When Is Local Search Both Effective and Efficient?

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

Combinatorial optimization problems implicitly define *fitness landscapes* that combine the numeric structure of the "fitness" function to be maximized with the combinatorial structure of which assignments are "adjacent". Local search starts at an assignment in this landscape and successively moves assignments until no further improvement is possible among the adjacent assignments. Classic analyses of local search algorithms have focused more on the question of effectiveness ("did we find a *good* solution?") and often implicitly assumed that there are no doubts about their efficiency ("did we find it *quickly*?"). But there are many reasons to doubt the efficiency of local search. Even if we focus on fitness landscapes on the hypercube that are single peaked on every subcube (known as semismooth fitness landscapes, completely unimodal pseudo-Boolean functions, or acyclic unique sink orientations) where effectiveness is obvious, many local search algorithms are known to be inefficient. Since fitness landscapes are unwieldy exponentially large objects, we focus on their polynomial-sized representations by instances of valued constraint satisfaction problems (VCSP). We define a "direction" for valued constraints such that *directed* VCSPs generate semismooth fitness landscapes. We call directed VCSPs *oriented* if they do not have any pair of variables with arcs in both directions. Since recognizing if a VCSP-instance is directed or oriented is coNP-complete, we generalized oriented VCSPs as *conditionally-smooth* fitness landscapes where the structural property of "conditionally-smooth" *is* recognizable in polynomial time for a VCSP-instance. We prove that many popular local search algorithms like random ascent, simulated annealing, history-based rules, jumping rules, and the Kernighan-Lin heuristic are very efficient on conditionally-smooth landscapes. But conditionally-smooth landscapes are still expressive enough so that other well-regarded local search algorithms like steepest ascent and random facet require a super-polynomial number of steps to find the fitness peak.

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

Keywords and phrases valued constraint satisfaction problem, local search, algorithm analysis, constraint graphs, pseudo-Boolean functions, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.59

Related Version *Full Version*: https://arxiv.org/abs/2410.02634

1 Introduction

Local search algorithms start at an initial assignment and successively move to better adjacent assignments until no further improvement is possible. As with all algorithms, we can ask about their effectiveness ("did we find a *good* solution?") and efficiency ("did we find it *quickly*?"). Often, we do not know the precise answers to these questions, but still choose to use local search in combinatorial optimisation [1, 12, 18, 42, 49].[1] In practice, local search seems to efficiently find effective solutions. But practice is not theory.

[1] Sometimes local search is chosen because it is easy to implement. If we do not expect any algorithm to be effective and efficient on all inputs – say for NP-hard combinatorial optimization problems – then why not choose an algorithm that will take less effort to implement and maintain? At other times, local search is forced on us. If we want to understand the evolution of biological systems [30, 31] or of social systems in business [26, 38] and economics [45], or even just energy-minimization in physical systems, then we need to study the effectiveness and efficiency of the local search algorithms followed by nature.

As theorists, we can view combinatorial optimization problems as implicitly defining *fitness landscapes* that combine the numeric structure of the "fitness" function to be maximized with the combinatorial structure of which assignments are "adjacent" [1, 12, 32]. Since many local search algorithms stop at local peaks in these fitness landscapes, much work asks questions of effectiveness like "are these peaks 'good enough'?" [1, 18]. Such questions of effectiveness can be avoided by focusing on just single-peaked fitness landscapes. Or even more stringently, by focusing on fitness landscapes on the hypercube of assignments that are single peaked on every subcube. These landscapes are known – depending on the research community – as semismooth fitness landscapes [30], completely unimodal pseudo-Boolean functions [21], or acyclic unique-sink orientations (AUSOs) of the hypercube [48, 50]. Semismooth fitness landscapes have a single peak, a short ascent from any starting point to this peak, and have nothing resembling other local peaks to potentially block local search (Propositions 3 and 4; Hammer et al. [21], Poelwijk et al. [44], and Kaznatcheev [30]; see Section 3). Local search is effective in finding the best possible result in semismooth landscapes, but can it do so efficiently?

The (in)efficiency of finding local peaks for general combinatorial optimization problems is captured by the complexity class of polynomial local search (PLS; [28]): if $\mathsf{FP} \neq \mathsf{PLS}$ then there is no polynomial time algorithm (even non-local) that can find a local peak in general fitness landscapes. Many problems that generate these hard landscapes are complete under *tight* PLS-reductions, so have families of instances and initial assignments such that all ascents (i.e., sequences of adjacent assignments with strictly increasing fitness; Section 3) to any local peak are exponentially long [5, 28]. Thus, ascent-following local search will not find a local peak in polynomial time, even if $\mathsf{FP} = \mathsf{PLS}$.[2] In contrast, semismooth fitness landscapes always have some short ascent to their unique fitness peak. Thus, any results about (in)efficiency will have to be stated in terms of specific local search algorithms.[3] And there exist constructions of semismooth landscapes that cannot be solved efficiently by various popular local search algorithms including SteepestAscent [13, 27, 30, 34], RandomAscent [22, 39], RandomFacet [17], jumping rules [47], and many others. It is generally believed that for any particular local search algorithm, there will be some family of semismooth fitness landscapes that show that the algorithm is not efficient. On semismooth fitness landscapes, even if local search is always effective, it is not always efficient.

This raises our main question for this paper: *when is local search efficient on single-peaked landscapes?* In a classic complexity theory setting, one wants to keep fixed a given problem (or class of problem instances) and seek for the simplest or most natural algorithm that solves this problem effectively (i.e., correctly) and efficiently (i.e., in polynomial time). We flip this formula around. We fix the algorithms and seek the most complex class of problem instances on which these algorithms can solve the problem effectively and efficiently. Specifically, we fix a collection of many popular local search algorithms and seek to find a large class of

[2] Restricting PLS-complete problems to a subset of instances that are tractable in polynomial time does not mean that particular local search algorithms will also find a peak in polynomial time. For example, Weighted 2-SAT is a PLS-complete problem [37, 46] and thus binary Boolean valued constraint satisfaction problems (VCSP) are PLS-complete. For VCSP-instances of bounded treewidth, the global peak can be found in polynomial time [4, 6]. Thus, bounded-treewdith VCSP is not PLS-complete. But exponentially long ascents exist (and greedy local search algorithms like SteepestAscent can take exponentially long) even with binary Boolean VCSP-instances of pathwidth 2 [32, 7, 33, 34, 51]. What is easy for non-local search is not necessarily easy for local search.

[3] The special case of finding the peak in semismooth landscapes is reducible to Unique-End-of-Potential-Line – the complete problem for $\mathsf{UEOPL} \subseteq \mathsf{PLS} \cap \mathsf{PPAD}$ [15, 19]. It is believed that UEOPL is strictly easier than PLS, but still not tractable in polynomial time. In a blackbox setting (no access to a concise description of the fitness landscape), UEOPL and PLS are not tractable in polynomial time.

single-peaked fitness landscapes on which this collection of many local search algorithms is efficient (we abbreviate this as *efficient-for-many*; and we let our fixed collection be algorithms from a broader algorithm-class that we name poly-bypass).

Given that blackbox fitness landscapes are unwieldy exponentially large objects, we open the blackbox by representing fitness landscapes by instances of valued constraint satisfaction problems (VCSPs) [32].[4] Specifically, we study binary Boolean VCSPs, also known as quadratic pseudo-Boolean functions. Since arbitrary binary Boolean VCSPs are PLS-complete [5, 28, 37, 46], we will be looking for subclasses of binary Boolean VCSPs that implement single-peaked fitness landscapes on which poly-bypass local search algorithms can find the peak efficiently. Thus, our goal is to find an easy to check structural property of VCSPs that captures the most expressive subclass of binary Boolean VCSPs for which many popular local search algorithms are both effective and efficient.

2 Summary of Results

To avoid concerns over (in)effectiveness, we start Section 4 by identifying the binary Boolean VCSPs that implement semismooth fitness landscapes. We do this by assigning each edge in the constraint graph of a binary Boolean VCSP corresponding to a valued constraint between variable x_i and x_j one of two types: directed ($i \to j$ or $j \to i$) or bidirected ($i \leftrightarrow j$). We show that VCSPs without bidirectional edges are equivalent to semismooth fitness landscapes and name them *directed* VCSPs. Given many constructions of hard semismooth fitness landscapes [13, 17, 22, 27, 30, 39, 47], we do not expect local search to be efficient on directed VCSPs.

To find efficiency, we define *oriented* VCSPs as a further restriction on directed VCSPs. For each pair of variables x_i, x_j in an oriented VCSP there can be at most one of $i \to j$ or $j \to i$: the constraint graph is oriented, hence the name. We show that there are no directed cycles in oriented VCSPs: all oriented VCSPs induce a partial order where the preferred assignment of a variable x_j depends only on the assignments of variables x_i with i lower than j in the partial order. This means that once we condition on all x_i with i lower than j, the preferred assignment of x_j is independent of all other variables – like in a smooth landscape.

Although having a directed or oriented constraint graph is a natural property to define subclasses of VCSPs, it is a hard property to recognize. Specifically, SubsetSum can be solved by determining the direction of constraints (Proposition B.1 for $i \leftrightarrow j$ and Proposition B.2 for both $i \to j$ and $j \to i$). Thus, given an *arbitrary* binary Boolean VCSP, the problem of checking if it is directed or oriented is coNP-complete (Corollary B.3). Furthermore, even if we are given a *directed* binary Boolean VCSP, checking if it is oriented is coNP-complete (Corollary B.4). The only silver lining is that if we take the maximum degree of the constraint graph of the VCSP-instance as a parameter then checking if a binary Boolean VCSP is directed or oriented is fixed-parameter tractable (Algorithm 2 and Appendix B.2).

