an input to $G_j$, then $i < j$.

We reduce $C$ to a CNF boolean formula $\Phi$ with variables $x_1, \ldots, x_n, f_1, \ldots, f_n, g_1, \ldots, g_M$ and $t_1^i, \ldots, t_M^i$ for each $1 \leq i \leq n$. Obviously, the variables $x_1, \ldots, x_n$ represent the input to the circuit, and $g_1, \ldots, g_M$ represent the output of the gates. For each input $x_i$, the variables $t_1^i, \ldots, t_M^i$ (called test circuits) represent the new output of $C$'s gates if we were to flip the value of $x_i$. Since $G_1, \ldots, G_n$ just copy the input, in general we want $g_j = t_j^i = x_j$ for $1 \leq j \leq n$, except that we want $t_1^i = \overline{x}_i$. The variables $f_1, \ldots, f_n$ are used to flip the inputs $x_1, \ldots, x_n$. Normally all $f_i$ will be 0, but when we are ready to flip $x_i$, we first set $f_i$ to 1 and then flip $x_i$ and reset $g_1, \ldots, g_M$, and then reset $f_i$ to 0. This allows us to recompute
the circuit's new output from \( t_1^\prime, \ldots, t_M^\prime \) with a single flip of \( f_1 \), and will be a key point in the construction.

We divide the clauses into three classes of constraints: hard constraints, medium weights and small weights. The hard constraints have the heaviest weight, and it will turn out that if any hard constraint is violated, then it will always be possible to improve their weight by flipping a single variable. Thus, a feasible solution must satisfy the hard constraints in order to have any hope of being locally optimal. The medium weights represent the output of the circuit, and the small weights give a credit for setting the test circuits correctly.

**Hard Constraints** The largest constraint is that at most one \( f_i \) can be set to 1. This can be expressed as

\[
\bigwedge_{1 \leq i < j \leq n} (\overline{t_i} + t_j)^{2^M}
\]

where \((\psi)^w\) means that all clauses in \( \psi \) have weight \( w \).

The second hard constraint is that the output of the circuit is computed correctly. If all \( f_i = 0 \), then we use \( g_1, \ldots, g_M \) to compute the circuit, and if some \( f_i = 1 \) then we use \( t_1^\prime, \ldots, t_M^\prime \). Thus, the constraint we wish to express is

\[
(g_{\text{gate 1 correct}})^{2^M} \cdots (g_{\text{gate M correct}})^{2^M+1}
\]

where "gate \( j \) correct" means

\[
\left( (\bigvee f_i = 0) \Rightarrow (g_j \text{ correct}) \right)
\]

\[
\land \left( \bigwedge_{1 \leq i < n, i \neq j} (f_i = 1) \Rightarrow (t_j^\prime \text{ correct}) \right)
\]

For example, consider the gate "\( g_j = g_a \land g_b \)." This can be expressed in CNF as

\[
(g_j + \overline{g_a} + \overline{g_b})(\overline{g_j} + g_a + g_b)(\overline{g_j} + g_a)
\]

So, the constraint "gate \( j \) correct" can be expressed

\[
\begin{align*}
(f_1 + \cdots + f_n + g_j + \overline{g_a} + \overline{g_b}) \\
(f_1 + \cdots + f_n + \overline{g_j} + g_a + g_b) \\
(f_1 + \cdots + f_n + \overline{g_j} + \overline{g_a} + g_b) \\
\land \left( \bigwedge_{1 \leq i \leq n, i \neq j} (\overline{f_i} + t_j^\prime + \overline{t_i} + t_i^\prime) \right)
\end{align*}
\]

A similar construction is possible for or and not gates. Notice that there is no constraint that \( t_i^\prime = x_i \). This will be necessary for allowing \( x_i \) to be flipped at the appropriate time.

**Medium Weights** The next largest constraints, the medium weights, represent the outputs \( Y_1, \ldots, Y_n \). Recall that the output of the circuit is represented in a subset of the gates and that the output is taken from \( g_1, \ldots, g_M \) if \( \bigvee f_i = 0 \), or from \( t_1^\prime, \ldots, t_M^\prime \) if \( f_i = 1 \). These weights are computed similarly to the constraints that the gates are set correctly and can be expressed

\[
(\text{output } 1 \text{ is } 1)^{2^{M+1}} \cdots (\text{output } n \text{ is } 1)^{2^{M+1}}
\]

For example, if \( g_j \) is an output gate, "output \( j \) is 1" is expressed as

\[
(f_1 + \cdots + f_n + g_j) \land \left( \bigwedge_{1 \leq i \leq n} (\overline{f_i} + t_j^\prime) \right)
\]

**Small Weights** There are three groups of small weights. The largest gives a credit for setting \( x_i = t_i^\prime \) when \( f_i = 1 \) and can be expressed as

\[
\bigwedge_{1 \leq i \leq n} (\overline{f_i} + \overline{x_i} + t_i^\prime)^{2^M} (f_i + x_i + t_i^\prime)^{2^M}
\]

The second group gives a credit for setting \( f_i = 0 \) and can be expressed as

\[
(\overline{f_1})^{2^M} \cdots (\overline{f_n})^{2^M}
\]
The third and smallest group gives a credit for setting the values of the test circuits correctly, except that here, "$t_i$ correct" means $t_i = \overline{z_i}$. This can be expressed as

$$\left((g_1 \text{ correct})^{2^M} \cdots (g_M \text{ correct})^{2^1}\right) \land \left(\bigwedge_{1 \leq i \leq n} (t_i^1 \text{ correct})^{2^M} \cdots (t_i^M \text{ correct})^{2^1}\right)$$

(10)

where "$g_j$ correct" is expressed as in (4). Then $\Phi$ is the product of the clauses in (1), (2), (6), (8), (9) and (10).

