Representing fitness landscapes
by valued constraints
to understand the complexity of local search

Artem Kaznatcheev[1,2], David A. Cohen[3], and Peter G. Jeavons[1]

[1] Department of Computer Science, University of Oxford, UK
[2] Department of Translational Hematology & Oncology Research, Cleveland Clinic, Cleveland, OH, USA
[3] Department of Computer Science, Royal Holloway, University of London, UK

Abstract. Local search is widely used to solve combinatorial optimisation problems and to model biological evolution, but the performance of local search algorithms on different kinds of fitness landscapes is poorly understood. Here we introduce a natural approach to modelling fitness landscapes using valued constraints. This allows us to investigate minimal representations (normal forms) and to consider the effects of the structure of the constraint graph on the tractability of local search. First, we show that for fitness landscapes representable by binary Boolean valued constraints there is a minimal necessary constraint graph that can be easily computed. Second, we consider landscapes as equivalent if they allow the same (improving) local search moves; we show that a minimal normal form still exists, but is NP-hard to compute. Next we consider the complexity of local search on fitness landscapes modelled by valued constraints with restricted forms of constraint graph. In the binary Boolean case, we prove that a tree-structured constraint graph gives a tight quadratic bound on the number of improving moves made by any local search; hence, any landscape that can be represented by such a model will be tractable for local search. We build two families of examples to show that both the conditions in our tractability result are essential. With domain size three, even just a path of binary constraints can model a landscape with an exponentially long sequence of improving moves. With a treewidth two constraint graph, even with a maximum degree of three, binary Boolean constraints can model a landscape with an exponentially long sequence of improving moves.

1 Introduction

Local search techniques are widely used to solve combinatorial optimisation problems, and have been intensively studied since the 1980's [1,13,14,16,20]. They have also played a central role in the theory of biological evolution, ever since Sewall Wright [23] introduced the idea of viewing the evolution of populations of organisms as a local search process over a space of possible genotypes with associated fitness values that became known as a "fitness landscape".

The term *fitness landscape* is now used to designate any structure (A, f, N) consisting of a set of points A, a function f defined on those points, and a neighbourhood function N on those points, that indicates which pairs of points are sufficiently close to be considered neighbours. A point x is said to be *locally optimal* if all neighbours are non-improving (i.e. $\forall y \in N(x)\ f(x) \geq f(y)$) and *globally optimal* if all points are non-improving. The *local search problem* for a fitness landscape is to find such a local optimum. We say the problem is solved by a *local search algorithm* if the only moves allowed in the procedure are from a point x to a point x' with $x' \in N(x)$ and $f(x') > f(x)$.

Many approaches have been developed to try to distinguish fitness landscapes where a local or global optimal point can be found efficiently by local search from those where such optimal points cannot be found efficiently. In the 1980's and 90's these attempts focused on statistical measures such as correlation between function values at various distances and various notions of *ruggedness* [14]. But, by the late 90's there were several studies highlighting the existence of fitness landscapes that were not rugged and yet were hard to optimise. Several new approaches have been developed recently, but the performance of local search algorithms on many kinds of fitness landscapes is still poorly understood [14,16,20].

An approach that has not yet been explored in any detail, is to extend the modelling and analysis techniques recently developed for *valued constraint satisfaction problems* [4,12,21,22,2,8] to analyse the computational difficulty of local search. In this paper we begin the development of a novel approach to understanding fitness landscapes based on representing those landscapes as valued constraint satisfaction problems (VCSPs), and studying the properties of the associated constraint graphs. In Section 3, we show how to efficiently construct a minimal representation (normal form) of fitness landscapes as VCSPs. In Section 4, we equate all fitness landscapes that have the same improving local search moves and show that a minimal form still exists for each equivlance class but is, in general, NP-hard to compute. Building on these results, the VCSP representation allows us to classify fitness landscapes in new ways, and hence to distinguish new classes of fitness landscapes with specific properties.

Since the WEIGHTED 2-SAT problem can be cast as a VCSP, finding a locally optimal solution for an arbitrary VCSP is a complete problem for the class of problems known as polynomial local search (PLS) [10,19,3]. This means that for a general VCSP it is expected to be computationally intractable even to find a local optimum by any method (not just by a local search algorithm). If we restrict to local search algorithms, then there exist standard constructions that produce families of fitness landscapes where every sequence of improving moves to a local optimum from some starting points is exponentially long [19]. On such landscapes, from such points, any local-search algorithm will require an exponentially long sequence of improving moves to reach a local optimum.

A key goal, therefore, is to identify classes of VCSPs where finding a locally optimal solution by local search is *tractable* (i.e., solvable in polynomial-time). In Section 5, we prove that fitness landscapes that can be represented by binary Boolean VCSPs with *tree-structured constraint graphs* can have only quadrat-

ically long sequences of improving moves – hence they are tractable for any local search algorithm. This is especially useful for investigating properties of biological evolution, as we discuss in the conclusion.

2 Background, notation, and general definitions

We will model the points, A, in our fitness landscapes as assignments to a collection of n variables, indexed by the set $[n] = 1, 2, \ldots, n$, with domains $D_1, \ldots, D_n$. Hence each point corresponds to a vector $x \in D_1 \times \cdots \times D_n$. We will generally focus on uniform domains (i.e., cases where $D = D_1 = \cdots = D_n$), where this simplifies to $x \in D^n$. In particular, we will often be interested in Boolean domains, where $x \in \{0, 1\}^n$, so each point can be seen as a bit-vector.

The restriction of a variable assignment to some subset of variables, with indices in a set $S \subseteq [n]$, will be denoted $x[S]$, so $x[S] \in \prod_{j \in S} D_j$. To reference the assignment to the variable at position i, we will usually write x_i unless it is ambiguous, in which case we'll use the more general notation $x[i]$. If we want to modify x by changing a single variable, say the variable at position i, to some element $b \in D_i$, then we'll write $x[i \mapsto b]$.

Given a set of points, A, a **fitness function** on A is defined to be an integer-valued function defined on A, that is, a function $f : A \to \mathbb{Z}$. Because we are modelling fitness, rather than cost, we *maximise* our objective functions in this paper. All results can be carried over directly to the minimisation context.