Given the importance of the partial order of conditional independence but the difficulty of recognizing if a VCSP is oriented, with Definition 8 in Section 5, we abstract to a class of landscapes that we call conditionally-smooth. Just like smooth and semismooth landscapes, conditionally-smooth landscapes have only one local (and thus global) peak and there exist short ascents from any initial assignment to the peak (Proposition 9). In other words, just

[4] Our focus on easily checkable properties of polynomial-sized representations of problems instead of purely theoretical properties of the exponentially-large fitness landscapes implicit in the problems, is one of the big differences of our approach/results versus similar work in evolutionary computation [10, 12, 11].

as with semismooth landscapes, local search algorithms are effective on conditionally-smooth landscapes. Unlike directed or oriented VCSPs, however, we show how to recognize if an arbitrary binary Boolean VCSP is conditionally-smooth in polynomial time (Algorithm 3).

In Section 6, we show that – unlike general semismooth fitness landscapes (and like smooth landscapes and oriented VCSPs) – conditionally-smooth fitness landscapes are not just effective-for but also efficient-for-many local search algorithms.[5] Specifically, we show that conditionally-smooth fitness landscapes are solved efficiently by many local search algorithms including RandomAscent (Proposition 14); ShakenAscent (Proposition C.2); SimulatedAnnealing (Proposition 15); ZadehsRule, LeastRecentlyConsidered, and other history-based local search (Proposition C.6); AntipodalJump and JumpToBest (Proposition C.8); KerninghanLin (Proposition C.10); and RandomJump (Proposition 16) – all of these are examples a broader class of local search algorithms that we name *poly-bypass* local search algorithms (Definition 10). Conditionally-smooth landscapes are a more expressive class than the tree-structured binary Boolean VCSPs that (fully) capture the class of fitness landscapes that are efficient-for-all local search algorithms [32]. Thus, the conditionally-smooth structural property permits families of landscapes that are complex enough to break the efficiency of some popular local search algorithms that are not *poly-bypass* algorithms. In Section 7, we show that there are families of conditionally-smooth landscapes where finding the peak takes an exponential number of steps for SteepestAscent (Theorem 19) and takes a superpolynomial number of steps for RandomFacet (Theorem 23). Given that both SteepestAscent and RandomFacet [17] are often considered to be very good local search algorithms,[6] their inefficiency on conditionally-smooth landscapes tells us that conditionally-smooth landscapes are not a trivially easy to solve class of landscapes.

3 Effectiveness from smooth to semismooth fitness landscapes

We consider *assignments* $x \in \{0,1\}^n$ from the *n-dimensional hypercube* where x_i refers to the i-th entry of x. To refer to a substring with indexes $S \subseteq [n]$, we will write the partial assignment $y = x[S] \in \{0,1\}^S$. If we want to modify a substring with indexes $S \subseteq [n]$ to match a partial assignment $y \in \{0,1\}^S$ (or any string $y \in \{0,1\}^{|S|}$), we write $x[S \mapsto y]$. We abbreviate $x[\{i\} \mapsto b]$ by just $x[i \mapsto b]$. Two assignments are adjacent if they differ on a single bit: $x, y \in \{0,1\}^n$ are adjacent if there exists an index $i \in [n]$ such that $y = x[i \mapsto \bar{x}_i]$, where $\bar{x}_i := 1 - x_i$ is the negation of x_i.

This combinatorial structure of adjacent assignments can be combined with the numeric structure of a pseudo-Boolean function (that we call the *fitness function*) to create a *fitness landscape* [54, 30, 32]. A fitness landscapes f associates to each assignment x the integer $f(x)$ and a set of adjacent assignments. Given an assignment x, we let $\phi^+(x) = \{i \mid i \in$

[5] We do not focus on *efficient-for-all* (or more formally: landscapes where *all* ascents are polynomial length) because we think that this case is too strict and has been largely resolved. Kaznatcheev, Cohen and Jeavons [32] showed that all ascents have length $\leq \binom{n+1}{2}$ in fitness landscapes that are implementable by binary Boolean VCSP with tree-structured constraint graphs. Efficient-for-all cannot be pushed much further than Boolean trees: Kaznatcheev, Cohen and Jeavons [32] also gave examples of VCSPs with exponential ascents from (a) domains of size ≥ 3 and path-structured constraint graphs, or (b) Boolean domains and constraint graphs of pathwidth 2. This is why, we switch from the question of efficient-for-all to the question of *efficient-for-many*.

[6] RandomFacet is currently considered the best known algorithm for solving semismooth fitness landscapes. It finds the peak in any semismooth fitness landscapes in a superpolynomial but subexponential number of steps – even with the landscape given as a black-box [17]. In Corollary 22, we show that the family of semismooth landscapes that saturate this worst case behavior for RandomFacet are conditionally-smooth.

$[n]$ and $f(x[i \mapsto \bar{x}_i]) > f(x)\}$ be the indexes of variables that increase fitness when flipped (out-map) and $\phi^-(x) = \{i \mid i \in [n]$ and $f(x) \geq f(x[i \mapsto \bar{x}_i])\}$ be the set of indexes that lower or do not increase fitness when flipped (in-map). An assignment x is a *local peak* in f if for all y adjacent to x we have $f(x) \geq f(y)$ (i.e., if $\phi^+(x)$ is empty).[7] A sequence of assignments $x^0, x^1, \dots, x^T$ is an *ascent* if every x^{t-1} and x^t are adjacent with the latter having higher fitness (i.e., $\forall t \leq T$ $x^t = x^{t-1}[i \mapsto \overline{x_i^{t-1}}]$ and $i \in \phi^+(x^{t-1})$) and x^T is a local peak.

A fitness landscape f is *smooth* if for each $i \in [n]$ there exists an assignment x_i^* such that for all assignments x we have $f(x[i \mapsto x_i^*]) > f(x[i \mapsto \overline{x_i^*}])$. In other words, in a smooth landscape, each variable x_i has a preferred assignment x_i^* that is independent of how other variables are assigned. It is easy to see that all ascents are short (i.e, $\leq n$) in smooth fitness landscapes, so smooth landscapes are effective- and efficient-for-all local search algorithms.

We can relax the definition of smooth while maintaining effectiveness: a fitness landscape on the hypercube is a *semismooth fitness landscape* if every subcube is single peaked [30]. These are also known as completely unimodal pseudo-Boolean functions [21], or acyclic unique-sink orientations (AUSOs) of the hypercube [48, 50]. Semismooth fitness landscapes have a nice characterization in terms of the biological concept of sign epistasis [30, 44, 43, 53]:

▶ **Definition 1** (Kaznatcheev, Cohen and Jeavons [32]). *We say index i* sign-depends *on j in background x (and write $j \xrightarrow{x} i$) if $f(x[i \mapsto \bar{x}_i]) > f(x)$ and $f(x[\{i,j\} \mapsto \bar{x}_i\bar{x}_j]) \leq f(x[j \mapsto \bar{x}_j])$ If there is no background assignment x such that $j \xrightarrow{x} i$ then we say that i* does not sign-depend *on j (write $j \not\to i$). If for all $j \neq i$ we have $j \not\to i$ then we say that i is* sign-independent*.*

This terminology of "sign" comes from the observation that the sign of the fitness effect of a change in x_i depends on the value of x_j. Since the sign of the fitness effect of a change in x_i just indicates the preferred assignment of x_i (with a positive sign indicating $x_i^* = 1$ and negative indicating $x_i^* = 0$), it is easy to link this definition to smooth landscapes: a smooth landscape is a fitness landscapes where all indexes are sign-independent. Or stated in the negative, a smooth landscape is one without sign-dependence. We will relax this negative definition to get semismooth landscapes by instead excluding the concept of "reciprocal sign epistasis" [43, 44]:

▶ **Definition 2** (Poelwijk et al. [44]). *If there exists a background assignment x such that $j \xrightarrow{x} i$ and $i \xrightarrow{x} j$ then we say that i and j have* reciprocal sign epistasis *in background x (and write $i \overset{x}{\leftrightarrow} j$). If there is no background assignment x such that $i \overset{x}{\leftrightarrow} j$ then we say i and j* do not have reciprocal sign epistasis*, and use the symbol $i \not\leftrightarrow j$.*

If the background x is clear form context or not important then we drop the superscript in the above notations and just write $j \to i$ or $i \leftrightarrow j$. The absence of reciprocal sign epistasis is clearly necessary for a fitness landscape to be semismooth, but it is also sufficient:

[7] Note that local peaks are not necessarily "strict", they can have adjacent assignments of equal (but not greater) fitness. So what some call a "fitness plateau" is for us a collection of adjacent local peaks. This transforms how we think about popular "hard" landscapes like Needle from the evolutionary computation literature. A landscape f is a Needle landscape if there exists a single assignment x^{needle} such that $f(x^{\text{needle}}) = 1$ and for all other assignments $x \neq x^{\text{needle}}$ $f(x) = 0$. We use the scare quotes around hard because for our definition of local peaks, Needle landscapes are trivial: every assignment is either a local peak (for x^{needle} and any $x \neq x^{\text{needle}}$ not directly adjacent to x^{needle}) or directly adjacent to a local peak (for y directly adjacent to x^{needle}) and are thus solved by any local search algorithm in at most one step. Needle landscapes are only hard if we are looking for a global peak instead of any local peak. So from the perspective of our effective vs efficient distinction, many intractability results that rely on the difficulty of 'navigating fitness plateaus or crossing fitness valleys' are statement of ineffectiveness rather than inefficiency. This is one of the big differences between our approach and results, and the approach and results in the evolutionary computation literature [52, 25, 10, 12, 11, 23].

