Greed Is Slow on Sparse Graphs of Oriented Valued Constraints

Artem Kaznatcheev
Department of Mathematics, and Department of Information and Computing Sciences, Utrecht University, The Netherlands

Sofia Vazquez Alferez
Department of Mathematics, and Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract

Greedy local search is especially popular for solving valued constraint satisfaction problems (VCSPs). Since any method will be slow for some VCSPs, we ask: what is the simplest VCSP on which greedy local search is slow? We construct a VCSP on $6n$ Boolean variables for which greedy local search takes $7(2^n - 1)$ steps to find the unique peak. Our VCSP is simple in two ways. First, it is very sparse: its constraint graph has pathwidth 2 and maximum degree 3. This is the simplest VCSP on which some local search could be slow. Second, it is "oriented" – there is an ordering on the variables such that later variables are conditionally-independent of earlier ones. Being oriented allows many non-greedy local search methods to find the unique peak in a quadratic number of steps. Thus, we conclude that – among local search methods – greed is particularly slow.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming

Keywords and phrases valued constraint satisfaction problem, local search, algorithm analysis, constraint graphs

Digital Object Identifier 10.4230/LIPIcs.CP.2025.18

Related Version *Full Version*: `https://arxiv.org/abs/2506.11662`

Acknowledgements We want to thank Martin C. Cooper for his comments which greatly improved the quality of our writing and Melle van Marle for his insightful discussions.

1 Introduction

We cannot hope for a polynomial time algorithm to solve arbitrary combinatorial optimization problems. But we still need to try to do something to solve these hard problems. Given that many hard problems can be defined as finding an assignment $x \in \{0,1\}^n$ that maximizes some pseudo-Boolean function – that we call a *fitness function* – many turn to local search as a heuristic method [1]. Local search methods start at an initial assignment, then apply a fixed update rule to select a better adjacent assignment to move to until no further improvement is possible. Such a sequence of adjacent improving assignments is called an ascent. One of the most popular update rules is to always select the adjacent assignment that increases fitness by the largest amount, that is, to follow the steepest ascent– this update rule defines greedy local search [22]. Since greed is just one of many possible update rules, a natural question arises: is greed good? Or more specifically: is greedy local search fast or slow compared to other local search methods?

Since no local search method will be able to solve *all* instances of hard combinatorial optimization problems in a polynomial number of steps, we need a more nuanced criterium for what makes a local search method fast or slow. We find this nuance in looking at the performance of our methods on subsets of instances of differing simplicity instead of on all instances. Now, if a method cannot solve particularly simple sets of instances in a polynomial number of steps – at least when compared to other similar methods – then we call it slow.

To define our set of all instances and refine that to simple instances, we turn to valued constraint satisfaction problems (VCSPs). In the language of constraint programming, finding the maximum of a pseudo-Boolean function is the same as solving a Boolean VCSP. Given that even binary Boolean VCSPs can express both NP-hard and PLS-complete problems [3, 11], we define – in Section 2 – our set of all instances as the set of all binary Boolean VCSPs. Each binary VCSP can be interpreted as defining a constraint graph with an edge between any two variables that share a constraint. This allows us to use the sparsity of the resulting graphs as a measure of simplicity: we focus on sets of instances of bounded degree and of bounded treewidth or – more restrictively – pathwidth.

In terms of degree, Elsässer and Tscheuschner [8] showed that binary Boolean VCSPs are PLS-complete under tight reductions even if the constraints are restricted to MAXCUT. The tight PLS-reduction means that there are degree 5 instances where all ascents are exponential – including the steepest ascent taken by greedy local search. Monien and Tscheuschner [17] constructed a family of instances of degree 4 for which they claim all ascents are exponential. Finally, all ascents are quadratic when the constraint graph has maximum degree 2 (Theorem 5.6 in Kaznatcheev [13]). Taken together, this only leaves open the question of efficiency of greedy local search for binary Boolean VCSPs of degree 3.

Treewidth as a sparsity parameter has a more complicated – and perhaps more interesting – relationship to the efficiency of local search. There exists a polynomial time non-local search algorithm for finding not just a local maximum but the global maximum for VCSPs of bounded treewidth [2, 4]. Thus, binary Boolean VCSPs of bounded treewidth are not hard for PLS. Unlike bounded degree, bounded treewidth instances really are a class of simple instances. But the existence of a polynomial-time non-local search algorithm for solving bounded treewidth VCSPs, does not mean that local search will be efficient at finding even a local peak. Cohen et al. [5] have already shown that greedy local search requires an exponential number of steps for Boolean VCSPs of pathwidth 7. The simplest set of instances on which greedy local search fails was subsequently lowered to pathwidth 4 [15]. Recently, Kaznatcheev and Vazquez Alferez [16] showed that an old construction of Hopfield networks by Haken and Luby [9] provides a binary Boolean VCSP of pathwidth 3 on which greedy local search takes an exponential number of steps. Since Kaznatcheev, Cohen and Jeavons [14] have shown that *all* local search methods will take at most a quadratic number of steps on tree-structured binary Boolean VCSPs (i.e., treewidth 1), this leaves only the case of treewidth 2 as open for the question of whether greedy local search is fast or slow.

There are partial results for these two open questions on degree and treewidth – mostly focused on *all* or *some* ascents, rather than *steepest* ascents – but they cannot be further extended. Poljak [18] showed that if the VCSP instance is allowed only MAXCUT constraints then for degree 3 instances, all ascents are quadratic. However, Poljak's [18] technique of rewriting degree 3 MAXCUT constrain graphs by instances with small weights cannot extend to arbitrary binary constraints. There are degree 3 VCSP-instances that require exponentially large weights (Example 5.10 in Kaznatcheev [13]) and, in their Example 7.2, Kaznatcheev, Cohen and Jeavons [14] gave an explicit family of instances with max degree 3 where some ascents are exponential. Similarly, the encouragement-path proof techniques for showing that all local search methods are efficient on VCSPs of treewidth 1 [14] cannot be extended to treewidth 2. Specifically, Kaznatcheev, Cohen and Jeavons [14]'s Example 7.2 not only has maximum degree 3 but also pathwidth 2. However, although some ascents are exponentially long in this family of instances, the steepest ascents are all short and greedy local search can find the peak in a linear number of steps. More generally, Kaznatcheev and van Marle [15] optimistically conjectured that on *any* family of VCSP-instances of pathwidth 2, greedy local search will take at most a polynomial number of steps.