To complete the definition of a fitness landscape, we will define a **neighbourhood function** on the set of points A to be a function $N : A \to 2^A$. For simplicity, we will assume this function is symmetric in the sense that if $y \in N(x)$, then $x \in N(y)$, and we will call such a pair x and y **adjacent** points. Throughout the paper, we will focus on the case where the set of points A is the set of assignments $D_1 \times \cdots \times D_n$ and N is the **1-flip neighbourhood** defined by $y \in N(x)$ if and only if there is a variable position i such that $x_i \neq y_i$ and this is the only difference (i.e., $\forall j \neq i \quad x_j = y_j$). In the case of the Boolean domain, the graph of the function N, where the edges are the pairs of adjacent points, is the n-dimensional hypercube.

Definition 1 ([7,6]). *Given any fitness landscape (A, f, N), the corresponding* **fitness graph** *G has vertex set $V(G) = A$ and directed edge set $E(G) = \{xy \mid y \in N(x) \text{ and } f(y) > f(x)\}$.*

Note that the edges of the fitness graph consist of all pairs of adjacent points which have distinct values of the fitness function, and are oriented from the lower value of the fitness function to the higher value; such directed edges represent the possible moves that can be made by a local search algorithm.

A **(valued) constraint** with scope $S \subseteq [n]$ is a function $C_S : \prod_{j \in S} D_j \to \mathbb{Z}$. The **arity of a constraint** C_S is the size $|S|$ of its scope. For unary and binary constraints we will omit the set notation and just write C_i for $C_{\{i\}}$ or C_{ij} for $C_{\{i,j\}}$. We will represent the values taken by a unary constraint C_i for each domain element by an integer vector of length $|D_i|$, and represent the

values taken by a binary constraint C_{ij} for each pair of domain elements by an integer matrix, where x_i selects the row and x_j selects the column. A zero-valued constraint (of any arity) will be denoted by 0.

Definition 2. *An instance of the* valued constraint satisfaction problem (VCSP) *is a set of constraints* $\mathcal{C} = \{C_{S_1}, \ldots, C_{S_m}\}$. *We say that a VCSP-instance* $\mathcal{C}$ **implements** *a fitness function* f *if* $f(x) = \sum_{k=1}^{m} C_{S_k}(x[S_k])$.

The arity of a VCSP-instance is the maximum arity over its constraints; if this maximum arity is 2, then we will call it a *binary* VCSP-instance. The instance-size of a VCSP-instance is the number of bits needed to specify n, m and each constraint.

Given any VCSP-instance $\mathcal{C}$, we can take A as the set of all possible assignments, f as the fitness function implemented by $\mathcal{C}$, and N as the 1-flip neighbourhood, to obtain an associated fitness landscape, (A, f, N), and hence an associated fitness graph, $G_{\mathcal{C}}$, by Definition 1. Note that the vertex set of $G_{\mathcal{C}}$ is the set of possible assignments, A, and hence is exponential in the size of the instance, $\mathcal{C}$, in general. Each *binary* VCSP-instance also has an associated *constraint graph*, defined as follows, whose vertex set is polynomial in the size of the instance:

Definition 3. *Given any binary VCSP-instance* $\mathcal{C}$, *the corresponding* **constraint graph** *has vertices* $V(\mathcal{C}) = [n]$, *edges* $E(\mathcal{C}) = \{ij \mid C_{ij} \in \mathcal{C}, C_{ij} \neq 0\}$, *and constraint-neighbourhood function* $N_{\mathcal{C}}(i) = \{j \mid ij \in E(\mathcal{C})\}$.

3 Magnitude-equivalence

It is clear from Definition 2 that different VCSP-instances can implement the same fitness function. Consider, the following two small VCSP-instances:

$$\left(x_1\right) - \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} - \left(x_2\right) \qquad \text{vs.} \qquad C_{\emptyset} = 1 \qquad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \left(x_1\right) \qquad\qquad \left(x_2\right) \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Although these two instances have different constraint graphs, the fitness function they implement is $[f(00), f(01), f(10), f(11)] = [1, 2, 2, 3]$ in both cases. We capture this equivalence with the following definition:

Definition 4. *If two VCSP-instances* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *implement the same fitness function* f, *then we will say they are* **magnitude-equivalent***.*

We will show in this section that for binary Boolean VCSP-instances each equivalence class of magnitude-equivalent VCSP-instances has a *normal form*: a unique, minimal, and easy to compute representative member with special properties.

Definition 5. *A binary Boolean VCSP-instance* $\mathcal{C}$ *is* **simple** *if every unary constraint has* $C_i = \begin{pmatrix} 0 \\ c_i \end{pmatrix}$ *and every binary constraint has* $C_{ij} = \begin{pmatrix} 0 & 0 \\ 0 & c_{ij} \end{pmatrix}$.

In drawings of constraint graphs of simple VCSP-instances we will often denote the unary constraint $\begin{pmatrix} 0 \\ c_i \end{pmatrix}$ by c_i, and the binary constraint $\begin{pmatrix} 0 & 0 \\ 0 & c_{ij} \end{pmatrix}$ by c_{ij}.

We now give a direct proof of the following simplification result which is analogous to similar results using constraint propagation in standard VCSP [5].

Theorem 1. *Any binary Boolean VCSP-instance $\mathcal{C}'$ can be transformed into a unique simple VCSP-instance $\mathcal{C}$ that is magnitude-equivalent to $\mathcal{C}'$. Moreover, $\mathcal{C}$ can be constructed from $\mathcal{C}'$ in linear time.*

Proof. First two key observations: (1) Any unary Boolean constraint C_i' can be rewritten as a linear function: $c_i'(x) = (1-x_i)C_i'(0) + x_i C_i'(1)$; and (2) any binary Boolean constraint C_{ij}' can be rewritten as a multilinear polynomial of degree 2: $c_{ij}'(x) = (1-x_i)(1-x_j)C_{ij}'(0,0) + (1-x_i)x_j C_{ij}'(0,1) + x_i(1-x_j)C_{ij}'(1,0) + x_i x_j C_{ij}'(1,1)$. From this, we can simplify $\mathcal{C}'$ just by simplifying polynomials:

$$f(x) = C_\emptyset' + \sum_{i=1}^{n} C_i'(x_i) + \sum_{ij \in E(\mathcal{C}')} C_{ij}'(x_i, x_j) = C_\emptyset' + \sum_{i=1}^{n} c_i'(x) + \sum_{ij \in E(\mathcal{C}')} c_{ij}'(x) \quad (1)$$