▶ **Proposition 3** (Hammer et al. [21], Poelwijk et al. [44], and Kaznatcheev [30]). *A fitness landscape f on n bits is semismooth if and only if for all $i, j \in [n]$ $i \not\leftrightarrow j$.*

Conveniently, semismooth landscapes always have a short ascent to the unique peak:

▶ **Proposition 4** (Hammer et al. [21], and Kaznatcheev [30]). *A semismooth fitness landscape has only one local (thus global) peak at x^* and given any initial assignment x^0, there exists an ascent to x^* of Hamming distance (i.e., length $\leq n$).*

Unlike smooth landscapes, however, not all ascents in semismooth fitness landscapes are short, and finding and following a short ascent is not easy. The Klee-Minty cube [36] is a construction of a semismooth fitness landscape on $\{0,1\}^n$ with an ascent of length $2^n - 1$. As for local search algorithms, constructions exists such that semismooth landscapes are not tractable in polynomial time by various popular ascent-following local search algorithms including SteepestAscent [13, 27, 30], RandomAscent [22, 39], RandomFacet [17], jumping rules [47], and many others. Thus, local search is not efficient on *all* semismooth fitness landscapes. So we start our search for single-peaked landscapes that are efficient-for-many local search algorithms by looking for natural subclasses of semismooth fitness landscapes.

4 Representing semismooth fitness landscapes by directed VCSPs

As blackboxes, fitness landscapes are unwieldy exponentially large objects, so we open the blackbox by representing landscapes by instances of valued constraint satisfaction problems (VCSPs) [32]. We can then find which natural subclass of VCSPs represents semismooth fitness landscapes along with an algorithms for checking when a given VCSP-instance has these properties. A Boolean VCSP-instance is a set of constraint weights $\mathcal{C} = \{c_S\}$ where each weight $c_S \in \mathbb{Z} \setminus \{0\}$ has a *scope* $S \subseteq [n]$. This set of constraints *implements* a pseudo-Boolean function:

$$f(x) = \sum_{c_S \in \mathcal{C}} c_S \prod_{j \in S} x_j. \tag{1}$$

If $|S| \leq 2$ for all constraints then we say the VCSP is binary. We also view every binary VCSP-instance $\mathcal{C}$ as a *constraint graph* with edges $\{i,j\} \in E(\mathcal{C})$ if $c_{ij} := c_{\{i,j\}} \in \mathcal{C}$ and a neighbourhood function $N_{\mathcal{C}}(i) = \{j \mid \{i,j\} \in E(\mathcal{C})\}$. This constraint graph is a way of representing where the potential sign-dependencies are in the fitness landscape.

Given a binary Boolean VCSP $\mathcal{C}$ on the whole n-dimensional hypercube, it is sometimes useful to consider the binary Boolean VCSP $\mathcal{C}'$ restricted to just a subset of the indexes $R \subseteq [n]$ with the other variables in $S := [n] \setminus R$ fixed to some assignment $y \in \{0,1\}^S$ (i.e., restricted to the face $\{0,1\}^R y$). This restricted VCSP $\mathcal{C}'$ will have the same binary constraint as $\mathcal{C}$ (i.e., if $i, j \in R$ and $c_{ij} \in \mathcal{C}$ then $c_{ij} \in \mathcal{C}'$) but the unary constraints will change to what we call the *effective unaries* that "absorb" the binary constraints that cross the R-S cut:

▶ **Definition 5.** *Given a variable index i with neighbourhood $N(i)$ and $R \subseteq N(i)$ a set of indices we do not want to fix, we define the* effective unary $\hat{c}_i(x, R) = c_i + \sum_{j \in N(i) \setminus R} x_j c_{ij}$. *For simplicity, we write $\hat{c}_i(x)$ for $\hat{c}_i(x, \emptyset)$.*

If x and y are two assignments with $x[N(i) \setminus R] = y[N(i) \setminus R]$ then $\hat{c}_i(x, R) = \hat{c}_i(y, R)$. We use this to overload $\hat{c}_i$ to partial assignments: given $T \supseteq N(i) \setminus R$ and $y \in \{0,1\}^T$, we interpret $\hat{c}_i(y, R)$ as $\hat{c}_i(y 0^{[n] \setminus T}, R)$. From this, it is easy to see that restricting our VCSP $\mathcal{C}$ to $\mathcal{C}$' on the face $\{0,1\}^R y$ will change the unaries to $c'_i = \hat{c}_i(y, R)$ (for $i \in R$).

In terms of representation of smooth landscapes, it is clear than if all constraints in a VCSP are unary then the resulting fitness landscape is smooth. But even given a binary Boolean VCSP-instance with non-unary constraints, it is easy to check if that VCSP-instance implements a smooth landscape by checking if the unary constraints are "sufficiently large" compared to the relevant binary constraints. Or more generally, given any partial assignment $y \in \{0,1\}^S$ it is easy to check if the face $\{0,1\}^{[n]-S}y$ is smooth using the effective unaries. We will do this check one variable at a time. For a variable x_i to have a preferred assignment that is independent of how other variables are assigned, its unary must dominate over the binary constraints that it participates in – its unary must have big magnitude. For example, suppose that $c_i > 0$ then x_i will prefer to be 1 if all of its neighbours are 0. This preference must not change as any x_j with $j \in N(i)$ change to 1s. Since x_j with $c_{ij} < 0$ are the only ones that can lower x_i's preference for 1, this becomes equivalent to checking if $c_i > \sum_{j \in N(i) \text{ s.t. } c_{ij}<0} |c_{ij}|$. There exists a combination of x_js that flip x_i's preference if and only if this inequality is violated. In moving from this particular example to the general case, we can also generalize our algorithm to check not only if x_i has an independent preference in the whole landscape but also if it has a conditionally independent preference conditional on some variables with indexes $j \in S$ not being able to vary and fixed to some $y \in \{0,1\}^S$. To do this we only need to replace c_i by the effective unary of Definition 5 $\hat{c}_i = \hat{c}_i(y, N(i) \setminus S) := c_i + \sum_{j \in S \cap N(i)} y_j c_{ij}$ and consider c_{ij}s with $\text{sgn}(\hat{c}_i) \neq \text{sgn}(c_{ij})$ and $j \in N(i) \setminus S$. We encode this in ConditionallySignIndependent$(\mathcal{C}, i, S, y)$ of Algorithm 1.

Algorithm 1 ConditionallySignIndependent$(\mathcal{C}, i, S, y)$.

Require: VCSP-instance $\mathcal{C}$, variable index i, subset of indexes $S \subset [n]$, an assignment y.
1: Compute $\hat{c} \leftarrow \hat{c}_i(y, N(i) \setminus S) = c_i + \sum_{j \in S \cap N(i)} y_j c_{ij}$
2: Compute $c_{\text{flip}} \leftarrow \sum_{j \in N(i) \setminus S} c_{ij}$ when $\text{sgn}(c_{ij}) \neq \text{sgn}(\hat{c})$.
3: **return** $|\hat{c}| > |c_{\text{flip}}|$

Given a binary Boolean VCSP-instance $\mathcal{C}$, this algorithms runs in time linear in the maximum degree $\Delta(\mathcal{C})$ of the constraint graph. To check if $\mathcal{C}$ implements a smooth landscape, run ConditionallySignIndependent$(\mathcal{C}, i, \emptyset, 0^n)$ on every i and return the conjunction of their outputs in $O(\Delta(\mathcal{C})n)$ time.