In this paper, we resolve these two open questions on the efficiency of greedy local search for VCSP-instances of degree 3 and treewidth 2.[1] In Section 3, we construct VCSP-instances $\mathcal{C}^{\pm}_{n,\leq n}$ on $6n$ Boolean variables with $7n-1$ binary constraints and $6n$ unary constraints with a constraint graph of maximum degree 3 and pathwidth 2 (Proposition 3). In Section 4, we show that greedy local search takes $7(2^n-1)$ steps to solve these simple VCSP-instances (Theorem 6). We construct $\mathcal{C}^{\pm}_{n,\leq m}$ as a path of m gadgets $\mathcal{C}^{\pm}_{n,m}, \mathcal{C}^{-}_{n,m-1}, \ldots, \mathcal{C}^{-}_{n,2}, \mathcal{C}^{-}_{n,1}$ with consecutive gadgets joined by a single binary constraint. Our proof of long steepest ascents is by induction on the number of gadgets in the path. We show that the steepest ascent first flips bits only in the first gadget $\mathcal{C}^{\pm}_{n,m}$ until it flips the variable that participates in the binary constraint linking $\mathcal{C}^{\pm}_{n,m}$ to $\mathcal{C}^{-}_{n,m-1}$. When this linking variable flips to 1 this gives us the inductive hypothesis of $\mathcal{C}^{+}_{n,\leq m-1}$ and when it flips to 0, it gives the inductive hypothesis of $\mathcal{C}^{-}_{n,\leq m-1}$. The constraint weights in the gadget are chosen in such a way that after this linking variable is flipped, the next potential flip in $\mathcal{C}^{\pm}_{n,m}$ increases fitness by a lower amount than the flips in $\mathcal{C}^{\pm}_{n,\leq m-1}$. Thus the steepest ascent continues in the part of the VCSP corresponding to the inductive hypothesis for $7(2^{m-1}-1)$ steps, until it reaches the local peak of the sub-instance and finally allows flips to continue in $\mathcal{C}^{\pm}_{n,m}$ that eventually result in the linking variable flipping back, repeating the recursive process for a second time.

We also show that our VCSP-instances are what Kaznatcheev and Vazquez Alferez [16] defined as "oriented" (Proposition 4). This means that the exponential running time of greedy local search on our instances stands in stark contrast to many other non-greedy local search methods that Kaznatcheev and Vazquez Alferez [16] have shown to take a quadratic or fewer steps to solve oriented VCSPs. From this we conclude that among local search methods, greed is particularly slow.

2 Background

Given some finite set V of variable indexes with size $|V| = d$,[2] an *assignment* is a d-dimensional Boolean vector $x \in \{0,1\}^d$. For every $i \in V$, x_i refers to the ith entry of x. To refer to a substring with variable indexes $S \subseteq V$, we write the partial assignment $x[S] \in \{0,1\}^S$. To complete a partial assignment, we write $x \in \{0,1\}^S y[V \setminus S]$ to mean that $x_i = y_i$ for $i \in V \setminus S$ and free otherwise. When assignments are sufficiently short we give x and y explicitly, for example, $x_1 0 x_3$ refers to an assignment to 3 variables where the second variable is set to 0. If we want to change just the assignment x_i at index i to a value $b \in \{0,1\}$ then we will write $x[i:b]$. Two assignments are *adjacent* if they differ on a single variable. Or more formally: $x, y \in \{0,1\}^d$ are adjacent if there exists an index $i \in [d]$ such that $y = x[i : \overline{x_i}]$, where $\overline{x_i} = 1 - x_i$.

A *fitness landscape* is any pseudo-Boolean function $f : \{0,1\}^V \to \mathbb{Z}$ together with the above notion of adjacent assignments. Given any $S \subseteq V$, the sublandscape on S given background $y \in \{0,1\}^{V \setminus S}$ is the function f restricted to inputs $x \in \{0,1\}^S y[V \setminus S]$. An

[1] In parallel work to ours, van Marle [21] developed a similar construction of exponential steepest ascent. His construction has the same constraint graph – with degree 3 and pathwidth 2 – but slightly different weights. The key difference is a much more involved proof of exponential steepest ascents: van Marle [21] shows how to simulate a particular prior construction of *some* exponential ascent as a *steepest* ascent on a "padded" instance. This is similar to the prior approaches taken by Cohen et al. [5] and Kaznatcheev and van Marle [15]. In contrast, we focus on how to compose individual gadget's fitness landscapes and their steepest (partial) ascents rather than introducing new subgadgets for simulating whole ascents. We find our approach simpler, and think that future work can more easily generalize our approach to other local search methods.

[2] Most often this is $V = [d]$, but in our construction of $\mathcal{C}^{\pm}_{n,\leq m}$ in Section 3 it will be $V = [m] \times [6]$.

assignment x is called a *local peak* in a fitness landscape if $f(x) \geq f(y)$ for all y adjacent to x. A sequence of assignments $x^0, x^1, \ldots, x^T$ is called an *ascent* in the fitness landscape if x^{t-1} and x^t are adjacent for all $t \in [T]$, $f(x^{t-1}) < f(x^t)$, and x^T is a local peak. An ascent is called a *steepest* ascent, if for all $t \in [T]$ and all assignments y adjacent to x^{t-1} it is the case that $f(x^t) \geq f(y)$. Any local search method (that only takes increasing steps) follows an ascent. Greedy local search follows a steepest ascent.

A binary Boolean valued constraint satisfaction problem (VCSP) on d Boolean variables with indexes in V is a set of constraints $\mathcal{C} = \{c_S\}$. Each constraint is a weight $c_S \in \mathbb{Z} \setminus \{0\}$ with an associated *scope* $S \subseteq V$ of size $|S| \leq 2$. We say that c_S is a *unary* constraint if $|S| = 1$, and a *binary* constraint if $|S| = 2$. Overloading notation, the set of constraint $\mathcal{C}$ *implements* a pseudo-Boolean function $\mathcal{C} : \{0,1\}^d \to \mathbb{Z}$ that we call the *fitness function*:

$$\mathcal{C}(x) = c_\emptyset + \sum_{c_i \in \mathcal{C}} c_i x_i + \sum_{c_{\{i,j\}} \in \mathcal{C}} c_{\{i,j\}} x_i x_j \tag{1}$$

Solving $\mathcal{C}$ means finding an assignment x, that maximizes the fitness function $\mathcal{C}(x)$. [3]

When discussing graph-theoretical properties of $\mathcal{C}$, we treat the scopes of binary constraints as edges. So $V(\mathcal{C}) = V$ and $E(\mathcal{C}) = \{i, j \mid i \neq j \text{ and } c_{\{i,j\}} \in \mathcal{C}\}$ is the set of scopes of the binary constraints of $\mathcal{C}$. For each variable index $i \in V$, we define the neighbourhood $N_{\mathcal{C}}(i) = \{j \mid i, j \in E(\mathcal{C})\}$ as the set of variable indexes in $V \setminus \{i\}$ that appear in a constraint with i. In this paper, we measure the simplicity of a VCSP-instance $\mathcal{C}$ by the maximum degree and by the pathwidth of its constraint graph. Given a graph $G = (V, E)$, the *pathwidth* of G is the minimum possible width of a path decomposition of G. A *path decomposition* of G is a sequence of sets $P = \{X_1, X_2, \ldots, X_p\}$ where $X_r \subset V(G)$ for $r \in [p]$ with the following three properties:

1. Every vertex $v \in V(G)$ is in at least one set X_r.
2. For every edge $\{u, v\} \in E(G)$ there exists an $r \in [p]$ such that X_r contains both u and v.
3. For every vertex $v \in V(G)$, if $v \in X_r \cap X_s$ then $u \in X_\ell$ for all ℓ such that $r \leq \ell \leq s$.

The *width* of a path decomposition is defined as $\max_{X_r \in P} |X_r| - 1$. We refer the reader to [6] for more details and for the definition of treewidth, and to [7] for standard graph terminology.