$$= C_\emptyset + \sum_{i=1}^{n} x_i c_i + \sum_{1 \le i < j \le n} x_i x_j c_{ij} \quad (2)$$

where we note that the last part of Equation 1 is a sum of a constant, some linear functions, and some multilinear polynomials of degree 2, and is thus itself a multilinear polynomial of degree 2 (or less). Equation 2 then follows from Equation 1 by multiplying out into monomials and then grouping the coefficients of each similar monomial. This can be done in time linear in the number of constraints and the number of bits needed to encode their coefficients. We note that Equation 2 corresponds to a VCSP-Instance $\mathcal{C}$ comprising a null-term $C_\emptyset$, unary constraints $C_i = \begin{pmatrix} 0 \\ c_i \end{pmatrix}$, and binary constraints $C_{ij} = \begin{pmatrix} 0 & 0 \\ 0 & c_{ij} \end{pmatrix}$. □

The next result shows that a simple VCSP-instance has the minimal constraint graph of any binary instance that implements the same fitness function:

Theorem 2. *Let $\mathcal{C}$ be a simple binary Boolean VCSP-instance. If the binary Boolean VCSP-instance $\mathcal{C}'$ is magnitude-equivalent to $\mathcal{C}$, then $E(\mathcal{C}) \subseteq E(\mathcal{C}')$.*

Proof. Let $e_i \in \{0,1\}^n$ be a variable assignment that sets the ith variable to one, and all other variables to zero. Similarly, let $e_{ij} \in \{0,1\}^n$ be a variable assignment that sets the ith and jth variables to one, and all other variables to zero. Since $\mathcal{C}$ implements f, we have:

$$f(e_{ij}) - f(e_i) - f(e_j) + f(0^n) = c_{ij} \quad (3)$$

where we take $c_{ij} = 0$ if $ij \notin E(\mathcal{C})$. Similarly, if $\mathcal{C}'$ also implements f, we have:

$$f(e_{ij}) - f(e_i) - f(e_j) + f(0^n) = C_{ij}'(1,1) - C_{ij}'(1,0) - C_{ij}'(0,1) + C_{ij}'(0,0)$$

If $ij \in E(\mathcal{C})$ then $c_{ij} \neq 0$, so $C_{ij}'(1,1) - C_{ij}'(1,0) - C_{ij}'(0,1) + C_{ij}'(0,0) \neq 0$ and hence $ij \in E(\mathcal{C}')$. □

4 Sign-equivalence

In the previous section we considered the equivalence class of all VCSP-instances which implement precisely the same fitness function. However, when investigating the performance of local search algorithms, the exact values of the fitness function are not always relevant; it may be sufficient to consider only the fitness graph.

For example, consider a fitness function f, implemented by a VCSP-instance $\mathcal{C}$, where all fitness values are distinct, but there is at least one pair i, j of positions with no constraint C_{ij}. Now consider the new fitness function $f'(x) = 2f(x) + C_{ij}(x_i, x_j)$ where $C_{ij} = [0, 0; 0, 1]$. The fitness graph corresponding to f' is unchanged (since all fitness values given by $2f(x)$ differ by at least 2, every edge is still present in the fitness graph, and no orientations are changed by the new constraint), but we cannot eliminate this new C_{ij} constraint without changing the precise values of the fitness function. To capture this similarity between f and f', we introduce a more abstract equivalence relation:

Definition 6. *If two VCSP-instances $\mathcal{C}_1$ and $\mathcal{C}_2$ give rise to the same fitness graph, then we will say they are* **sign-equivalent***.*

As with magnitude-equivalence, we will show that for binary Boolean VCSP-instances it is possible to define a normal form or minimal representative member of each equivalence class of sign-equivalent VCSP-instances with a unique minimal constraint graph. Unfortunately, we will see that, unlike the situation for magnitude-equivalence, this minimum sign-equivalent constraint-graph is NP-hard to compute.

Definition 7. *In a Boolean fitness graph G with vertex set $\{0, 1\}^n$, we will say that i* **sign-depends** *on j if there exists an assignment $x \in \{0, 1\}^n$ such that:*

$$xx[i \mapsto \overline{x}_i] \in E(G) \quad but \quad x[j \mapsto \overline{x}_j]x[i \mapsto \overline{x}_i, j \mapsto \overline{x}_j] \notin E(G) \qquad (4)$$

Note that i **sign-depends** on j if and only if, for any fitness function f that corresponds to the fitness graph G, there exists $x \in \{0, 1\}^n$ such that:

$$\mathrm{sgn}(f(x[i \mapsto \overline{x}_i]) - f(x)) \neq \mathrm{sgn}(f(x[i \mapsto \overline{x}_i, j \mapsto \overline{x}_j]) - f(x[j \mapsto \overline{x}_j])). \qquad (5)$$

We will say that i and j **sign-interact** if i sign-depends on j or j sign-depends on i (or both). If i and j do not sign-interact then we will say that they are **sign-independent**.

Definition 8. *A simple binary Boolean VCSP-instance $\mathcal{C}$ with associated fitness graph $G_{\mathcal{C}}$ is called* **trim** *if for all $ij \in E(\mathcal{C})$, i and j sign-interact in $G_{\mathcal{C}}$.*

Our sign-equivalent analog of Theorem 1 guarantees a normal form:

Theorem 3. *Any simple binary Boolean VCSP-instance $\mathcal{C}'$ can be transformed into a trim VCSP-instance $\mathcal{C}$ that is sign-equivalent to $\mathcal{C}'$.*

To prove Theorem 3 we now establish two propositions: Proposition 1 connects the magnitude of constraints with their effect on fitness graphs, and Proposition 2 connects the magnitude of constraints to sign-interaction.