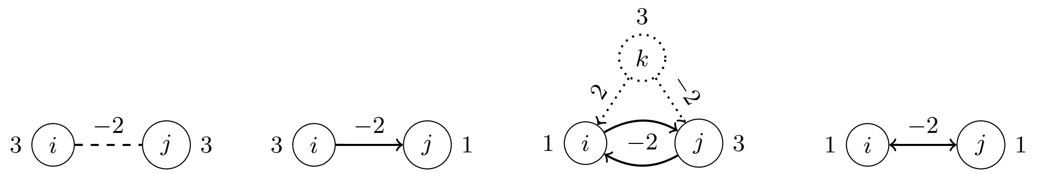
Now let us return to view the constraint graph as a way of represetning where the potential sign-dependencies are in a fitness landscape. If VCSP-instance with binary constraints is smooth then this tells us that the edges in the constraint-graph did not actually encode any actual sign-dependence. This idea can be taken further by converting edges to arcs and assigning "directions" to the binary constraint c_{ij} based on how its weight compares to the effective unaries $\hat{c}_i$ across various background assignments (see Figure 1 for illustration):

▶ **Definition 6.** *For* VCSP*-instance $\mathcal{C}$ we set the* arcs $A(\mathcal{C})$ *such that for $\{i,j\} \in E(\mathcal{C})$:*

(1) *we set $i \leftrightarrow j \in A(\mathcal{C})$ if there exists an assignment x with $|c_{ij}| > \max\{|\hat{c}_i(x,\{i,j\})|, |\hat{c}_j(x,\{i,j\})|\}$ with $sgn(c_{ij}) \neq sgn(\hat{c}_i(x,\{i,j\}))$ and $sgn(c_{ij}) \neq sgn(\hat{c}_j(x,\{i,j\}))$.*

(2) *otherwise we set:* ***(a)*** *$i \to j \in A(\mathcal{C})$ if there exists an assignment y such that $|c_{ij}| > |\hat{c}_j(y,\{i,j\})|$ with $sgn(c_{ij}) \neq sgn(\hat{c}_j(y,\{i,j\}))$, and* ***(b)*** *$i \leftarrow j \in A(\mathcal{C})$ if there exists an assignment z such that $|c_{ij}| > |\hat{c}_i(z,\{i,j\})|$ with $sgn(c_{ij}) \neq sgn(\hat{c}_i(z,\{i,j\}))$.*

From this, each edge $\{i,j\} \in E(\mathcal{C})$ is assigned only one of the three kinds of arcs: $\{i \leftrightarrow j\}$ or $\{i \to j$ and/or $i \leftarrow j\}$ or $\{\}$. In Figure 1, we provide some minimal prototypical examples of VCSP-instances that have (a) no direction for the constraint with scope $\{i,j\}$,



(a) $\{\} \in A(\mathcal{C})$. **(b)** $i \to j \in A(\mathcal{C})$. **(c)** $i \to j, i \leftarrow j \in A(\mathcal{C})$. **(d)** $i \leftrightarrow j \in A(\mathcal{C})$.

Figure 1 Four VCSP instances illustrating the different arc directions of Definition 6. Weights of unary constraints are next to nodes and weights of binary constraints are above the edges.

(b) $i \to j \in A(\mathcal{C})$, (c) both $i \to j, i \leftarrow j \in A(\mathcal{C})$, or (d) $i \leftrightarrow j \in A(\mathcal{C})$. Most of these examples are on two variables. Since it is impossible to have both both $i \to j$ and $i \leftarrow j$ without a $k \in N(i) \cup N(j)$ (Proposition A.4), the minimal instance in Figure 1c requires three variables. It is easy to check that the overloading of the sign-dependence and reciprocal sign epistasis symbols is appropriate. Most importantly, the fitness landscape will have reciprocal sign epistasis if and only if the VCSP-instance $\mathcal{C}$ implementing it has $\{i \leftrightarrow j\} \in A(\mathcal{C})$.

The advantage of Definition 6 over black-box features of the fitness landscape is its statement in terms of properties of just the VCSP-instance. This means, for example, that if we want to check the potential direction of a constraint with scope $\{i, j\} \in E(\mathcal{C})$ with $c_{ij} < 0$ then we need to compare it to the effective unaries $\hat{c}_i(x, \{i, j\})$ and $\hat{c}_j(x, \{i, j\})$ for various choices of x. If we find an x such that both $|c_{ij}| > \hat{c}_i(x, \{i, j\}) > 0$ and $|c_{ij}| > \hat{c}_j(x, \{i, j\}) > 0$ are simultaneously satisfied then we output that $i \leftrightarrow j \in A(\mathcal{C})$. If we only ever satisfy one or fewer of these equations for all choices of x then we need to output a subset of $\{i \to j, i \leftarrow j\}$ depending on which of the comparisons was true. Outputing $\{\}$ if neither comparison was ever satisfied, $i \to j$ if only the first was satisfied, $i \leftarrow j$ if only the second, and $\{i \to j, i \leftarrow j\}$ if each was true in a different background. Finally, to be fixed-parameter tractable, it is important to use that $\hat{c}_i(x, R) = \hat{c}_i(y, R)$ when $x[N(i) \setminus R] = y[N(i) \setminus R]$ to limit our search to just partial assignments $x \in \{0,1\}^{N(i) \cup N(j) \setminus \{i,j\}}$. We formalize this as Algorithm 2:

Algorithm 2 ArcDirection$(\mathcal{C}, i, j)$.

Require: VCSP-instance $\mathcal{C}$ and variable indexes i and j.
Initialize $A \leftarrow \{\}$
for $x \in \{0,1\}^{N(i) \cup N(j) \setminus \{i,j\}}$ **do**
  Set $B \leftarrow \{\}$, $\hat{c}_i = \hat{c}_i(x, \{i, j\})$, $\hat{c}_j = \hat{c}_j(x, \{i, j\})$
  if $|\hat{c}_i| < |c_{ij}|$ and $\mathrm{sgn}(\hat{c}_i) \neq \mathrm{sgn}(c_{ij})$ **then** $B \leftarrow B \cup \{i \to j\}$
  if $|\hat{c}_j| < |c_{ij}|$ and $\mathrm{sgn}(\hat{c}_j) \neq \mathrm{sgn}(c_{ij})$ **then** $B \leftarrow B \cup \{i \leftarrow j\}$
  if $|B| < 2$ **then** $A \leftarrow A \cup B$ **else return** $\{i \leftrightarrow j\}$
end for
return A

The resulting worst-case runtime is $2^{O(\Delta(\mathcal{C}))}$, or an overall runtime of $2^{O(\Delta(\mathcal{C}))}O(\Delta(\mathcal{C})n)$ to determine the direction of all arcs in the VCSP-instance $\mathcal{C}$. This is fixed-parameter tractable when parameterized by the maximum number of constraints incident on a variable $(\Delta(\mathcal{C}))$. A fully polynomial time algorithm for finding arc directions is unlikely given that the questions "is $i \leftrightarrow j \in A(\mathcal{C})$?" (Proposition B.1) and "are both $i \to j$ and $j \to i$ in $A(\mathcal{C})$?" (Proposition B.2) are NP-complete by reduction from SubsetSum. However, we can still define two natural subclasses of VCSPs by restricting the bidirected constraint graph:

▶ **Definition 7.** *We say that a VCSP instance $\mathcal{C}$ is* directed *if $\mathcal{C}$ has no bidirected arcs. We say that a directed $\mathcal{C}$ is* oriented *if it has at most one arc for every pair of variables $i \neq j$.*

In Appendix A, we extend Proposition 3 to representations to show that a quadratic pseudo-Boolean f is semismooth if and only if the corresponding VCSP is directed (Proposition A.1), any triangle-free directed VCSP is oriented (Theorem A.3), and oriented VCSPs are always acyclic (Proposition A.5). Theorem A.3 and Proposition A.5 let us view oriented VCSPs as a kind of generalization of the tree-structured VCSPs that Kaznatcheev, Cohen, and Jeavons [32] showed are efficient-for-all local search algorithms. Specifically, we replace the undirected acyclicity of trees by the directed acyclicity of DAGs. Unfortunately, checking if a VCSP is directed or oriented is coNP-complete (Corollary B.3 and Corollary B.4).

5 From oriented VCSPs to conditionally-smooth fitness landscapes

One of these nice features of our two natural subclasses of directed and oriented VCSPs is that they let us capture when fitness landscapes are both effective-for-all and efficient-for-many local search algorithms. Directed VCSPs capture the semismooth fitness landscapes that are effective-for-all local search algorithms, but that have no known efficient algorithms. Oriented VCSPs are then a further restriction to capture those semismooth fitness landscapes that are efficient-for-many local search algorithms. By focusing on the main aspect of acyclicity that makes oriented VCSP tractable, we can generalize that class to a slightly larger class of conditionally-smooth fitness landscapes that are also single-peaked (and so effective-for-all local search algorithms) but also recognizable from the implementing VCSP-instance.

The acyclicity of an oriented VCSP-instance (Proposition A.5) lets us define a strict partial order $\prec$ as the transitive closure of the constraint graph and the corresponding down sets $\downarrow j = \{i \mid i \prec j\}$. As we show later, what makes oriented VCSPs efficient for many local search algorithms is that this order respects conditional independence (from Algorithm 1). Specifically Proposition A.7: for oriented VCSP-instance $\mathcal{C}$, $\forall y \in \{0,1\}^{\downarrow j}$ ConditionallySignIndependent$(\mathcal{C}, j, \downarrow j, y) = \textsc{True}$. Proposition A.7 is a powerful defining feature of oriented VCSPs, but it is more powerful (and restrictive) than necessary for efficiency. To get our definition of the larger class of *conditionally-smooth fitness landscapes*, we can relax from conditional sign independence for any background $y \in \{0,1\}^{\downarrow j}$ to just a single background $y = x^*[\downarrow j]$ where x^* is the peak of a single-peaked landscape:

▶ **Definition 8.** *Given a strict partially ordered set* $([n], \prec)$ *and* $\downarrow j = \{i \mid i \prec j\}$*, we call a fitness landscape* f *on* $\{0,1\}^n$ *a* $\prec$-smooth fitness landscape *with optimum* x^* *when for all* $j \in [n]$ *and* $x \in \{0,1\}^n$*, if* $x[\downarrow j] = x^*[\downarrow j]$ *then* $f(x[j \mapsto x^*_j]) > f(x[j \mapsto \overline{x^*_j}])$.[8] *We say* f *is a* conditionally-smooth *fitness landscape if there exists some* $\prec$ *such that* f *is* $\prec$*-smooth.*

Conditionally-smooth landscapes generalize both oriented VCSPs and recursively combed AUSOs (Definition A.8). Although conditionally-smooth landscapes are not always semismooth, they are singled peaked and have direct ascents from any assignment to the peak:

▶ **Proposition 9.** *A conditionally-smooth fitness landscape has only one local (thus global) peak at* x^* *and given any initial assignment* x^0*, there exists an ascent to* x^* *of Hamming distance (i.e., length* $\leq n$*).*

Proof. Let f be a $\prec$-smooth landscape. Take any assignment $x \neq x^*$ and let i be the $\prec$-smallest index such that $x_i \neq x^*_i$. Then $f(x[i \mapsto \overline{x_i}]) > f(x)$. ◀

[8] Thus $\emptyset$-smooth fitness landscapes are just smooth fitness landscapes.