It is often useful to see how the value of the fitness function $\mathcal{C}$ changes when a single variable is modified. In particular, we denote with $\nabla_i \mathcal{C}(x) = \mathcal{C}(x[i:1]) - \mathcal{C}(x[i:0])$ the fitness change associated with changing variable $x_i = 0$ to $x_i = 1$ given some background assignment x. It is easy to see that $\nabla_i \mathcal{C}(x) = c_i + \sum_{j \in N_{\mathcal{C}}(i)} c_{\{i,j\}} x_j$, and the value of $\nabla_i \mathcal{C}(x)$ depends only on the assignment to variables with indexes in $N_{\mathcal{C}}(i)$. We use this to overload $\nabla_i \mathcal{C}$ to partial assignments: if $y \in \{0,1\}^{N(i)}$ we consider $\nabla_i \mathcal{C}(y)$ to be well defined.

We say that x_i has *preferred assignment* 1 in background x if $\nabla_i \mathcal{C}(x) > 0$ and preferred assignment 0 in background x if $\nabla_i \mathcal{C}(x) < 0$.

---

[3] One could also consider a VCSP-instance as a set of constraints $\hat{\mathcal{C}} = \{C_S\}$ where each $C_S : \{0,1\}^S \to \mathbb{Z}$ is a binary function. This formulation is equivalent to our formulation above because an arbitrary constraint with scope S can be expressed as a polynomial. For example, take $S = \{i, j\}$:

$$\begin{aligned} C_S(x_i, x_j) &= C_S(0,0)(1-x_i)(1-x_j) + C_S(1,0)x_i(1-x_j) + C_S(0,1)(1-x_i)x_j + C_S(1,1)x_i x_j \\ &= C_S(0,0) + (C_S(1,0) - C_S(0,0))x_i + (C_S(0,1) - C_S(0,0))x_j \\ &\quad + (C_S(1,1) - C_S(0,1) - C_S(1,0) + C_S(0,0))x_i x_j \end{aligned}$$

The second equality groups alike monomials. One can convert from the $\hat{\mathcal{C}}$ formulation to the $\mathcal{C}$ formulation by summing alike monomial terms across all the constraints. That is, the constant terms $C_S(0,0)$ are aggregated into $c_\emptyset$, all the $C_{\{i,\_\}}$ (i.e., coefficients appearing before x_i) aggregate into c_i, and $c_{\{i,j\}} = C_{\{i,j\}}(1,1) - C_{\{i,j\}}(0,1) - C_{\{i,j\}}(1,0) + C_{\{i,j\}}(0,0)$ (i.e., coefficient appearing before $x_i x_j$). It takes linear time to covert from $\hat{\mathcal{C}}$ to $\mathcal{C}$ (see Theorem 3.4 in Kaznatcheev, Cohen and Jeavons [14]).

▶ **Definition 1.** *Given two indexes $i \neq j$ we say that i* sign-depends *on j in background assignment x when $sign(\nabla_i \mathcal{C}(x)) \neq sign(\nabla_i \mathcal{C}(x[j : \overline{x_j}]))$. If there is no background assignment x such that i sign-depends on j then we say that i does not sign depend on j. If for all $j \neq i$ we have that i does not sign-depend on j then we say that i is sign independent.*

In other words, Definition 1 tells us that if i sign-depends on j, the preferred assignment of variable x_i depends on the assignment to variable x_j.

▶ **Definition 2** (Kaznatcheev and Vazquez Alferez [16])**.** *A* VCSP*-instance $\mathcal{C}$ is* oriented *if for every pair of indexes $\{i, j\}$ we have that either i does not sign-depend on j or j does not sign depend on i. If a* VCSP*-instance $\mathcal{C}$ is oriented and j sign depends on i we assign a direction from i to j to the edge $\{i, j\} \in E(\mathcal{C})$ (i.e. we orient the edge, hence the name).*

The constraint graphs of oriented VCSPs have no directed cycles [16]. Thus, if y is any assignment to the first k variables of a topological ordering of the variables of an oriented constraint graph, then k is sign-independent given background y. In other words, if $\mathcal{C}$ is oriented, there is an ordering on the variables such that later variables are conditionally-independent of earlier ones. This implies that the fitness landscape of $\mathcal{C}$ is single peaked on every sublandscape [16]. Depending on the research community, such landscapes are known as semismooth fitness landscapes [12, 16], completely unimodal pseudo-Boolean functions [10], or acyclic unique-sink orientations of the hypercube [19, 20]. Given that fitness landscapes implemented by oriented VCSPs are single-peaked, we use $x^*(\mathcal{C})$ to denote the peak of the landscape implemented by the oriented VCSP-instance $\mathcal{C}$.

3 Construction of $\mathcal{C}^{\pm}_{n,\leq m}$

Given two parameters $1 \leq m \leq n$, in this section, we construct the VCSP-instances $\mathcal{C}^{+}_{n,\leq m}$ and $\mathcal{C}^{-}_{n,\leq m}$ on $6m$ variables and, in Section 4, show that they both have a steepest ascent of length $7(2^m - 1)$. We construct the $\mathcal{C}^{\pm}_{n,\leq m}$ as a path of m gadgets $\mathcal{C}^{\pm}_{n,m}, \mathcal{C}^{-}_{n,m-1}, \ldots, \mathcal{C}^{-}_{n,2}, \mathcal{C}^{-}_{n,1}$ where each gadget $\mathcal{C}^{\pm}_{n,k}$ is defined on 6 variables $V_k = \{(k, i) \mid 1 \leq i \leq 6\}$. For notational convenience, we define $V_{\leq m} := \cup_{k=1}^{m} V_k$. Note that the two VCSPs are exactly the same except on the mth gadget where $\mathcal{C}^{+}_{n,\leq m}$ has $\mathcal{C}^{+}_{n,m}$ and $\mathcal{C}^{-}_{n,\leq m}$ has $\mathcal{C}^{-}_{n,m}$. We will present the construction of the gadgets $\mathcal{C}^{\pm}_{n,k}$ in two stages. First, we will define the scopes of all the constraints to get the constraint graph and show that these constraint graphs are very sparse (Proposition 3). Second, we will assign weights to the constraints and show that the VCSPs are oriented (Proposition 4).

Both the $\mathcal{C}^{-}_{n,k}$ and $\mathcal{C}^{+}_{n,k}$ gadgets have all six unary constraints and the same six binary constraints with scopes $\{(k, 1), (k, 2)\}$, $\{(k, 2), (k, 3)\}$, $\{(k, 3), (k, 6)\}$, $\{(k, 1), (k, 4)\}$, $\{(k, 4), (k, 5)\}$, $\{(k, 5), (k, 6)\}$. Finally, we connect adjacent gadgets with a single binary constraint with scope $\{(k, 6), (k-1, 1)\}$. The constraint graph of $\mathcal{C}^{-}_{n,k}$ along with the connections to the adjacent gadgets at $k+1$ and $k-1$ are shown in Figure 1. It is not hard to check based on the above definition that the constraint graphs of $\mathcal{C}^{\pm}_{n,\leq m}$ are sparse:

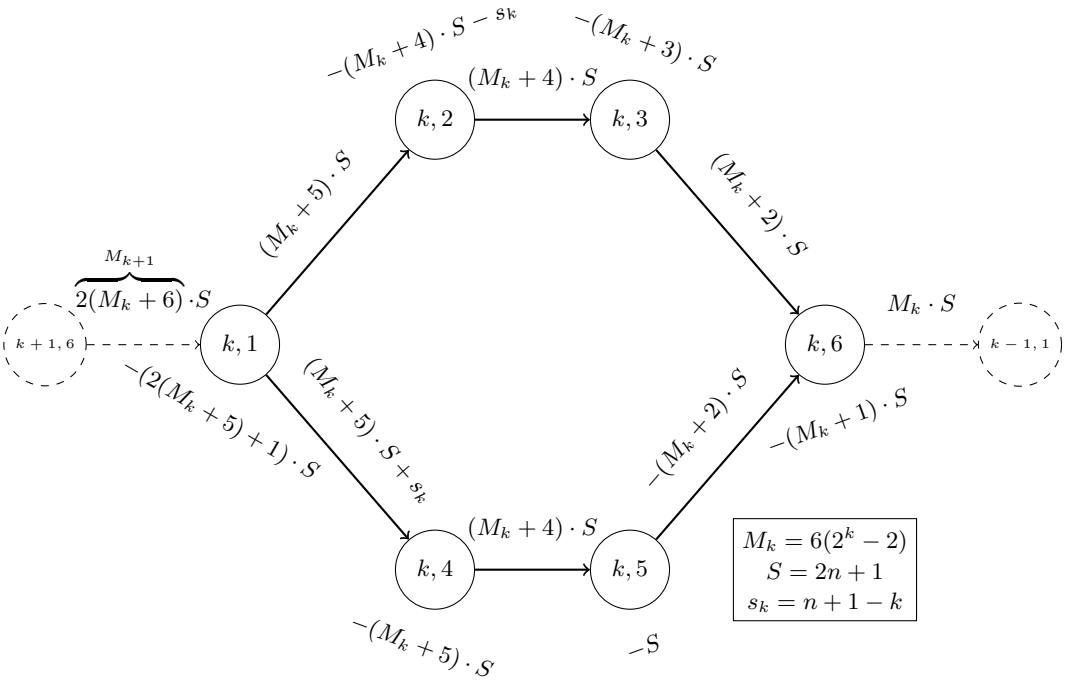


Figure 1 A gadget $\mathcal{C}^-_{n,k}$ with $M_k = 6(2^k-2)$, $S = 2n+1$, and $s_k = n+1-k$. The constraints of the kth of n gadgets are shown: the weights of unary constraints are next to their variables and the weights of binary constraints are above the edges that specify their scope. The orientation of the arcs is displayed, showing the instance is oriented. The dotted edges and vertices illustrate the connection to the neighboring gadgets. For the right boundary: there is no 0th gadget and thus no constraint with scope $\{(1,6),(0,1)\}$. For the left boundary: both of $\mathcal{C}^{\pm}_{n,\leq m}$ have no constraint with scope $\{(m+1,6),(m,1)\}$ but $\mathcal{C}^+_{n,m}$ also changes the weight of the unary on $(m,1)$ to $c^+_{(m,1)} = c^-_{(m,1)} + c_{(m+1,6),(m,1))} = S$.

▶ **Proposition 3.** *The constraint graph of $\mathcal{C}^{\pm}_{n,\leq m}$ has maximum degree 3 and pathwidth 2.*

Proof. The constraint graph of each $\mathcal{C}^{\pm}_{n,k}$ is a cycle so every vertex has degree 2. To create $\mathcal{C}^{\pm}_{n,\leq m}$ we add a single edge between consecutive gadgets, this raises the maximum degree to 3. For the pathwidth:
($\Rightarrow$) Path decomposition for $\mathcal{C}^{\pm}_{n,k}$ and adjacent variables: $\{(k+1,6),(k,1)\}$, $\{(k,1),(k,2),(k,4)\}$, $\{(k,2),(k,3),(k,4)\}$, $\{(k,3),(k,4),(k,5)\}$, $\{(k,3),(k,5),(k,6)\}$, $\{(k,6),(k-1,1)\}$.
($\Leftarrow$) Contracting $\{(k,1),(k,2)\}$, $\{(k,3),(k,6)\}$ and $\{(k,4),(k,5)\}$ shows K_3 is a minor of $\mathcal{C}^{\pm}_{n,k}$. ◀

We define the weights of the constraints sequentially using parameters $M_k = 6(2^k-2)$, $S = 2n+1$, and $s_k = n+1-k$. Since $C^-_{n,k}$ and $C^+_{n,k}$ are the same except for the unary on $(k,1)$, we use c for all the weights except for $c^-_{(k,1)}$ (Equation (14)) and $c^+_{(k,1)}$ (Equation (16)):

$$
\begin{aligned}
c_{\{(k,6),(k-1,1)\}} &= && M_k S && (2)\\
c_{(k,6)} &= -(|c_{\{(k,6),(k-1,1)\}}| + S) &&= -(M_k+1)S && (3)\\
c_{\{(k,3),(k,6)\}} &= |c_{(k,6)}| + S &&= (M_k+2)S && (4)\\
c_{\{(k,5),(k,6)\}} &= -|c_{\{(k,3),(k,6)\}}| &&= -(M_k+2)S && (5)\\
c_{(k,3)} &= -(|c_{\{(k,3),(k,6)\}}| + S) &&= -(M_k+3)S && (6)\\
c_{\{(k,2),(k,3)\}} &= |c_{(k,3)}| + S &&= (M_k+4)S && (7)\\
c_{(k,2)} &= -(|c_{\{(k,2),(k,3)\}}| + s_k) &&= -(M_k+4)S - s_k && (8)\\
c_{\{(k,1),(k,2)\}} &= |c_{(k,2)}| + S - s_k &&= (M_k+5)S && (9)\\
c_{(k,5)} &= &&= -S && (10)\\
c_{\{(k,4),(k,5)\}} &= |c_{(k,5)} + c_{\{(k,5),(k,6)\}}| + S &&= (M_k+4)S && (11)\\
c_{(k,4)} &= -(|c_{\{(k,4),(k,5)\}}| + S) &&= -(M_k+5)S && (12)\\
c_{\{(k,1),(k,4)\}} &= |c_{(k,4)}| + s_k &&= (M_k+5)S + s_k && (13)\\
c^-_{(k,1)} &= -(|c_{\{(k,1),(k,2)\}} + c_{\{(k,1),(k,4)\}}| + S - s_k) &&= -(2(M_k+5)+1)S && (14)\\
c_{\{(k+1,6),(k,1)\}} &= |c_{(k,1)}| + S &&= \underbrace{2(M_k+6)}_{M_{k+1}} S && (15)\\
c^+_{(k,1)} &= c^-_{(k,1)} + c_{\{(k+1,6),(k,1)\}} &&= S && (16)
\end{aligned}
$$

The above weights are shown on the constraint graph in Figure 1. This structure of weights has three important features that ensure the VCSP is oriented, that let us recurse on smaller k, and that help us control the steepest ascent. We have set the above weights in such a way that the following three properties hold in $\mathcal{C}^-_{n,k}$:

(a) all the unaries are negative,

(b) the magnitude of each variable's unary is greater than the binary constraint from that variable to a variable in the gadget with higher second index (or than the sum of the two binaries in the case of $|c^-_{(k,1)}| > c_{\{(k,1),(k,2)\}} + c_{\{(k,1),(k,4)\}}$), and

(c) the sum of the weights of any subset of "incoming" binaries on a variable (i.e., binary constraint to that variable from a variable in the gadget with lower second index) is either non-positive or greater than the magnitude of the unary (or the sum of the unary and any negative "outgoing" binaries in the case of $|c_{\{(k,4),(k,5)\}}| > |c_{(k,5)}| + |c_{\{(k,5),(k,6)\}}|$).