Proposition 1. *Given a simple binary Boolean VCSP-instance $\mathcal{C}$ implementing a fitness function f, if removing the constraint C_{ij} changes the corresponding fitness graph, then for at least one $k \in \{i, j\}$ there exists some $x \in \{0,1\}^n$ with $x_i = x_j = 1$ such that:*

$$c_{ij} \geq f(x) - f(x[k \mapsto 0]) > 0 \quad or \quad c_{ij} \leq f(x) - f(x[k \mapsto 0]) < 0 \qquad (6)$$

Proof. Without loss of generality (by swapping i and j in the variable numbering if necessary), we can suppose that $k = i$. Consider two cases:

Case 1 ($c_{ij} > 0$): If removing C_{ij} changes the fitness graph, then there exists some $x \in \{0,1\}^n$ with $x_i = x_j = 1$ such that:

$$f(x) > f(x[i \mapsto 0]) \quad \text{but} \quad f(x) - c_{ij} \leq f(x[i \mapsto 0]). \qquad (7)$$

We can re-arrange Equation 7 to get $c_{ij} \geq f(x) - f(x[i \mapsto 0]) > 0$

Case 2 ($c_{ij} < 0$): This is the same as case 1, except that the direction of the inequalities in Equation 7 are reversed. $\square$

Proposition 2. *Given a simple binary Boolean VCSP-instance $\mathcal{C}$ implementing a fitness function f, if there exists a constraint C_{ij} in $\mathcal{C}$, some assignment $x \in \{0,1\}^n$ with $x_i = x_j = 1$, and some $k \in \{i, j\}$ such that:*

$$c_{ij} \geq f(x) - f(x[k \mapsto 0]) > 0 \quad or \quad c_{ij} \leq f(x) - f(x[k \mapsto 0]) < 0 \qquad (8)$$

then i sign-depends on j in the associated fitness graph $G_{\mathcal{C}}$.

Proof. As in the proof of Proposition 1, we can suppose that $k = i$ (by swapping i and j in the variable numbering if necessary). Also, as in the proof of Proposition 1, the case for $c_{ij} < 0$ is symmetric (by flipping the direction of inequalities) to $c_{ij} > 0$. Thus, we will just consider the case where $k = i$ and $c_{ij} > 0$:

Given that Equation 8 tells us that $f(x) > f(x[i \mapsto 0])$ (i.e., that $x[i \mapsto 0]x \in E(G_{\mathcal{C}})$), to establish that i sign-depends on j per Definition 7, we need to show that $f(x[j \mapsto 0]) \leq f(x[i \mapsto 0, j \mapsto 0])$ (i.e., that $x[i \mapsto 0, j \mapsto 0]x[j \mapsto 0] \notin E(G_{\mathcal{C}})$). So, let us look at the difference of the latter:

$$f(x[j \mapsto 0]) - f(x[i \mapsto 0, j \mapsto 0]) = f(x) - f(x[i \mapsto 0]) - c_{ij} \leq 0 \qquad (9)$$

where the equality follows from Definition 2 ($\mathcal{C}$ implements f) and Definition 5 ($\mathcal{C}$ is simple), and the inequality follows from the first part of Equation 8. $\square$

*Proof (of **Theorem 3**).* Note that Equations 6 and 8 specify the same conditions, hence the negation of this condition can be used to glue together the contrapositives of Proposition 2 (if i and j are sign-independent then Equation 6 does not hold) and Proposition 1 (if Equation 8 does not hold then C'_{ij} can be removed from $\mathcal{C}'$ without changing the corresponding fitness graph). So we can convert $\mathcal{C}'$ to a trim VCSP-instance that is sign-equivalent to $\mathcal{C}'$ by simply removing all $C'_{ij} \in \mathcal{C}'$ where i and j are sign-independent in the associated fitness graph $G_{\mathcal{C}'}$. $\square$

The next result is the sign-equivalence analog of Theorem 2. It shows that a trim VCSP-instance has the minimal constraint graph of any binary instance with the same associated fitness graph.

Theorem 4. *Let $\mathcal{C}$ be a trim binary Boolean VCSP-instance. If the binary Boolean VCSP-instance $\mathcal{C}'$ is sign-equivalent to $\mathcal{C}$, then $E(\mathcal{C}) \subseteq E(\mathcal{C}')$.*

To prove Theorem 4, we just need to show that constraints between sign-interacting positions cannot be removed while preserving sign-equivalence. That is, we just need the following proposition:

Proposition 3. *Let $\mathcal{C}$ be a binary Boolean VCSP-instance with associated fitness graph $G_{\mathcal{C}}$. If i, j sign-interact in $G_{\mathcal{C}}$, then the constraint C_{ij} in $\mathcal{C}$ is non-zero.*

Proof. Without loss of generality, assume that we have an edge in $G_{\mathcal{C}}$ from $x[i \mapsto \overline{x_i}]$ to x. Thus, the fitness function f implemented by $\mathcal{C}$ must satisfy the following two inequalities:

$$f(x) > f(x[i \mapsto \overline{x_i}]) \quad \text{and} \quad f(x[j \mapsto \overline{x_j}]) \leq f(x[i \mapsto \overline{x_i}, j \mapsto \overline{x_j}]) \qquad (10)$$

Define $g_i(x_i) = C_i(x_i) + \sum_{k \neq j} C_{ik}(x_i, x_k)$ and similarly for g_j. Also let $K_{ij}(x)$ be the part of f independent of x_i, x_j: i.e., $f(x) = K_{ij}(x) + g_i(x_i) + g_j(x_j) + C_{ij}(x_i, x_j)$. Rewriting (and simplifying) the two parts of Equation 10, we get:

$$g_i(x_i) + C_{ij}(x_i, x_j) > g_i(\overline{x_i}) + C_{ij}(\overline{x_i}, x_j) \qquad (11)$$
$$g_i(x_i) + C_{ij}(x_i, \overline{x_j}) \leq g_i(\overline{x_i}) + C_{ij}(\overline{x_i}, \overline{x_j}) \qquad (12)$$

These equations can be rotated to sandwich the g_i terms:

$$C_{ij}(x_i, x_j) - C_{ij}(\overline{x_i}, x_j) > g_i(\overline{x_i}) - g_i(x_i) \geq C_{ij}(x_i, \overline{x_j}) - C_{ij}(\overline{x_i}, \overline{x_j}) \qquad (13)$$

which simplifies to $C_{ij}(x_i, x_j) - C_{ij}(\overline{x_i}, x_j) > C_{ij}(x_i, \overline{x_j}) - C_{ij}(\overline{x_i}, \overline{x_j})$ and – due to the strict inequality – establishes that C_{ij} is non-zero. $\square$

However, unlike with magnitude-equivalence, it is NP-hard to determine a minimal sign-equivalent VCSP-instance, as the next result shows:
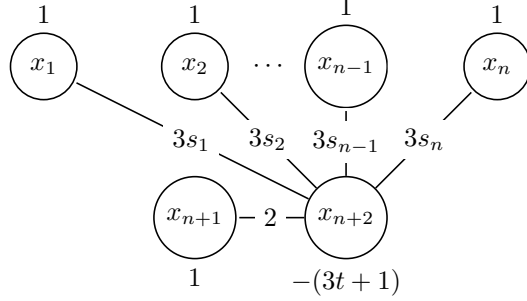
Theorem 5. *The problem of deciding whether i and j sign-interact in a given simple binary Boolean VCSP-instance is NP-complete.*

Proof. To see that this problem is in NP, note that we can provide a variable assignment x as a certificate and check that under that variable assignment either i sign-depends on j or j sign-depends on i (or both).