Unlike with directed or oriented VCSPs, we can check if a binary Boolean VCSP instance implements a fitness landscape that is conditionally smooth in polynomial time. In fact, if the VCSP does implement a conditionally-smooth fitness landscape then our recognition Algorithm 3 even returns a partial order $\prec$ and the preferred assignment x^* such that $\mathcal{C}$ is $\prec$-smooth. The overall approach is similar to checking if a VCSP is smooth (i.e., $\emptyset$-smooth) with the difference being that subsequent calls to $\mathsf{ConditionallySignIndependent}(\mathcal{C}, i, S, y)$ adjust the set of fixed variables S and background assignment y based on previous calls. This gives the $\mathsf{ConditionallySmooth}(\mathcal{C})$ algorithm (Algorithm 3). This algorithm calls ConditionallySignIndependent at most $\binom{n}{2}$ times for an overall runtime of $\binom{n}{2}\Delta(\mathcal{C})$ (for details of the analysis, see Appendix B.3).

Algorithm 3 $\mathsf{ConditionallySmooth}(\mathcal{C})$. Checking if a VCSP implements a $\prec$-smooth landscape.

Require: VCSP-instance $\mathcal{C}$

```
Initialize ≺ ← ∅, S ← ∅, x* ← 0^n
while S is not [n] do
    T ← ∅ and x^next ← x*
    for i ∈ [n] \ S do
        if ConditionallySignIndependent(C, i, S, x*) then
            if ĉ_i(x*, N(i)\S) > 0 then x_i^next ← 1
            T ← T + {i}
        end if
    end for
    if T is empty then return FALSE
    ≺ ← ≺ + S × T, S ← S + T, x* ← x^next
end while
return (≺, x*)
```

6 Efficient local search in conditionally-smooth landscapes

Local search starts at some initial assignment x^0 and takes steps to assignments $x^1, x^2, \ldots, x^T$ with x^T as a local peak. If, additionally, for every $0 \leq t < T$ we have $f(x^{t+1}) > f(x^t)$ then local search followed an ascent. For an arbitrary local search algorithm A we let $\mathsf{A}^t_f(x)$ denote t steps of A from x on fitness landscape f.

▶ **Definition 10.** *Given a polynomial $p(n)$, we say that an ascent-following*[9] *local search algorithm A is a $p(n)$*-bypass ascent following local search algorithm *if given any f, any corresponding run $x^0, x^1, \cdots, x^T$ of A on f, and all $s \in [T - p(n)]$, we have that $\cap_{t=s}^{s+p(n)} \phi^+(x^t)$ is empty (with high probability, for randomized algorithms). If some polynomial $p(n)$ exists, but its specific form is not important to us, then we say that the algorithm is a* poly-bypass *ascent-following local search algorithm.*

[9] One can modify Definition 10 to apply to ascent-biased algorithms (Appendix C.2) or jumping algorithms (Appendix C.4) instead of just ascent-following local search algorithms – but this makes the definition more unwieldy and unintuitive. We want to present poly-bypass algorithms as just a simple warm-up example to get at the main ideas of our later proofs. So here we focus on just ascent-following poly-bypass for simplicity, and go into the nuance of stochastic, ascent-biased, and jumping rules when we focus on proving even tighter bounds for specific popular examples of those local search algorithms.

For a $p(n)$-bypass algorithm, any index i that could have flipped to increase fitness at some step s (i.e., $i \in \phi(x^s)$), will become an index that cannot be flipped to increase fitness at some point in the subsequent $p(n)$ steps. This can happen either because the variable with index i flips or because some other indexes flip in a way that makes a flip at i no longer fitness increasing. In other words, no potential fitness-increasing flip was bypassed for more than $p(n)$ steps. Hence, the name.

Now, we can easily show that conditionally-smooth landscapes are efficient for $p(n)$-bypass local search algorithms.

▶ **Theorem 11.** *On a $\prec$-smooth landscape f on n bits, given any initial assignment x^0 at Hamming distance $m(\leq n)$ to the fitness peak x^*, any $p(n)$-bypass local search algorithm starting at initial assignments x^0 takes at most $m \cdot p(n)$ steps to find the peak.*

Proof. We prove this by induction on m. For $m = 0$, $x^0 = x^*$ and local search finishes without taking a step. Now we assume the inductive hypothesis is true for Hamming distance $\leq m - 1$ and show it is true if the Hamming distance between x^0 and x^* is m. Let i be the $\prec$-smallest index such that $x^0_i \neq x^*_i$. Since the algorithm is $p(n)$-bypass, sometime by the $p(n)$th step, it will be at an assignment such that the ith bit doesn't want to flip. Since i was $\prec$-smallest index such that $x^0_i \neq x^*_i$ that means that the ith will always want to be in state x^*_i and once it flips, it won't flip back because the algorithm is ascent-following, so $x^{p(n)}_i = x^*_i$. Thus $x^{p(n)}$ has at least one more variable assignment in common with x^* than x^0 did, so the Hamming distance between $x^{p(n)}$ and x^* is $\leq m - 1$. By the inductive hypothesis, the $p(n)$-bypass algorithm will find the fitness peak in at most $(m - 1) \cdot p(n)$ steps starting from $x^{p(n)}$. This gives us a total number of steps of less than $m \cdot p(n)$. ◀

For a specific local search algorithm, the bound in Theorem 11 can be rather loose. To provide tighter bounds on the number of steps taken by many popular local search algorithms, we will need a bit more fine-grained notation and more careful proofs. However, much like the proof of Theorem 11, all these proofs will rest on showing a bound on how long it takes to fix $\prec$-minimal indexes that disagree with x^* and repeatedly applying that bound until the assignments agree (Lemma 13).

To show that conditionally-smooth landscapes (and thus also oriented VCSPs) are efficient for many popular local search algorithms with tighter bounds, we need to state Lemma 13 precisely. This requires us to create a partition of the n indexes and show that many local search algorithms quickly and "permanently" fix variables with indexes progressing along the levels of this partition. Given a poset $([n], \prec)$ and variable index $i \in [n]$, define the upper set of i as $\uparrow i = \{j \mid i \preceq j\}$. We partition $[n]$ into height$([n], \prec)$-many *level sets* defined as $S_l = \{i \mid \text{height}(\uparrow i, \prec) = l\}$ where height$(S, \prec)$ is the height of the poset $(S, \prec)$. Additionally we define $S_0 = \emptyset$, $S_{<l} = \bigcup_{k=0}^{l-1} S_k$, and $S_{>l} = \bigcup_{k=l+1}^{n} S_k$. We relate these partitions to an assignment x via a conditionally-smooth landscape specific refinement of in- and out-maps:

▶ **Definition 12.** *Given a $\prec$-smooth landscape f on n bits and x an assignment, we define the maps $\phi^\ominus, \phi^\oplus : \{0,1\}^n \to 2^{[n]}$ by:*

- *$i \in \phi^\ominus(x)$ if for all $j \preceq i$ we have $j \in \phi^-(x)$ (we say i is* correct *at x), and*
- *$i \in \phi^\oplus(x)$ if $i \in \phi^+(x)$ but for all $j \prec i$ we have $j \in \phi^-(x)$ (we say i is at* border *at x).*

Note that given any ascent $x^0, x^1, \ldots, x^T$ in a conditionally-smooth landscape, we have $\phi^\ominus(x^0) \subseteq \phi^\ominus(x^1) \subseteq \ldots \phi^\ominus(x^T) = [n]$. We refer to the set $\overline{\phi^\ominus(x)} := [n] - \phi^\ominus(x)$ as the *free indices* at x. For an assignment x, define height$_f(x)$ (and width$_f(x)$) as the height (and width) of the poset $(\overline{\phi^\ominus(x)}, \prec)$. Note that for x with $\text{height}(x) = l$,[10] we have $S_{>l} \subseteq \phi^\ominus(x)$, $S_l \subseteq \phi^\ominus(x) \cup \phi^\oplus(x)$, and, for $x \neq x^*$, $\phi^\oplus(x) \cap S_l \neq \emptyset$. In other words, if x is at height $l \neq 0$ then all the variables with indexes at higher levels are set correctly, all free indexes at level l are at the border, and at least one index at level l is free. This also means that $\overline{\phi^\ominus(x[\phi^\oplus(x) \mapsto \overline{x[\phi^\oplus(x)]}])} \subseteq S_{<l}$.