These three properties ensure that the resulting VCSP is oriented:

▶ **Proposition 4.** *$\mathcal{C}^{\pm}_{n,k}$ and $\mathcal{C}^{\pm}_{n,\leq m}$ are oriented.*

Proof. First we prove that $\mathcal{C}^{\pm}_{n,k}$ are oriented as in Figure 1. Note that $\mathcal{C}^{\pm}_{n,k}$ are cycles and every vertex has degree 2. We will denote by (k,i) and (k,j) the two neighbors of an arbitrary variable $(k,h) \in V_k$. Table 1 shows the fitness change $\nabla^{\pm}_{(k,h)}\mathcal{C}(x_{(k,i)}x_{(k,j)})$ for all possible assignments to $x_{(k,i)}$ and $x_{(k,j)}$. The cells of Table 1 are colored according to the sign of $\nabla^{\pm}_{(k,h)}\mathcal{C}$. It suffices to check Table 1 against Definition 2: The sign of $\nabla^{+}_{(k,1)}$ is always positive, and the sign of $\nabla^{-}_{(k,1)}$ is always negative, regardless of the assignment to $(k,2)$ and $(k,4)$, so $(k,1)$ does not sign depend on either $(k,2)$ or $(k,4)$ in $\nabla^{\pm}_{(k,1)}$.

On the other hand, for the rows where $h = 2,3,4$ and 5 the two columns where $x_{(k,i)} = 0$ are negative whilst the two columns corresponding to $x_{(k,i)} = 1$ are positive. This means that $(k,2),(k,3),(k,4)$ and $(k,5)$ sign-depend, respectively, on $(k,1),(k,2),(k,1)$ and $(k,4)$;

Table 1 Fitness change $\nabla_{(k,h)}\mathcal{C}^{\pm}_{n,k}$ incurred by flipping the assignment of variable (k,h) from $x_{(k,h)}=0$ to $x_{(k,h)}=1$ given the assignments of its two neighbors $x_{(k,i)}$ and $x_{(k,j)}$ in $\mathcal{C}^{\pm}_{n,k}$. We abbreviate $\nabla_{(k,h)}\mathcal{C}^{\pm}_{n,k}$ as $\nabla^{\pm}_{(k,h)}$. The dark gray cells highlight negative changes, and light gray cells highlight positive changes.

				$(\ x_{(k,i)}\ ,\ x_{(k,j)}\ )$			
h	i	j	$\nabla^{\pm}_{(k,h)}$	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
1	4	2	$\nabla^{+}_{(k,1)}$	S	$(M_k{+}6)S{+}s_k$	$(M_k{+}6)S$	$(2M_k{+}11)S{+}s_k$
			$\nabla^{-}_{(k,1)}$	$-(2(M_k{+}5){+}1)S$	$-(M_k{+}6)S{+}s_k$	$-(M_k{+}6)S$	$-S{+}s_k$
2	1	3	$\nabla^{\pm}_{(k,2)}$	$-(M_k{+}4)S{-}s_k$	$-s_k$	$S{-}s_k$	$(M_k{+}5)S{-}s_k$
3	2	6	$\nabla^{\pm}_{(k,3)}$	$-(M_k{+}3)S$	$-S$	S	$(M_k{+}3)S$
4	1	5	$\nabla^{\pm}_{(k,4)}$	$-(M_k{+}5)S$	$-S$	s_k	$(M_k{+}4)S{+}s_k$
5	4	6	$\nabla^{\pm}_{(k,5)}$	$-S$	$-(M_k{+}3)S$	$(M_k{+}3)S$	S
6	3	5	$\nabla^{\pm}_{(k,6)}$	$-(M_k{+}1)S$	S	$-(2M_k{+}3)S$	$-(M_k{+}1)S$

but do not sign-depend, respectively, on $(k,3),(k,6),(k,5)$ and $(k,6)$. Finally, from the fact that the row with $h=6$ has different signs on the two columns where $x_{(k,3)}=0$, and also different signs on the two columns where $x_{(k,5)}=1$ we can see that $(k,6)$ sign-depends on $(k,3)$ and $(k,5)$ in $\nabla^{\pm}_{(k,1)}$.

Second, to show that $\mathcal{C}^{\pm}_{n,\leq m}$ are oriented, it suffices to show that for $1\leq k<m$ the preferred assignment to $x_{(k+1,6)}$ is independent of the assignment to $x_{(k,1)}$. This is equivalent to showing that the sign of the last row in Table 1 does not change if we add M_kS to each column, which is obviously true. This is sufficient to show the statement of the proposition.

To show the additional property that the orientation of the edge $\{x_{(k+1,6)},x_{(k,1)}\}$ is as it appears in Figure 1, we must show that $(k,1)$ sign-depends on the assignment to $(k+1,6)$. But this can be seen from the fact that the first row of our table is equivalent to having $x_{(k+1,6)}=1$ in the background, and the second row is equivalent to having $x_{(k+1,6)}=0$. Since the first and second rows have different signs the sign-dependence follows. ◀

The weight of $c^{+}_{(k,1)}$ in Equation (16) is set so that we have the next two properties:

(d) If we fix $x_{(k,6)}=0$ then the sublandscape spanned by $V_{\leq k-1}$ is the same as the landscape implemented by $\mathcal{C}^{-}_{n,\leq k-1}$, and

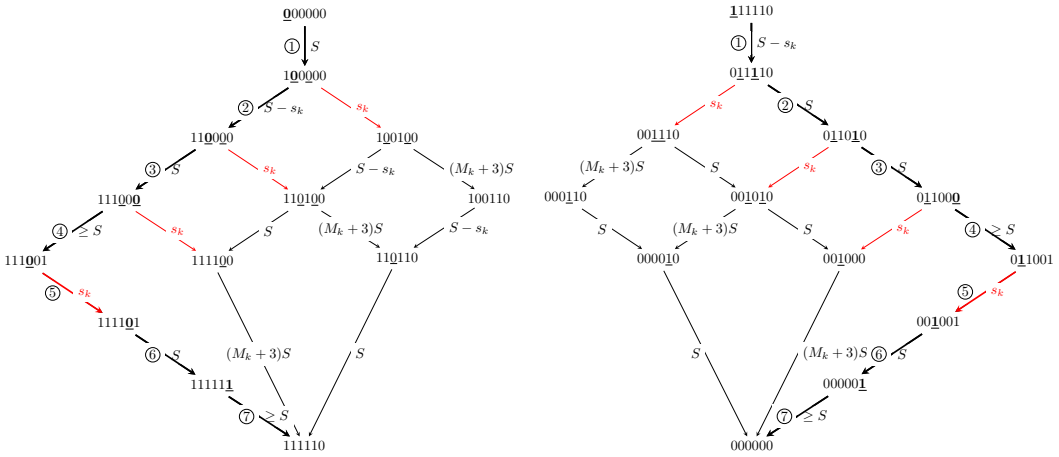
(e) if we fix $x_{(k,6)}=1$ then the sublandscape spanned by $V_{\leq k-1}$ is the same as the landscape implemented by $\mathcal{C}^{+}_{n,\leq k-1}$.