We will establish NP-hardness by reduction from the SUBSETSUM problem, which is known to be NP-complete [9]: A set of integers $\{s_1, \ldots, s_n\}$ and a target t is a yes-instance of the SUBSETSUM problem if there exists some subset $S \subseteq [n]$ such that $\sum_{i \in S} s_i = t$.

Now consider a simple binary Boolean VCSP-instance $\mathcal{C}$ on $n + 2$ variables, that implements fitness function f and has associated fitness graph $G_{\mathcal{C}}$, whose constraint graph has the shape of a star, with central variable position $n + 2$:

$$x_1 \quad x_2 \quad \cdots \quad x_{n-1} \quad x_n$$

(figure: nodes x_1 (label 1), x_2 (label 1), $\cdots$, x_{n-1} (label 1), x_n (label 1), with edges labeled $3s_1$, $3s_2$, $3s_{n-1}$, $3s_n$ to node x_{n+2}; node x_{n+1} (label 1) connected to x_{n+2} by edge labeled -2; x_{n+2} labeled $-(3t+1)$.)

Claim: $\langle \{s_1, \ldots, s_n\}, t \rangle$ is a yes-instance of SUBSETSUM if and only if $n+1$ and $n+2$ sign-interact.

We clearly have that for all $x \in \{0,1\}^{n+2}$, $f(x[n+1 \mapsto 1]) > f(x[n+1 \mapsto 0])$, so $n+1$ does not sign-depend on $n+2$. Thus our claim becomes equivalent to verifying the conditions under which $n+2$ sign-depends on $n+1$. Let's look at the two directions of the if and only if in the claim:

Case 1 ($\Rightarrow$): If $\langle \{s_1, \ldots, s_n\}, t \rangle \in$ SUBSETSUM, then there is a subset $S \subseteq [n]$ such that $\sum_{i \in S} s_i = t$. Let $e_S \in \{0,1\}^n$ be the variable assignment such that for any $i \in S$, $e_S[i] = 1$ and for any $j \notin S$, $e_S[j] = 0$. We have that:

$$f(e_S 01) = |S| - 1 \qquad\qquad f(e_S 11) = |S| + 2$$
$$f(e_S 00) = |S| \qquad\qquad\qquad f(e_S 10) = |S| + 1$$

By Equation 5, these imply that $n+2$ sign-depends on $n+1$.

Case 2 ($\Leftarrow$): If $\langle \{s_1, \ldots, s_n\}, t \rangle \notin$ SUBSETSUM, then for any $S \subseteq [n]$ we either have $\sum_{i \in S} s_i \leq t-1$ or $\sum_{i \in S} s_i \geq t+1$. Thus, given an arbitrary assignment $e_S \in \{0,1\}$ we have two subcases:

If $\sum_{i \in S} s_i \leq t-1$ then: Or, if $\sum_{i \in S} s_i \geq t-1$ then:

$$f(e_S 01) - f(e_S 00) \leq -4 \qquad\qquad f(e_S 01) - f(e_S 00) \geq 2$$
$$f(e_S 11) - f(e_S 10) \leq -2 \qquad\qquad f(e_S 11) - f(e_S 10) \geq 4$$

In either subcase, $\operatorname{sgn}(f(e_S 01) - f(e_S 00)) = \operatorname{sgn}(f(e_S 11) - f(e_S 10))$, so by Equation 5, $n+2$ does not sign-depend on $n+1$. $\qquad\square$

5   Tree-structured Boolean VCSP-instances

In this section, we will prove the following:

Theorem 6. *For a binary Boolean VCSP instance $\mathcal{C}$ on n variables, if the constraint-graph of $\mathcal{C}$ is a tree, then any directed path in the associated fitness graph $G_\mathcal{C}$ has length at most $\binom{n}{2} + n$.*

Note that this result bounds the length of *any* directed path in $G_\mathcal{C}$, not just the path taken by a particular local-search algorithm. Thus, on such landscapes even choosing the worst possible sequence of improving moves results in a local optimum being found in polynomial time.

We will show in Section 6 that the conditions of being Boolean and tree-structured are essential to obtain a polynomial bound on the length of all paths. To see that the bound in Theorem 6 is the best possible for binary Boolean tree-structured VCSP-instances, consider the example below:

Example 1. (**Path of length** $\binom{n}{2} + n$) Consider the following binary Boolean VCSP-instance $\mathcal{C}$:

$$
\boxed{x_1} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \boxed{x_2} \cdot \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \cdot \boxed{x_3} \cdot \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} - \cdots - \begin{pmatrix} n-1 & 0 \\ 0 & n-1 \end{pmatrix} - \boxed{x_n} \begin{pmatrix} 0 \\ n \end{pmatrix}
$$

To obtain a path of length $\binom{n}{2}+n$ in the corresponding fitness graph $G_\mathcal{C}$, consider an initial variable assignment of $x = (10)^{\frac{n}{2}}$ if n is even and $x = 0(10)^{\frac{n-1}{2}}$ if n is odd, and always select the leftmost variable that is able to flip. This will increase the fitness by 1 at each step, starting from 0 to $\binom{n}{2} + n$.

For example, when $n = 4$, this gives the following sequence of 11 assignments, each of which increases the value of the fitness function by 1:

$$0101 \to 1101 \to 1001 \to 0001 \to 0011 \to 0111 \to 1111 \to 1110 \to 1100 \to 1000 \to 0000 \quad (14)$$

For the proof of Theorem 6, we introduce some further definitions.