▶ **Lemma 13.** *Given a $\prec$-smooth fitness landscape f on n bits and any assignment x with $l := \mathit{height}_f(x)$, let $Y : \Omega \times \mathbb{N} \to \{0,1\}^n$ be a stochastic process such that $Y_t \sim \mathsf{A}^t_f(x)$ is the random variable of outcomes of t applications of the local search step algorithm A^1_f starting from x and let the random variable $\tau_{<l}(\omega) := \inf\{t \mid \overline{\phi^\ominus(Y_t(\omega))} \in S_{<l}\}$ be the number of steps to decrement height. If the expected number of steps to decrement height is:*

$$\mathbb{E}\{\tau_{<l}(\omega)\} \leq p(n,l) \leq q(n) \tag{2}$$

then the expected total number of steps taken by A *to find the peak from an initial assignment x^0 is $\leq \sum_{l=1}^{\mathit{height}_f(x^0)} p(n,l) \leq \mathit{height}_f(x_0) q(n) \leq \mathit{height}(\prec) q(n)$.*

Proof. This follows by induction on $\text{height}_f(x^0)$ and linearity of expected value. ◀

We use Lemma 13 to show that conditionally-smooth fitness landscapes are efficient-for-many local search algorithms, whether they follow ascents (RandomAscent; Proposition 14), occasionally step to adjacent assignments of lower fitness (SimulatedAnnealing; Proposition 15), or even if they step to non-adjacent assignments of higher fitness (RandomJump; Proposition 16).

Since Lemma 13 is expressed in terms of a stochastic process, we begin by applying it to the prototypical stochastic local search algorithm: RandomAscent [22, 29, 39]. Given an assignment x, the step $\mathsf{RandomAscent}^1_f$ simply returns a fitter adjacent assignment uniformly at random. Formally, if $Y^{\text{RA}}(x) \sim \mathsf{RandomAscent}^1_f(x)$ then:

$$\Pr\{Y^{\text{RA}}(x) = x[i \mapsto \bar{x}_i]\} = \begin{cases} \frac{1}{|\phi^+(x)|} & \text{if } i \in \phi^+(x) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

By bounding the expected number of steps to decrement height (Lemma C.1) and applying Lemma 13 we bound the expected total number of steps for RandomAscent:

▶ **Proposition 14** (Appendix C.1). *On a $\prec$-smooth landscape f on n bits, the expected number of steps taken by* RandomAscent *to find the peak from initial assignment x^0 is:*

$$\leq |\overline{\phi^\ominus(x^0)}| + \mathit{width}_f(x^0)(1 + \log \mathit{width}_f(x^0)) \binom{\mathit{height}_f(x^0) - 1}{2} \tag{4}$$

$$\leq n + \mathit{width}(\prec)(1 + \log \mathit{width}(\prec)) \binom{\mathit{height}(\prec) - 1}{2}. \tag{5}$$

Good bounds are also possible for deterministic ascent-following algorithms like various history-based pivot rules [2, 8, 14, 55] that we discuss in Appendix C.3. But even if an ascent-following algorithm is not efficient on conditionally-smooth landscapes, it can become efficient

[10] If we define X_l as the set of all assignments x with $\text{height}(x) = l$, it is important to note that the resulting sets $X_0, \ldots, X_{\text{height}(\prec,[n])}$ are not necessarily monotonic in fitness. There can exist x at $\text{height}(x) = l$ and y, z with height $l - 1$ such that $f(y) < f(x) < f(z)$. Thus, even if we expressed our results in terms of sets of assignments (i.e., in the space of the fitness landscape rather than the more compact space of the representation that we use) our approach in this paper is still *not* a fitness-level method of the sort often used in the analysis of randomized search heuristics [52, 12, 23].

through combination with RandomAscent. Any ascent-following local search algorithm A can be made into ϵ-shaken $-$ A like this: with probability $1-\epsilon$ take take a step according to A, and with probability ϵ take a step according to RandomAscent. Since variables with indexes in $\phi^{\ominus}(x)$ will never be unflipped by ascents, this combined algorithm's expected runtime is less than an $1/\epsilon$-multiple of the bound in Proposition 14 (Proposition C.2).

Lemma 13 also applies to algorithms that occasionally take downhill steps like simulated annealing [1]. Formally, if $Y^{\text{SA}}_{t+1}(x) \sim \mathsf{SimulatedAnnealing}^1_f(x^t)$ then

$$\Pr\{Y^{\text{SA}}_{t+1} = y\} = \begin{cases} \frac{1}{n} & \text{if } i \in \phi^+(x) \text{ and } y = x[i \mapsto \overline{x_i}] \\ \frac{1}{n} \cdot r_t(f(x^t) - f(x^t[i \mapsto \overline{x^t_i}])) & \text{if } i \in \phi^-(x) \text{ and } y = x[i \mapsto \overline{x_i}] \\ 1 - \frac{|\phi^+(x)|}{n} - Z\frac{|\phi^-(x)|}{n} & \text{if } y = x \end{cases} \tag{6}$$

where the downstep probability $r_t(\Delta f) \to 0$ monotonically as $t \to \infty$ for any $\Delta f > 0$ and $Z = \frac{1}{|\phi^-(x)|}\sum_{i \in \phi^-(x)} r_t(f(x^t) - f(x^t[i \mapsto \overline{x^t_i}]))$ is the average downstep probability. A popular choice of downstep probability is $r_t(\Delta f) = \exp(\frac{\Delta f}{K(t)})$ with $K(t)$ a sequence of temperatures strictly decreasing to 0. But any downstep probability can be used to define a burn-in time $\tau^\alpha = \inf\{t \mid r_t(1) \leq \frac{\alpha}{n}\}$ that is a property of the algorithm and independent of the particular problem-instance (i.e., independent of the fitness landscape).

▶ **Proposition 15** (Appendix C.2)**.** *On a conditionally-smooth fitness landscape f on n bits, the expected number of steps taken by* SimulatedAnnealing *to find the peak is* $\leq \tau^\alpha + n^2\frac{(\exp(\alpha)-1)}{\alpha}$ *where* $\tau^\alpha = \inf\{t \mid r_t(1) \leq \frac{\alpha}{n}\}$.

RandomAscent and SimulatedAnnealing move to adjacent assignments, changing at most one variable at a time, and so require at least a linear number of steps. But Lemma 13 also applies to local search algorithms that jump to non-adjacent assignments. Some such algorithms like AntipodalJump, JumpToBest, and RandomJump are especially popular on semismooth fitness landscapes since flipping any combination of variables with indexes in $\phi^+(x)$ results in a fitness increasing step [24, 29, 47]. But algorithms that take steps to non-adjacent assignments are also used in other contexts like the Kernighan-Lin heuristic for MaxCut [35]. Such algorithms can exploit short but wide partial orders. For example on $\prec$-smooth landscapes, the number of steps taken by deterministic algorithms like AntipodalJump, JumpToBest and Kernighan-Lin are independent of width($\prec$), taking at most height($\prec$)-steps (Propositions C.8 and C.10). Picking uniformly at random which subset of improving indexes to include in a jump – as done by the like RandomJump algorithm (see Equation (17) in Appendix C.4) – requires only a $\log(\text{width}(\prec))$-factor more steps than the deterministic jump rules:

▶ **Proposition 16** (Appendix C.4)**.** *Let f be a conditionally-smooth landscape and x^0 some assignment. The expected number of steps that* RandomJump *takes to find the peak is at most* $(\log(\mathit{width}_f(x^0)) + 2)\mathit{height}_f(x^0)$.

Thus, we see that many local search algorithms can efficiently find the peak in conditionally-smooth fitness landscapes in a quadratic or fewer number of steps.

7 Inefficient local search in conditionally-smooth landscapes

Finally, in this section, we want to argue that the effectiveness of many local search algorithms on conditionally-smooth landscapes is not a trivial observation. We do this by showing that conditionality-smooth landscapes are not tractable for some algorithms that are often considered to be very good local search algorithms (but that happen to not be poly-bypass).

Specifically, we will show that oriented VCSPs can represent families of landscapes where SteepestAscent takes an exponential number of steps; and that conditionally-smooth landscapes include families of semismooth landscapes where RandomFacet takes a superpolynomial number of steps. In other words, conditionally-smooth landscapes are expressive enough to contain very complicated kinds of fitness landscapes.

While studying Hopfield networks, Haken and Luby [20] created a family of binary Boolean VCSPs with exponentially long steepest ascents. Here we show that the Haken-Luby VCSP is oriented and has pathwidth 3. The Haken-Luby VCSP has variables with indexes $\{(k,i) | k \in [n], i \in [7]\}$ where each set of seven variables $\{(k,1),(k,2),\dots(k,7)\}$ forms a gadget. The gadgets form a chain by connecting $(k,7)$ to $(k-1,1)$. The magnitude of the constraints on the kth module is roughly proportional to $M_k = \frac{5}{6}(6^k - 6)$ and the exact constraints, with the exception of $c_{(n,1)} = (6M_n + 24)K$, where $K = 2n+1$ is constant for the instance, are given by Figure 2. For example, $c_{(k,5),(k,7)} = -(2M_k + 4)K$ and $c_{(k,7),(k-1,1)} = M_k K$. In Figure 2 we also included the direction of the constraints.