This allows us to analyze ascents on $\mathcal{C}^{\pm}_{n,\leq m}$ inductively because when $x_{(m,6)}$ is fixed to 1 or 0 we can use the analysis from our inductive hypothesis of $\mathcal{C}^{+}_{n,\leq m-1}$ or $\mathcal{C}^{-}_{n,\leq m-1}$, respectively. It also makes it very easy to check for the peaks of our landscapes:

▶ **Proposition 5.** *The semismooth fitness landscapes of $\mathcal{C}^{-}_{n,k}$, $\mathcal{C}^{-}_{n,\leq m}$, $\mathcal{C}^{+}_{n,k}$, and $\mathcal{C}^{+}_{n,\leq m}$ have their unique fitness peaks at $x^*(\mathcal{C}^{-}_{n,k})=000000$, $x^*(\mathcal{C}^{-}_{n,\leq m})=0^{6m}$, $x^*(\mathcal{C}^{+}_{n,k})=111110$, and $x^*(\mathcal{C}^{+}_{n,\leq m})=x^*(\mathcal{C}^{+}_{n,m})0^{6(m-1)}=1111100^{6(m-1)}$, respectively.*[4]

Proof. By property (a), all the unaries in $\mathcal{C}^{-}_{n,k}$ and $\mathcal{C}^{-}_{n,\leq m}$ are negative, so the all zero assignment is a local peak. Since the landscapes are semismooth from Proposition 4, this is also the unique global peak.

[4] For convenience, when we specify an assignment x the variables are ordered from left to right by decreasing first index and, to break ties, by increasing second index. For example, for $\mathcal{C}^{+}_{n,\leq m}$ the assignment $x=0100010^{6(m-1)}$ sets $x_{(m,2)}=1, x_{(m,6)}=1$ and all other variables to 0.



(a) Partial fitness landscape of $\mathcal{C}^{+}_{n,k}$. **(b)** Partial fitness landscape of $\mathcal{C}^{-}_{n,k}$.

Figure 2 All ascents from 000000 in the fitness landscape of $\mathcal{C}^{+}_{n,k}$ **(a)** and from 111110 in the fitness landscape of $\mathcal{C}^{-}_{n,k}$ **(b)**. The increase in fitness of each step is shown on the edges, with small increments (s_k) highlighted in red. The steepest ascent is shown in bold. For emphasis, bits that lead to a fitness increase when flipped are underlined, and the bit flipped by steepest ascent is bolded and numbered in the same way as in Equations (19)–(22) and Equations (23)–(26).

For $\mathcal{C}^{+}_{n,k}$, $x_{(k,1)}$ is conditionally-independent of all other variables and has $c^{+}_{(k,1)} = S > 0$, so the preferred assignment is $x_{(k,1)} = 1$. With $x_{(k,1)}$ set to 1, the preferred assignments for $x_{(k,2)}$ and $x_{(k,4)}$ are also 1; and based on those, the preferred assignments for $x_{(k,3)}$ and $x_{(k,5)}$ are also 1. This leaves $x_{(k,6)}$ which, conditional on $x_{(k,3)} = x_{(k,5)} = 1$, prefers $x_{(k,6)} = 0$. Overall, this gives $x^*(\mathcal{C}^{+}_{n,k}) = 111110$.

Since $x_{(k,6)} = 0$, property (d) implies that $x^*(\mathcal{C}^{+}_{n,\leq m}) = x^*(\mathcal{C}^{+}_{n,m})x^*(\mathcal{C}^{-}_{n,\leq m-1})$. ◀

Finally, the difference in increase of Equations (8) and (13) from the other weights ensures that:

(f) steps from assignments $0\underline{1}1x_4x_5x_6$ to $0\mathbf{0}1x_4x_5x_6$ and from $1x_2x_3\underline{0}0x_6$ to $1x_2x_3\mathbf{1}0x_6$ increase fitness by exactly $s_k \leq n$, and

(g) all other fitness increasing steps increase fitness by at least $S - s_k > n$.

As we will see in the next section, these "small" steps allow us to control in what order each block V_k of variables appears in the steps of the steepest ascent. Combined with properties (d) and (e) this lets us show the exponential steepest ascent by induction on m.

4 Exponential steepest ascent in the landscape of $\mathcal{C}^{\pm}_{n,\leq m}$

We can now show that it takes a large number of steps to go from the assignment $x^*(\mathcal{C}^{-}_{n,\leq m}) = 0^{6m}$ to the peak $x^*(\mathcal{C}^{+}_{n,\leq m}) = 1111100^{6(m-1)}$ in the semismooth fitness landscape implemented by $\mathcal{C}^{+}_{n,\leq m}$ (and vice versa for the landscape implemented by $\mathcal{C}^{-}_{n,\leq m}$):

▶ **Theorem 6.** *Both the steepest ascents starting from $x^*(\mathcal{C}^{-}_{n,\leq m}) = 0^{6m}$ in the fitness landscape of $\mathcal{C}^{+}_{n,\leq m}$ and from $x^*(\mathcal{C}^{+}_{n,\leq m}) = 1111100^{6(m-1)}$ in the fitness landscape of $\mathcal{C}^{-}_{n,\leq m}$ have length $7(2^m - 1)$, where each step increases fitness by at least s_m.*

Proof. Our proof is by induction on the number of gadgets m. We will show that by adding the gadget $\mathcal{C}^{\pm}_{n,m}$, steepest ascents of length T_{m-1} in the landscape of $\mathcal{C}^{\pm}_{n,\leq m-1}$ will convert to steepest ascents of length $T_m = 7 + 2T_{m-1}$ in the landscape of $\mathcal{C}^{\pm}_{n,\leq m}$. To do this, we will

look at all the ascents that take us from $x^*(\mathcal{C}^-_{n,m}) = 000000$ to $x^*(\mathcal{C}^+_{n,m}) = 111110$ in the landscape of the gadget $\mathcal{C}^+_{n,m}$ and vice-versa for the gadget $\mathcal{C}^+_{n,m}$. All of these ascents are in Figure 2a for $\mathcal{C}^+_{n,m}$ and in Figure 2b for $\mathcal{C}^-_{n,m}$. Each arrow in Figure 2 is labeled by the fitness increase from flipping the appropriate bit. The steepest ascent is bolded.

As we can see from the figures, although there exists a minimal ascent of length five between these two assignments that does not flip the $x_{(m,6)}$ variable, this is not the steepest ascent. Instead, the steepest ascent from $x^*(\mathcal{C}^-_{n,m})$ to $x^*(\mathcal{C}^+_{n,m})$ in the fitness landscape of $\mathcal{C}^+_{n,m}$ (and vice-versa for $\mathcal{C}^-_{n,m}$) takes seven steps and flips $x_{(m,6)}$ twice (at step ④ and ⑦).

This double flip of $x_{(m,6)}$ is what creates the recursion in $\mathcal{C}^\pm_{n,\leq m}$ that forces the steepest ascent in $\mathcal{C}^\pm_m$ to trigger twice as many ascents in $\mathcal{C}^\pm_{m-1}$. Specifically, although the first four steps of the steepest ascent in the landscape of $\mathcal{C}^\pm_{n,m}$ increase fitness by a large amount $\geq S - s_m > n$, step ⑤ increases fitness by only s_m. Thus, the steepest ascent in the sublandscape spanned by V_m "pauses" after step ④ and lets the steepest ascents in V_{m-1} take over with steps that increase fitness by an amount $\geq s_{m-1} > s_m$.