Definition 9. *Given any directed path $p = x^1 \dots x^t \dots x^T$ in a fitness graph G, define the flip function as $m(t) = (i \mapsto b)$ where $x^{t+1} \oplus x^t = e_i$ and $b = x_i^{t+1}$ (i.e., the i-th variable is flipped at time t to value b).*

For illustration, consider the sequence of moves listed in Equation 14 of Example 1. It corresponds to the following flip function:

$\mathbf{t}$	1	2	3	4	5	6	7	8	9	10
$\mathbf{m(t)}$	$1 \mapsto 1$	$2 \mapsto 0$	$1 \mapsto 0$	$3 \mapsto 1$	$2 \mapsto 1$	$1 \mapsto 1$	$4 \mapsto 0$	$3 \mapsto 0$	$2 \mapsto 0$	$1 \mapsto 0$

To obtain the bound on the length of paths given in Theorem 6, we will identify a structure in the flip function, to bound the maximum possible value for T.

Definition 10. *We say that a flip $m(t') = (j \mapsto c)$ **supports** a flip $m(t) = (i \mapsto b)$ if $t' < t$ and $C_{ij}(b,c) - C_{ij}(\bar{b},c) > C_{ij}(b,\bar{c}) - C_{ij}(\bar{b},\bar{c})$; if $x_j^t = c$, then the support is said to be **strong**.*

It is useful to note that the inequality on C_{ij} is symmetric in the sense that:

$$
\begin{aligned}
& C_{ij}(b,c) - C_{ij}(\bar{b},c) > C_{ij}(b,\bar{c}) - C_{ij}(\bar{b},\bar{c}) \\
\Leftrightarrow\ & C_{ij}(b,c) - C_{ij}(b,\bar{c}) > C_{ij}(\bar{b},c) - C_{ij}(\bar{b},\bar{c}) \quad (15) \\
\Leftrightarrow\ & C_{ji}(c,b) - C_{ji}(\bar{c},b) > C_{ji}(c,\bar{b}) - C_{ji}(\bar{c},\bar{b})
\end{aligned}
$$

Definition 11. *Given a binary Boolean VCSP-instance $\mathcal{C}$ implementing fitness function f, the **fitness contribution** of the variable at position i in assignment x, restricted to $S \subseteq [n]$ is defined to be:*

$$f_i^S(b|x) = \begin{cases} C_i(b) & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases} + \sum_{j \in N_{\mathcal{C}}(i) \cap S} C_{ij}(b, x_j) \tag{16}$$

if $S = [n]$ then we just write f_i rather than $f_i^{[n]}$.

Note that for any path p in G, if $m(t) = (i \mapsto b)$ then $f_i(b|x^t) > f_i(\bar{b}|x^t)$.

We now introduce an *encouragement* relation between a flip and its most recent strong supporting flip, if there is one:

Definition 12. *We say that a flip $m(t) = (i \mapsto b)$ is **encouraged** by its most recent strong supporting flip $m(t') = (j \mapsto c)$, and write $(t', j \mapsto c) \leftarrow (t, i \mapsto b)$.*

*If there are no strong supporting flips, then we say that a flip $m(t) = (i \mapsto b)$ is **courageous**, and write $\perp \leftarrow (t, i \mapsto b)$.*

Note that if $(t', j \mapsto c) \leftarrow (t, i \mapsto b)$, then $t' < t$ and $i \in N_{\mathcal{C}}(j)$.

For illustration, consider the sequence of moves listed in Equation 14 of Example 1. It corresponds to the following encouragement graph:

$$\perp \leftarrow (1, 1 \mapsto 1) \qquad \perp \leftarrow (2, 2 \mapsto 0) \;\leftarrow (3, 1 \mapsto 0)$$
$$\perp \leftarrow (4, 3 \mapsto 1) \leftarrow (5, 2 \mapsto 1) \leftarrow (6, 1 \mapsto 1)$$
$$\perp \leftarrow (7, 4 \mapsto 0) \leftarrow (8, 3 \mapsto 0) \leftarrow (9, 2 \mapsto 0) \leftarrow (10, 1 \mapsto 0)$$

Proposition 4. *If $(t_1, j \mapsto c) \leftarrow (t_2, i \mapsto b)$ (or if $\perp \leftarrow (t_2, i \mapsto b)$, set $t_1 = 0$) then for all $t_1 < t' \leq t_2$ we have $f_i(b|x^{t'}) - f_i(\bar{b}|x^{t'}) \geq f_i(b|x^{t_2}) - f_i(\bar{b}|x^{t_2}) > 0$.*

Proof. Define the set of temporary supports S_w as the set of positions of flips after t_1 that supported $(t_2, i \mapsto b)$ but weren't strong (i.e., they were flipped back by the time we got to t_2: for supportive $(t'', k \mapsto a)$ with $t'' > t_1$ we have $k \in S_w \Rightarrow a \neq x^{t_2}[k]$).

Consider any flip $m(t') = (k \mapsto a)$ for $t' \in [t_1 + 1, t_2 - 1]$. Since it either didn't support $(t_2, i \mapsto b)$ (and so had $C_{ij}(b, a) - C_{ij}(b, \bar{a}) \leq C_{ij}(\bar{b}, a) - C_{ij}(\bar{b}, \bar{a})$ by Equation 15) or was a temporary support, we have that:

$$f_i^{[n]-S_w}(b|x^{t'+1}) - f_i^{[n]-S_w}(\bar{b}|x^{t'+1}) \leq f_i^{[n]-S_w}(b|x^{t'}) - f_i^{[n]-S_w}(\bar{b}|x^{t'}) \tag{17}$$

Thus $\delta_i(t') = f_i^{[n]-S_w}(b|x^{t'}) - f_i^{[n]-S_w}(\bar{b}|x^{t'})$ is monotonically non-increasing in t' over the time interval $[t_1 + 1, t_2]$. So:

$$f_i^{[n]-S_w}(b|x^{t'}) - f_i^{[n]-S_w}(\bar{b}|x^{t'}) \geq f_i^{[n]-S_w}(b|x^{t_2}) - f_i^{[n]-S_w}(\bar{b}|x^{t_2}) \tag{18}$$

Since every position $k \in S_w$ supported $(t_2, i \mapsto b)$ but is absent in x^{t_2}, we must have $f_i^{S_w}(b|x^{t_2}) - f_i^{S_w}(\bar{b}|x^{t_2}) \leq f_i^{S_w}(b|x^{t'}) - f_i^{S_w}(\bar{b}|x^{t'})$. Noting that $f_i = f_i^{[n]-S_w} + f_i^{S_w}$ then lets us combine this with Equation 18 (and the fact that $f_i(b|x^{t_2}) > f_i(\bar{b}|x^{t_2})$) to complete the proposition. $\qquad\square$