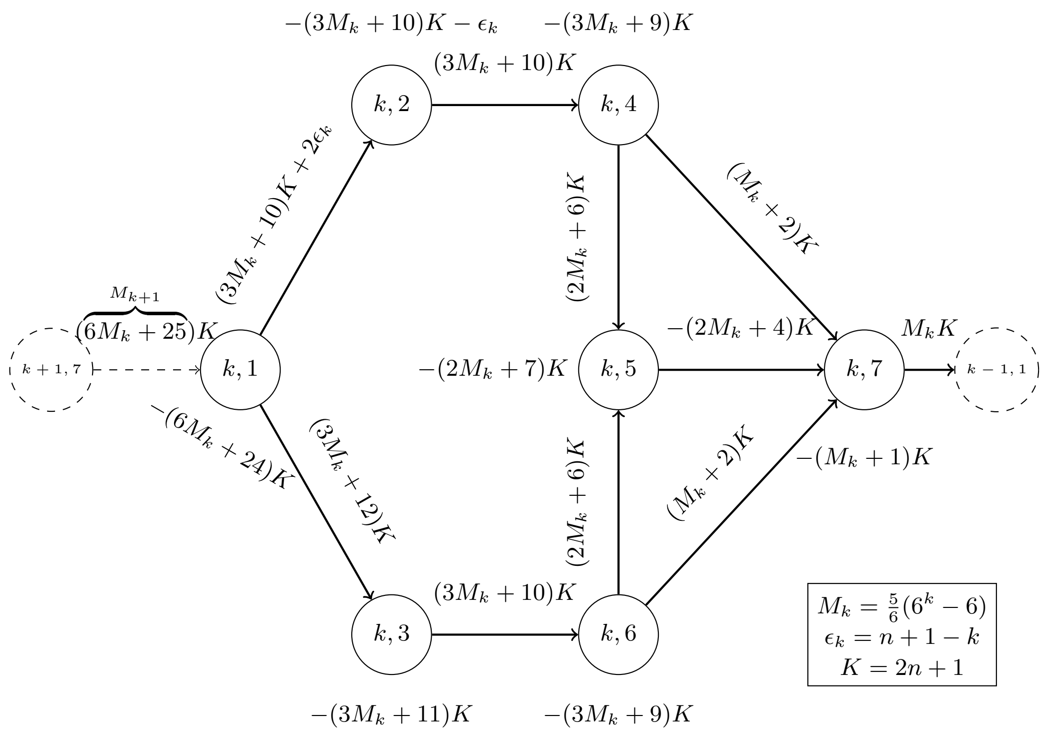


Figure 2 Haken-Luby gadget with $M_k = \frac{5}{6}(6^k - 6)$, $\epsilon_k = n+1-k$, and $K = 2n+1$. Constraints of the kth of n gadgets are shown: weights of unary constraints are next to their variables and weights of binary constraints are above the edges that specify their scope. Arcs are oriented according to Definition 2, showing that the instance is oriented. Dotted arcs and vertices illustrate the connection to the neighboring gadgets. For the boundaries: the unary of $(n,1)$ is $(6M_n + 24)K > 0$, $M_1 = 0$ and there is no binary constraint $c_{(1,7),(0,1)}$.

▶ **Proposition 17.** *Haken-Luby* VCSP *is an oriented* VCSP.

Proof. It suffices to check the constraints in Figure 2 against Definition 6. For example, check the constraint with scope $S = \{(k,5),(k,7)\}$ and weight $c_S = -(2M_k+4)K < 0$. First, look at $\hat{c}_{(k,5)}(x,S)$: among all partial assignments in $\{0,1\}^{\{(k,4),(k,6)\}}$, only the assignment that sets

$x_{(k,4)} = 1$ and $x_{(k,6)} = 1$ yields a non-negative $\hat{c}_{(k,5)}(11, S) = 2(2M_k + 6)K - (2M_k + 7)K = (2M_k + 5)K > 0$; since $|c_S| = (2M_k + 4)K \not> (2M_k + 5)K = \hat{c}_{(k,5)}(11, S)$ it follows that $(k, 7) \not\to (k, 5)$. Second, look at $\hat{c}_{(k,7)}(x, S)$: the assignment that sets $x_{(k,4)} = 1, x_{(k,6)} = 0$ and $x_{(k-1,1)} = 0$ yields $\hat{c}_{(k,7)}(100, S) = ((1 - 1)M_k + (2 - 1))K = K > 0$; since $|c_S| = (2M_k + 4)K > K = \hat{c}_{(k,7)}(100, S)$, it follows that that $(k, 5) \to (k, 7)$. The other constraints can be checked similarly. ◀

▶ **Proposition 18.** *Haken-Luby* VCSP *has pathwidth* 3.[11]

Proof.

(⇒) Path decomposition for k-th gadget: $\{(k + 1, 7), (k, 1)\}$, $\{(k, 1), (k, 2), (k, 3), (k, 4)\}$, $\{(k, 3), (k, 4), (k, 5), (k, 6)\}$, $\{(k, 4), (k, 5), (k, 6), (k, 7)\}$, $\{(k, 7), (k - 1, 1)\}$.

(⇐) Contracting $(k, 3)$, $(k, 1)$, $(k, 2)$ and $(k, 4)$ yields a K_4 minor. ◀

Proposition 17 and 18 together with Haken and Luby's [20] proof that Haken-Luby VCSPs have an exponential steepest ascent, gives us:

▶ **Theorem 19** (Haken and Luby [20])**.** *There are oriented* VCSP*s on* $7n$ *bits with constraint graphs of pathwidth* 3 *such that* SteepestAscent *follows an ascent of length* $\geq 2^n$*.*

Unaware of much older Haken and Luby [20], Cohen et al. [7] claimed to show pathwidth 7 as best lower bound for a VCSP with exponential steepest ascents, which was later improved to pathwidth 4 [33]. Since Proposition 18 shows that the Haken-Luby VCSP has pathwidth 3, this means it was the lowest pathwidth construction all along. That Haken-Luby is oriented gives the bonus that the resulting fitness landscapes are semismooth – something that was not shown for the other constructions [7, 33]. Very recently, Kaznatcheev and Vazquez Alferez [34] and van Marle [51] produced a construction similar to Haken and Luby [20] that reduced the bound to pathwidth 2, and showed that their construction is an oriented VCSP. This is the lowest possible pathwidth with exponential steepest ascent because all ascents are quadratic for tree-structured VCSPs [32]. So some of the simplest oriented VCSPs are already hard for greedy local search.

Now, we prove that conditionally-smooth landscape are intractable for RandomFacet [17], by showing that conditionally-smooth landscapes can express Matoušek AUSOs [17, 40].

▶ **Definition 20** (Gärtner [17], Matoušek [40])**.** *Given any* n *parity functions* $P_i : \{0, 1\}^{R_i} \to \{0, 1\}$ *with scopes such that* $i \in R_i \subseteq [i]$*, a landscape on* $\{0, 1\}^n$ *with fitness function* $f(x) = -\sum_{i=1}^{n} 2^{n-i} P_i(x[R_i])$ *is a* Matoušek AUSO.

AUSOs are often defined just in terms of their outmap (ϕ^+), without specific fitness values, so we had to make a specific choice in Definition 20. Our choice of fitness function, however, is the simplest one in terms of overall arity that can can implement the out-map of the Matoušek AUSOs. Specifically, Batman [3] showed that any VCSP implementing the same out-map as Matoušek AUSOs must have non-zero constraints with scopes R_i. Overall, the fitness function of a Matousek AUSO is very well behaved, in particular:

▶ **Proposition 21.** *If* f *is a Matoušek AUSO on* n *bits then for all* $k \in [n]$ *and* $y \in \{0, 1\}^{[k-1]}$ *there is a preferred assignment* $b \in \{0, 1\}$ *such that* $\forall z \in \{0, 1\}^{[n]-[k]}$ $f(ybz) > f(y\bar{b}z)$*.*

Proof. This follows from rewriting the big sum for f in 3 terms:

$$f(xby) = -\left(2^{n-(k-1)} \sum_{i=1}^{k-1} 2^{(k-1)-i} P_i(y[R_i]))\right) - 2^{n-k} P_k(yb[R_k]) - \left(2^{n-k} \sum_{i=1}^{n-k} 2^{-i} P_i(ybz[R_i])\right) \quad (7)$$

[11] For standard definitions of pathwidth/treewidth, see [9].

and noting that the first summand is independent of b and z, the last summand can sum to at most $-(2^{n-k}-1)$, and that the middle summand is independent of z and saves us 2^{n-k} in the sum if we set $b = \overline{\text{Parity}(y[R_k - \{k\}])}$. ◀

From this, it follows that:

▶ **Corollary 22.** *Matoušek AUSOs are both conditionally-smooth and semismooth.*

Given the similarity of Proposition 21 and Proposition A.7, if we generalized the definition of oriented binary VCSPs to arbitrary arity then Matoušek AUSOs would be oriented. The real power of Corollary 22 is that we can combine it with the super-polynomial lower-bound on the runtime of RandomFacet (Theorem 4.2 from Gärtner [17]) to give:

▶ **Theorem 23** (Gärtner [17])**.** *There exist families of fitness landscapes that are both conditionally-smooth and semismooth such that* RandomFacet *follows an ascent with expected length* $exp(\Theta(\sqrt{n}))$.

Given that RandomFacet is the best known algorithm for semismooth fitness landscapes, it is surprising to see it performing at its worst-case on conditionally-smooth landscapes.