Notice that the weights are constructed in descending powers of two. This technique makes it easier to argue about the structure of a local optimum because a higher power of two will completely dominate all of the lower powers. Another point is that the CNF constraints in (1), (3), (4), (5), (7), (8) and (9) are carefully constructed so that if a constraint is violated, then exactly one of the clauses is unsatisfied. Although it may take several clauses to express one constraint, the difference between satisfying the constraint and violating it is just the weight on a single clause. Thus, we may pretend that each constraint is expressed by a single clause.

**Claim** Let $B$ be a locally optimal assignment for $\Phi$. Then $B$ corresponds to a locally optimal solution for $C$.

First we claim that at most one $f_i = 1$ in $B$. If not, then the weight from (1) can be improved by a single flip of some $f_i$ to 0.

Secondly, we claim that the gates representing the circuit, either $g_1, \ldots, g_M$ or $t_1^1, \ldots, t_M^M$, are correct. Again, if not, the weight from (2) can be improved by a single flip of the incorrect gate; and again, this weight dominates any other weight that could be lost. Therefore, $B$ must satisfy the hard constraints. This implies that $B$ corresponds to some feasible solution, $A$, for $C$, and also that the output of $A$ can be determined from the weight of $B$. (Although there is no constraint that $t_i^i = z_i$ if $f_i = 1$, use $t_i^i$ for the value of input $i$ in $A$. Later we will see that all $f_i$ must be 0 in $B$ anyway.)

The remainder of the proof splits depending on whether or not some $f_i = 1$. Suppose first that all $f_i = 0$; later we will see that the second case is actually not possible. Then we already know that $g_1, \ldots, g_M$ are correct. Suppose that some $t_j$ is incorrect (including the case that $t_j$ is incorrect, meaning $t_j^i \neq \overline{z_i}$). Because all $f_i = 0$, the test gates play no part in the hard constraints, the medium weights or in the largest of the small weights. Thus, flipping $t_j$ improves the weight of $B$ from (10) and it does not destroy any of the heavier weights. So, the test gates must be correct. This says that the new output of flipping any one of the inputs is available in $B$ by the single flip of some $f_i = 1$. Because the test gates are correct, flipping $f_i = 1$ would not violate the hard constraints, but it would represent the new medium weights. Thus, if $B$ is locally optimal, then flipping one input cannot improve the output in $C$, and thus $A$ is locally optimal for $C$.

Now suppose that $B$ is locally optimal and that $f_i = 1$ for some $1 \leq i \leq n$. From the hard constraints, we already know that $t_1^1, \ldots, t_M^M$ are correct and reflect the circuit's output. Here it is $g_1, \ldots, g_M$ that play no role in the hard constraints or the medium weights. Firstly, $f_i = 1$ implies that $z_i$ does not affect the hard constraint for $t_i^i$ correct. So, if $z_i \neq t_i^i$, then we can improve $B$'s weight by flipping $x_i = t_i^i$, because the weight of (8) dominates the weight of any other clause containing $x_i$. Therefore, $x_i = t_i^i$ in $B$. Then, if any $g_i$ is incorrect, we can improve
$B$'s weight by flipping the incorrect gate because $g_1, \ldots, g_M$ doesn't affect the hard constraints as long as $f_i = 1$. Thus, $g_1, \ldots, g_M$ must be correct, and then $x_i = t_i^1$ implies that $g_1, \ldots, g_M$ must have the identical values as $t_1^1, \ldots, t_M^1$. But then, flipping $f_i$ to 0 preserves the hard constraints and the medium weights and also picks up the weight from (9). This implies that $B$ is not locally optimal; and, in fact, all $f_i = 0$ in any locally optimal solution for $\Phi$. This completes the proof. □

3 An Update

After submitting this paper, substantial progress has been made on this problem. First, Schäffer and Yannakakis [SY89] extended the result for weighted CNF satisfiability to clauses of bounded size. Then, they improved their result to clauses of size 2 and also to not-all-equal 3-SAT (a special case of weighted 2-CNF satisfiability). A corollary of this result is that graph partitioning is PLS-complete under the 2-opt neighborhood structure.

Inspired by this result, the author extended the result for weighted CNF satisfiability to variables with a bounded number of occurrences and also to simultaneous bounded-size clauses and bounded number of occurrences of variables. The significance of this result is that it has implications for traveling salesman. In particular, the author has a preliminary sketch of a construction to show that traveling salesman is PLS-complete under the $k$-opt neighborhood structure for a fixed (and currently very large) value of $k$.

References


[SY89] Alejandro Schäffer and Mihalis Yannakakis. PLS reduction from weighted sat to weighted not all equal 3 sat. Preliminary manuscript, 1989.