With this intuition in mind, look at the steepest ascent in the fitness landscape of $\mathcal{C}^+_{n,\leq m}$ starting from 0^{6m}. For brevity, define the (partial) assignments $x^*_\pm$ as the peaks of $\mathcal{C}^\pm_{n,\leq m-1}$:

$$x^*_- := x^*(\mathcal{C}^-_{n,\leq m-1}) = 0^{6(m-1)}, \text{ and} \tag{17}$$

$$x^*_+ := x^*(\mathcal{C}^+_{n,\leq m-1}) = 1111100^{6(m-2)}. \tag{18}$$

Now we can rewrite our starting assignment $x^*(\mathcal{C}^-_{n,\leq m}) = 0^{6m}$ as $000000x^*_-$ and note that the first four flips are entirely in V_m:

$$\overbrace{\underline{\mathbf{0}}00000x^*_-}^{x^*(\mathcal{C}^-_{n,\leq m})} \xrightarrow{①} 1\underline{\mathbf{0}}0000x^*_- \xrightarrow{②} 11\underline{\mathbf{0}}000x^*_- \xrightarrow{③} 111\underline{00}\underline{\mathbf{0}}x^*_- \xrightarrow{④} 111\underline{00}1\underline{\mathbf{x}^*_-} \tag{19}$$

where the variables that can flip are underlined and the variable that will be flipped by steepest ascent is bolded. For step ① there is only one choice to flip. On the subsequent two steps (②,③) the first of the two 0s is chosen by steepest ascent because they increase fitness by $S - s_m$ and S (which are both $> n$) while flipping the second 0 to a 1 would increase fitness by only $s_m \leq n$.

Step ④ is the most interesting. It flips $x_{(m,6)}$ since that increases fitness by $S > n \geq s_m$. In so doing, this flip transforms the landscape for variables on $V_{\leq m-1}$ from the one implemented by $\mathcal{C}^-_{n,\leq m-1}$ to the one implemented by $\mathcal{C}^+_{n,\leq m-1}$. In the landscape of $\mathcal{C}^+_{n,\leq m-1}$, x^*_- is no longer the peak, and hence it is underlined. Furthermore x^*_- is bolded because, by construction, all fitness increasing flips of variables in $V_{\leq m-1}$ increase fitness by $\geq s_{m-1} > s_m$ and so steepest ascent will flip variables in $V_{\leq m-1}$ instead of the $x_{(m,3)}$ flip that only increases fitness by s_m:

$$111\underline{00}1\underline{\mathbf{x}^*_-} \xrightarrow{T_{m-1} \text{ steps of } V_{\leq m-1} \text{ in landscape of } \mathcal{C}^+_{n,\leq m-1}} 111\underline{\mathbf{0}}01x^*_+ \tag{20}$$

Once all the steps in $V_{\leq m-1}$ are taken, steepest ascent can return to V_m, where the only remaining fitness-increasing step is the small step at $x_{(m,4)}$ that increases fitness by only s_m. This step subsequently opens two more steps in V_m:

$$111\underline{\mathbf{0}}01x^*_+ \xrightarrow{⑤} 1111\underline{\mathbf{0}}1x^*_+ \xrightarrow{⑥} 11111\underline{\mathbf{1}}x^*_+ \xrightarrow{⑦} 111110\underline{\mathbf{x}^*_+} \tag{21}$$

As with the first four steps in Equation (19), the most interesting step is the final step (⑦). It flips $x_{(m,6)}$ from 1 to $x_{(m,6)} = 0$. In so doing, this flip transforms the landscape for variables on $V_{\leq m-1}$ from the one implemented by $\mathcal{C}^+_{n,\leq m-1}$ to the one implemented by

$\mathcal{C}^-_{n,\leq m-1}$. Thus, "undoing" step ④. In the landscape of $\mathcal{C}^-_{n,\leq m-1}$, x^*_+ is no longer the peak, and hence underlined and bolded. Steepest ascent finishes with all remaining steps in $V_{\leq m-1}$:

$$111110\underline{\mathbf{x}^*_+} \xrightarrow{T_{m-1} \text{ steps of } V_{\leq m-1} \text{ in landscape of } \mathcal{C}^-_{n,\leq m-1}} \underbrace{111110x^*_-}_{x^*(\mathcal{C}^+_{n,\leq m})} \tag{22}$$

Similar to the steepest ascent in Equations (19)–(22), the steepest ascent in the fitness landscape of $\mathcal{C}^-_{n,\leq m}$ starting from the assignment $x^*(\mathcal{C}^+_{n,\leq m})$ has the following steps:

$$\overbrace{\underline{\mathbf{1}}11110x^*_-}^{x^*(\mathcal{C}^+_{n,\leq m})} \xrightarrow{①} 01\underline{\mathbf{1}}10x^*_- \xrightarrow{②} 011\underline{\mathbf{1}}0x^*_- \xrightarrow{③} 01100\underline{\mathbf{0}}x^*_- \xrightarrow{④} 011001\underline{\mathbf{x}^*_-} \tag{23}$$

$$011001\underline{\mathbf{x}^*_-} \xrightarrow{T_{m-1} \text{ steps of } V_{\leq m-1} \text{ in landscape of } \mathcal{C}^+_{n,\leq m-1}} 0\underline{\mathbf{1}}1001x^*_+ \tag{24}$$

$$0\underline{\mathbf{1}}1001x^*_+ \xrightarrow{⑤} 00\underline{\mathbf{1}}001x^*_+ \xrightarrow{⑥} 00000\underline{\mathbf{1}}x^*_+ \xrightarrow{⑦} 000000\underline{\mathbf{x}^*_+} \tag{25}$$

$$000000\underline{\mathbf{x}^*_+} \xrightarrow{T_{m-1} \text{ steps of } V_{\leq m-1} \text{ in landscape of } \mathcal{C}^-_{n,\leq m-1}} \underbrace{000000x^*_-}_{x^*(\mathcal{C}^-_{n,\leq m})} \tag{26}$$

Finally, both the steepest ascent in the fitness landscape of $\mathcal{C}^+_{n,\leq m}$ starting from the assignment $x^*(\mathcal{C}^-_{n,\leq m})$ (Equations (19)–(22)) and the steepest ascent in the fitness landscape of $\mathcal{C}^-_{n,\leq m}$ starting from the assignment $x^*(\mathcal{C}^+_{n,\leq m})$ (Equations (23)–(26)) have lengths of $T_m = 7+2T_{m-1}$ steps. The recurrence of $T_1 = 7$ and $T_m = 7 + 2T_{m-1}$ is solved by $T_m = 7(2^m - 1)$. ◀

Combining Theorem 6 with that greedy local search follows steepest ascents, and using the facts that our instances are very sparse by Proposition 3, and that they are oriented by Proposition 4, we conclude that greed is slow on sparse graphs of oriented valued constraints:

▶ **Theorem 7.** *Greedy local search can take* $7(2^n - 1)$ *steps to find the unique local optimum in oriented binary Boolean* VCSP*-instances* $\mathcal{C}^{\pm}_{n,\leq n}$ *on* $6n$ *variables with* $6n$ *unary constraints and* $7n - 1$ *binary constraints, maximum degree* 3 *and pathwidth* 2.