8 Conclusion and Future Work

Overall, we showed that conditionally-smooth fitness landscapes – a polynomial-time testable structural property of VCSPs that generalizes the natural notion of oriented VCSPs – are an expressive subclass of Boolean VCSPs for which many (but not all) popular local search algorithm are both effective and efficient. Future work could further generalize conditionally-smooth landscapes from Boolean to higher-valence domains. This might allow us to engage with directed VCSPs where each strongly connected component is of small size $D \in O(\log n)$ by modeling the whole connected component as one domain with 2^D values. Would this class still be efficient-for-many local search algorithms? Is local search fixed-parameter tractable when parameterized by the size of the VCSP's largest strongly connected component?

We hope that our results on conditionally-smooth fitness landscapes contribute to a fuller understanding of when local search is both effective and efficient. This is interesting for theory as it grows our understanding of parameterized complexity, especially for PLS. This theory can guide us to what properties practical fitness landscapes might have in the real-world cases where local search seems to work well. Finally, we are excited about the use of these results for theory-building in the natural sciences. When local search is used by nature, we often do not have perfect understanding of which local search algorithm nature follows. But we might have strong beliefs about nature's algorithm being efficient. In this case, a good understanding of what landscapes are efficient-for-many local search algorithms can help us to reduce the set of fitness landscapes that we consider theoretically possible in nature. This can be especially useful in fields like evolutionary biology [30, 31] and economics [45] where we have only limited empirical measurements of nature's fitness landscapes.

References

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

2 Yoshikazu Aoshima, David Avis, Theresa Deering, Yoshitake Matsumoto, and Sonoko Moriyama. On the existence of Hamiltonian paths for history based pivot rules on acyclic unique sink orientations of hypercubes. *Discrete Applied Mathematics*, 160(15):2104–2115, 2012. doi:10.1016/J.DAM.2012.05.023.

3 Taylan Batman. Arity of polynomials for equivalence classes of pseudo-Boolean functions. Bachelor's thesis, Utrecht University, 2025.

4 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`.

5 Michaela Borzechowski. The complexity class polynomial local search (PLS) and PLS-complete problems. Bachelor's thesis, Freie Universitat Berlin, 2016.

6 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`.

7 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`.

8 William H. Cunningham. Theoretical properties of the network simplex method. *Mathematics of Operations Research*, 4(2):196–208, 1979. `doi:10.1287/MOOR.4.2.196`.

9 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`.

10 Duc-Cuong Dang, Anton Eremeev, and Per Kristian Lehre. Non-elitist evolutionary algorithms excel in fitness landscapes with sparse deceptive regions and dense valleys. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1133–1141, 2021. `doi:10.1145/3449639.3459398`.

11 Duc-Cuong Dang and Per Kristian Lehre. The slo hierarchy of pseudo-boolean functions and runtime of evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1551–1559, 2024.

12 Benjamin Doerr and Frank Neumann, editors. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer, 2020. Also available at `http://www.lix.polytechnique.fr/Labo/Benjamin.Doerr/doerr_neumann_book.html`.

13 MR Emamy-K. The worst case behavior of a greedy algorithm for a class of pseudo-boolean functions. *Discrete Applied Mathematics*, 23(3):285–287, 1989. `doi:10.1016/0166-218X(89)90018-8`.

14 Yahya Fathi and Craig Tovey. Affirmative action algorithms. *Mathematical Programming*, 34(3):292–301, 1986. `doi:10.1007/BF01582232`.

15 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique end of potential line. *Journal of Computer and System Sciences*, 114:1–35, 2020. `doi:10.1016/J.JCSS.2020.05.007`.

16 Yuan Gao, Bernd Gärtner, and Jourdain Lamperski. A new combinatorial property of geometric unique sink orientations. *arXiv preprint arXiv:2008.08992*, 2022.

17 Bernd Gärtner. The random-facet simplex algorithm on combinatorial cubes. *Random Structures & Algorithms*, 20(3):353–381, 2002. `doi:10.1002/RSA.10034`.

18 Teofilo F. Gonzalez, editor. *Handbook of Approximation Algorithms and Metaheuristics: Methologies and Traditional Applications*, volume 1. Chapman and Hall/CRC, New York, 2 edition, 2018.

19 Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Further collapses in tfnp. *arXiv preprint arXiv:2202.07761*, 2022. `arXiv:2202.07761`.

20 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`.

21 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`.

22 Thomas Dueholm Hansen and Uri Zwick. Random-edge is slower than random-facet on abstract cubes. *Leibniz International Proceedings in Informatics*, 55:51–1, 2016. doi:10.4230/LIPIcs.ICALP.2016.51.

23 Jun He and Yuren Zhou. Drift analysis with fitness levels for elitist evolutionary algorithms. *Evolutionary Computation*, 33(1):1–25, 2025. doi:10.1162/EVCO_A_00349.

24 Ronald A Howard. *Dynamic programming and Markov Processes*. John Wiley, 1960.

25 Thomas Jansen and Ingo Wegener. Evolutionary algorithms-how to cope with plateaus of constant fitness and when to reject strings of the same fitness. *IEEE Transactions on Evolutionary Computation*, 5(6):589–599, 2002. doi:10.1109/4235.974841.

26 Johnny Jermias and Lindawati Gani. Integrating business strategy, organizational configurations and management accounting systems with business unit effectiveness: a fitness landscape approach. *Management Accounting Research*, 15(2):179–200, 2004.

27 Robert G Jeroslow. The simplex algorithm with the pivot rule of maximizing criterion improvement. *Discrete Mathematics*, 4(4):367–377, 1973. doi:10.1016/0012-365X(73)90171-4.

28 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.

29 Gil Kalai. Combinatorics with a geometric flavor. In *Visions in Mathematics: GAFA 2000 Special volume, Part II*, pages 742–791. Springer, 2000.

30 Artem Kaznatcheev. Computational complexity as an ultimate constraint on evolution. *Genetics*, 212(1):245–265, 2019.

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

32 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.

33 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.

34 Artem Kaznatcheev and Sofia Vazquez Alferez. Greed is slow on sparse graphs of oriented valued constraints. In *31st International Conference on Principles and Practice of Constraint Programming (CP 2025)*, pages 44:1–44:13, 2025. doi:10.4230/LIPIcs.CP.2025.18.

35 Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970. doi:10.1002/J.1538-7305.1970.TB01770.X.

36 Victor Klee and George J Minty. How good is the simplex algorithm. *Inequalities*, 3(3):159–175, 1972.

37 Mark W. Krentel. On finding and verifying locally optimal solutions. *SIAM Journal on Computing*, 19(4):742–749, 1990. doi:10.1137/0219052.

38 Daniel A Levinthal. Adaptation on rugged landscapes. *Management Science*, 43(7):934–950, 1997.

39 J. Matousek and T. Szabo. RANDOM EDGE can be exponential on abstract cubes. *Advances in Mathematics*, 204:262–277, 2006.

40 Jiří Matoušek. Lower bounds for a subexponential optimization algorithm. *Random Structures & Algorithms*, 5(4):591–607, 1994. doi:10.1002/RSA.3240050408.

41 Jiří Matoušek. The number of unique-sink orientations of the hypercube. *Combinatorica*, 26(1):91–99, 2006. doi:10.1007/S00493-006-0007-0.

42 Wil Michiels, Emile Aarts, and Jan Korst. *Theoretical aspects of local search*, volume 25. Springer, 2007. doi:10.1007/978-3-540-35854-1.

43 F.J. Poelwijk, D.J. Kiviet, D.M. Weinreich, and S.J. Tans. Empirical fitness landscapes reveal accessible evolutionary paths. *Nature*, 445:383–386, 2007.

44 F.J. Poelwijk, T.-N. Sorin, D.J. Kiviet, and S.J. Tans. Reciprocal sign epistasis is a necessary condition for multi-peaked fitness landscapes. *Journal of Theoretical Biology*, 272:141–144, 2011.

45 T Roughgarden. Computing equilibria: A computational complexity perspective. *Economic Theory*, 42:193–236, 2010.

46 Alejandro A. Schäffer and Mihalis Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991. `doi:10.1137/0220004`.

47 Ingo Schurr and Tibor Szabó. Jumping doesn't help in abstract cubes. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 225–235. Springer, 2005.

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

49 Thomas Stützle. *Local search algorithms for combinatorial problems: Analysis, Improvements, and New Applications*. PhD thesis, TU Darmstadt, 1998.

50 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`.

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

52 Ingo Wegener. Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In *Evolutionary Optimization*, pages 349–369. Springer, 2002.

53 D.M. Weinreich, R.A. Watson, and L. Chan. Sign epistasis and genetic constraint on evolutionary trajectories. *Evolution*, 59:1165–1174, 2005.

54 S. Wright. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *Proc. of the 6th International Congress on Genetics*, pages 355–366, 1932.

55 Norman Zadeh. What is the worst case behavior of the simplex algorithm? In *Polyhedral computation*, volume 48 of *CRM Proceedings and Lecture Notes*, pages 131–143. American Mathematical Society, Providence, RI, 2009.

A Appendices

All appendices are available in the full version of this article at `https://arxiv.org/abs/2410.02634v3`.

These include all the appendix results (along with their proofs) that are mentioned in the main text:

- Appendices A, B.2, B.3, C.1, C.2, C.3 and C.4;
- Corollaries B.3 and B.4;
- Definition A.8 (adapted from [16] – see also [17, 41]);
- Lemma C.1;
- Propositions A.1, A.4, A.5, A.7, B.1, B.2, C.2, C.6, C.8, and C.10; and
- Theorem A.3.