11

By Definition 12, each flip can only be encouraged by at most one other flip, so each node in the encouragement graph has out-degree at most one. Directed graphs where each vertex has at most one parent are forests, so the encouragement graph is a forest. This forest has a component for each courageous flip, and we will now show that there are at most n of these:

Proposition 5. *At each variable position i, only the first flip can be courageous.*

Proof. Consider a courageous flip $\perp \leftarrow (t, i \mapsto b)$, by Proposition 4, we know that for all $t' < t$: $f_i(b|x^{t'}) - f_i(\bar{b}|x^{t'}) \geq f_i(b|x^t) - f_i(\bar{b}|x^t) > 0$ Thus, there is no time $t' \leq t$ such that i could have flipped to $\bar{b}$: hence i was always at $\bar{b}$ for $t' \leq t$. So the courageous flip had to be the first flip at that position. $\qquad\square$

We will now prove that an encouragement tree cannot double-back on itself in position (Proposition 6), and that every branch is a branch in position (Proposition 7). When the constraint graph is itself a tree, this will imply that each tree in the encouragement forest is a sub-tree of the constraint graph.

Proposition 6. *If $(t_1, i \mapsto a) \leftarrow (t_2, j \mapsto b) \leftarrow (t_3, k \mapsto c)$ then $i \neq k$*

Proof. Since $(t_1, i \mapsto a)$ strongly supported $(t_2, j \mapsto b)$, we have $x_i^{t_2} = a$. If, for the sake of contradiction, we assume that $i = k$ then $a = c$ (because if we had $c = \bar{a}$ then the two encouragements would force a contradiction via clashing Equations 15) and by Proposition 4: $f_i(a|x^{t'}) - f_i(\bar{a}|x^{t'}) \geq f_i(a|x^{t_3}) - f_i(\bar{a}|x^{t_3}) \geq 0$ for all $t_2 < t' \leq t_3$. But this means that i cannot be flipped to $\bar{a}$ and thus $m(t_3) = (i, a)$ is not a legal flip. This is a contradiction and so $i \neq k$. $\quad\square$

Proposition 7. *For all i, j and $t_1 < t_2 \leq t_3$: if $(t_1, i \mapsto a) \leftarrow (t_2, j \mapsto b)$ and $(t_1, i \mapsto a) \leftarrow (t_3, j \mapsto c)$, then $t_2 = t_3$.*

Proof. From Proposition 4, we can see that for all $t' \in [t_1 + 1, t_3]$, $f_j(c|x^{t'}) - f_j(\bar{c}|x^{t'}) > 0$, so $b = c$ and j couldn't have flipped from c to $\bar{c}$ between t_2 and t_3. Thus, for $(t_2, j \mapsto c)$ to be a legal flip, we must have $t_2 = t_3$. $\qquad\square$

Now, if we look along the arrows then each flip in p is the start of a path of encouraged-by links that ends at one of the n courageous flips.

One final case to exclude is that there might be two encouragement paths that go in the opposite direction over the same positions. This cannot happen:

Proposition 8. *Having both of the following encouragement paths is impossible:*

$$\perp \leftarrow (t_1, i_1 \mapsto b_1) \leftarrow (t_2, i_2 \mapsto b_2) \qquad\qquad \leftarrow \cdots \leftarrow \quad (t_m, i_m \mapsto b_m) \quad (19)$$
$$\perp \leftarrow (s_m, i_m \mapsto c_m) \leftarrow (s_{m-1}, i_{m-1} \mapsto c_{m-1}) \quad \leftarrow \cdots \leftarrow \quad (s_1, i_1 \mapsto c_1) \quad (20)$$

Proof. Without loss of generality (by relabeling), we can assume that $t_1 < s_1$. We can extend this with the following claim:

Claim: If $t_k < s_k$ then $t_{k+1} < s_{k+1}$

Since $(t_k, i_k \mapsto b_k) \leftarrow (t_{k+1}, i_{k+1} \mapsto b_{k+1})$, we have, for all $t \in [t_k + 1, t_{k+1}]$, $x^t[i_k] = b_k$. Thus we can't have i_k flipping in that interval, so $s_k > t_{k+1}$.

12

But now look at $(s_{k+1}, i_{k+1} \mapsto c_{k+1}) \leftarrow (s_k, i_k \mapsto c_k)$. This shows that we also have, for all $t' \in [s_{k+1} + 1, s_k]$, $x^{t'}[i_{k+1}] = c_{k+1}$. So for both flips at i_{k+1} to happen, we need $s_{k+1} > t_{k+1}$.

Applying the claim repeatedly gets us $t_m < s_m$. But this means that i_m flipped before $m(s_m)$, so by Proposition 5 $(s_m, i_m \mapsto c_m)$ could not have been courageous. □

*Proof (of **Theorem 6**).* Consider any path p in the fitness graph, and its corresponding flip function m. By the completeness of Definition 12, we know that every flip must have been either courageous or encouraged.

Any encouraged flip is the end-point of a unique (non-zero length) encouragement path in the constraint graph starting from some courageous flip (where Proposition 6 established that they're encouragement paths, not walks; and Proposition 7 established that the encouragement paths are uniquely determined by the variable positions that they pass through.) From Proposition 8, we know that there cannot be two encouragement paths that traverse the same positions but in opposite directions. Thus, there can only be as many non-zero-length encouragement paths as undirected paths in our constraint graph. Since our constraint graph is a tree, an undirected non-zero length path is uniquely determined by its pair of endpoints. Thus, there are at most $\binom{n}{2}$ of these paths.

From Proposition 5, there are at most n courageous flips (encouragement paths of length 0). Thus, our path p must have length at most $n + \binom{n}{2}$. □

6 Long paths in landscapes with simple constraint graphs

In this section we show that the conditions in Theorem 6 are essential. We exhibit binary VCSP-instances with very simple constraint graphs where the associated fitness graphs have exponentially-long directed paths.