5 Discussion

There are two important structural features of our construction: that it is sparse (Proposition 3) and that is is oriented (Proposition 4). Sparseness resolves in the negative the two open questions on the efficiency of greedy local search for VCSP-instances of degree 3 and treewidth 2. Given that all ascents are short for VCSPs of degree 2 [13] and treewidth 1 [14], the $\mathcal{C}^{\pm}_{n,\leq n}$ family of instances that are hard for greedy local search belong to the simplest class of instances where *some* local search has the possibility to fail. That $\mathcal{C}^{\pm}_{n,\leq n}$ is an oriented VCSP (Proposition 4), reminds us that this failure of greedy local search is not due to the popular suspect of "bad" local peaks blocking the way to a hard to find global peak. Oriented VCSPs only produce semismooth fitness landscapes that are single peaked on each sublandscape [16]: there are no "bad" local peaks in this family of instances, there is the single global peak. Further, in semismooth fitness landscapes, there is always a short ascent of Hamming distance from any initial assignments to unique peak [10, 12]. A short ascents is always available in $\mathcal{C}^{\pm}_{n,\leq n}$, but greed stops us from find it.

Many other local search heuristics do not lose their way on oriented VCSPs. In contrast to Theorem 6, consider how a local search method that chooses improving steps at random – known as random ascent – behaves on $\mathcal{C}^{-}_{n,\leq n}$ starting from any assignment. Since there are $6n$ variables, any improving bit flip has a probability of at least $1/6n$ of being chosen as the next step. Thus, if $x_{(n,1)} \neq 0 (= x^*_{(n,1)})$ then after an expected number of at most $6n$ steps, it will be flipped to 0. Given the oriented constraints, it cannot be flipped back for the rest of the run. Next, if – at this point in the run – the current assignment has $x_{(n,2)} \neq 0$ or $x_{(n,4)} \neq 0$ then in an expected number of about $2 \cdot 6n$ steps, $x_{(n,2)}$ and $x_{(n,4)}$ will also be flipped to 0. Given the oriented constraints and that the only in-edge to these variables is from $x_{(n,1)}$ which is now fixed to 0, $x_{(n,2)}$ and $x_{(n,4)}$ will not be flipped back for the rest of the run. This logic continues for $x_{(n,3)}$ and $x_{(n,5)}$, then $x_{(n,6)}$, then $x_{(n-1,1)}$ and all other remaining variables following the arrows of the oriented constraints (i.e., in the topological order of the oriented VCSP). This simple argument gives us a bound of $(6n)^2$ on the expected number of steps for random ascent to find the peak.

The behavior of inadvertently fixing variables to their optimal state in the topological order of an oriented VCSP is a feature of many local search methods, not just random ascent. Kaznatcheev and Vazquez Alferez [16] give a more careful analysis of random ascent on general oriented binary Boolean VCSPs. Applying their bound yields an expected number of at most $32n^2 - 18n$ steps to solve $\mathcal{C}^{\pm}_{n,\leq n}$. For the task of solving general oriented VCSPs they give quadratic bounds on the number of steps taken by, random ascent, simulated annealing, Zadeh's simplex rule, the Kerninghan-Lin heuristic, and various other local search methods. Thus, not only does greedy local search require an exponential number of steps on oriented VCSPs of maximum degree 3 and pathwidth 2, but lots of other local search methods can solve these instances in quadratic time. We conclude that, among local search methods, greed is slow.

References

1 Emile Aarts and Jan Karel Lenstra, editors. *Local Search in Combinatorial Optimization*. Princeton University Press, Princeton, NJ, USA, 2003.

2 Umberto Bertelè and Francesco Brioschi. On non-serial dynamic programming. *Journal of Combinatorial Theory, Series A*, 14(2):137–148, 1973. doi:10.1016/0097-3165(73)90016-2.

3 Michaela Borzechowski. The complexity class polynomial local search (pls) and pls-complete problems. Bachelor's thesis, Freie Universitat Berlin, 2016. URL: https://www.mi.fu-berlin.de/inf/groups/ag-ti/theses/download/Borzechowski16.pdf.

4 Clément Carbonnel, Miguel Romero, and Stanislav Živný. The complexity of general-valued constraint satisfaction problems seen from the other side. *SIAM Journal on Computing*, 51(1):19–69, 2022. doi:10.1137/19M1250121.

5 David A Cohen, Martin C Cooper, Artem Kaznatcheev, and Mark Wallace. Steepest ascent can be exponential in bounded treewidth problems. *Operations Research Letters*, 48:217–224, 2020. doi:10.1016/j.orl.2020.02.010.

6 Marek Cygan, Fedor V Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer International Publishing, 2015. doi:10.1007/978-3-319-21275-3.

7 Reinhard Diestel. *Graph Theory*. Springer Publishing Company, Incorporated, 5th edition, 2017.

8 Robert Elsässer and Tobias Tscheuschner. Settling the complexity of local max-cut (almost) completely. In *International Colloquium on Automata, Languages, and Programming*, pages 171–182. Springer, 2011. doi:10.1007/978-3-642-22006-7_15.

9 Armin Haken and Michael Luby. Steepest descent can take exponential time for symmetric connection networks. *Complex Syst.*, 2(2):191–196, April 1988. URL: `http://www.complex-systems.com/abstracts/v02_i02_a03.html`.

10 Peter L Hammer, Bruno Simeone, Th M Liebling, and Dominique de Werra. From linear separability to unimodality: A hierarchy of pseudo-boolean functions. *SIAM Journal on Discrete Mathematics*, 1(2):174–184, 1988. `doi:10.1137/0401019`.

11 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988. `doi:10.1016/0022-0000(88)90046-3`.

12 Artem Kaznatcheev. Computational complexity as an ultimate constraint on evolution. *Genetics*, 212(1):245–265, 2019. `doi:10.1534/genetics.119.302000`.

13 Artem Kaznatcheev. *Algorithmic Biology of Evolution and Ecology*. PhD thesis, University of Oxford, 2020.

14 Artem Kaznatcheev, David A Cohen, and Peter Jeavons. Representing fitness landscapes by valued constraints to understand the complexity of local search. *Journal of Artificial Intelligence Research*, 69:1077–1102, 2020. `doi:10.1613/jair.1.12156`.

15 Artem Kaznatcheev and Melle van Marle. Exponential steepest ascent from valued constraint graphs of pathwidth four. In *30th International Conference on Principles and Practice of Constraint Programming (CP 2024)*, pages 17:1–17:16, 2024. `doi:10.4230/LIPIcs.CP.2024.17`.

16 Artem Kaznatcheev and Sofia Vazquez Alferez. When is local search both effective and efficient?, 2024. `doi:10.48550/arXiv.2410.02634`.

17 Burkhard Monien and Tobias Tscheuschner. On the power of nodes of degree four in the local max-cut problem. In *Algorithms and Complexity: 7th International Conference, CIAC 2010, Rome, Italy, May 26-28, 2010. Proceedings 7*, pages 264–275. Springer, 2010. `doi:10.1007/978-3-642-13073-1_24`.

18 Svatopluk Poljak. Integer linear programs and local search for max-cut. *SIAM Journal on Computing*, 24(4):822–839, 1995. `doi:10.1137/S0097539793245350`.

19 Ingo A Schurr. *Unique sink orientations of cubes*. PhD thesis, ETH Zurich, 2004.

20 Tibor Szabó and Emo Welzl. Unique sink orientations of cubes. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 547–555. IEEE, 2001. `doi:10.1109/SFCS.2001.959931`.

21 Melle van Marle. Complexity of greedy local search for constraint satisfaction. Master's thesis, Utrecht University, 2025.

22 David P. Williamson and David B. Shmoys. *Greedy Algorithms and Local Search*, pages 27–56. Cambridge University Press, 2011.