Example 2. (**Domain size 3**) Consider a binary VCSP-instance $\mathcal{C}$, with variables $x_n, x_{n-1}, \ldots, x_2, x_1, x_0$, and constraints $\{C_{n,n-1}, \ldots, C_{32}, C_{21}, C_{10}\}$ over the uniform domain $D = \{0, 1, \triangleright\}$, where each constraint C_{ij} is represented by the following matrix:

$$C_{ij} = 3^{i-1} \begin{pmatrix} 1\ 2\ 3 \\ 2\ 3\ 1 \\ 3\ 1\ 2 \end{pmatrix}$$

Even though the constraint graph of $\mathcal{C}$ is just a path of length n, we now show the corresponding fitness graph, $G_{\mathcal{C}}$, contains a directed path of exponential length.

Notice that given two natural numbers $M, M' < 2^n$ written in binary as x^M, $x^{M'} \in \{0, 1\}^n$ with the least significant digit as x_0, we have that if $M' > M$ then $f(M') > f(M)$. Thus, counting up in binary from 0^{n+1} to 01^n is monotonically increasing in fitness. However, x^{M+1} is often more than a single flip away from x^M (consider the transition from $x^M = 01^n$ for an extreme example). We handle these multi-flip cases with our third domain value, $\triangleright$, as follows: (1) given $x^M =$

13

$y01^k$ where $y \in \{0,1\}^{n-k}$, we proceed to replace the 1s in the right-most block of 1s by $\rhd$, starting from x_{k-1}^M and moving to the right; (2) from $y0\rhd^k$ we can take a 1-flip to $y1\rhd^k$ (regardless of $y_0 = 0$ or 1); (3) from $x' = y1\rhd^k$, we replace the $\rhd$s by 0s, starting from the rightmost $\rhd$ (i.e., x_0') and moving to the left.

This lets our sequence of moves count in binary from 0^{n+1} to 01^n, while using extra steps with $\rhd$s to make sure all transitions are improving 1-flips; thus, this path in the fitness graph has a length greater than 2^n.

Our final example is a binary Boolean VCSP where the constraint graph has tree-width two and maximum degree three, but the associated fitness graph contains an exponentially long directed path. This example is a simplified and corrected version of a similar example for the MAX-CUT problem, described by Monien and Tscheuschner [15]. Note, however, that by allowing general valued constraints, instead of just MAX-CUT constraints, we are able to reduce the required maximum degree from 4 to 3.

Example 3. (**Tree-width 2**) Consider a binary Boolean VCSP-instance $\mathcal{C}$ with $n = 4K + 1$ variables. The constraint graph contains a sequence of disjoint cycles of length four, linked together by a single additional edge joining each consecutive pair of cycles. The final cycle is replaced by a single variable x_n with unary constraint $\begin{pmatrix} 0 \\ -w_K \end{pmatrix}$. Hence the constraint graph of $\mathcal{C}$ has maximum degree three and treewidth two. The i-th cycle (for $0 \leq i \leq K - 1$) has the following constraints (where the w_i values are defined recursively with $w_0 = 0$):
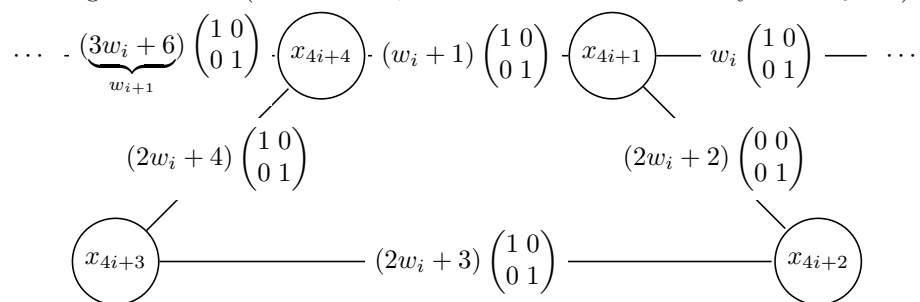
To begin the long path all variables are assigned 0, except $x_n = 1$. The path will proceed by always flipping variables in the smallest 4-cycle block possible.

Within each 4-cycle block, let us write the 4 variables by decreasing index as $x_{4i+4}x_{4i+3}x_{4i+2}x_{4i+1}$. We will make the following transitions within each cycle: if $x_{4(i+1)+1} = 1$ then we'll transition $0000 \to 1000 \to 1001 \to 1101$; if $x_{4(i+1)+1} = 0$ then we'll transition $1101 \to 0101 \to 0100 \to 0110 \to 0010 \to 0011 \to 0001 \to 0000$. Every time that x_{4i+1} is flipped from 0 to 1 or vice versa, we'll recurse to the $(i-1)$th cycle. Because x_{4i+1} ends up flipping from 1 to 0 twice as often as $x_{4(i+1)+1}$, this means that we double the number of flips in each cycle. Variable x_n will flip once, from 1 to 0, due to the unary constraint, which will cause $x_{4(K-1)+1}$ to flip twice from 1 to 0, which will cause $x_{4(K-2)+1}$ to flip four times from 1 to 0, and so on, until eventually this will cause x_1 to flip 2^K times from 1 to 0. Hence we have an improving path of length greater than 2^K.

7 Conclusion

In this paper, we have considered the broad class of fitness landscapes that can be modelled as arising from the combined effect of simple interactions of a few variables, where each of these interactions is described by an arbitrary valued constraint. Modelling fitness landscapes in this way allows us to classify them in new ways: for example by identifying a minimal constraint graph, and then characterising properties of this constraint graph.

We have shown that when a fitness landscape over Boolean variables has a (minimal) constraint graph that is tree-structured, then finding a local optimum by *any* local search algorithm takes only polynomial time. However, over a slightly larger domain, or allowing even slightly more general constraint graphs, we have shown examples where some local search algorithms can take exponential time to find even a local optimum.

Focusing on the maximum length of improving paths in a fitness graph, rather than the run-time of a particular local search algorithm, lets us use our results in settings where the details of the local search algorithm are unknown or highly contingent.

The most notable example of this is in modeling biological evolution. In the context of a model of biological evolution, each variable assignment represents the values of the alleles at a sequence of genetic loci. The constraint graph can then be interpreted as a gene-interaction network. The notion of sign-interaction that is central to Section 4 is based on the biological idea of *sign-epistasis* that is central to the analysis of evolutionary dynamics [17,18,6,11]. In such a model, different local search algorithms correspond to the evolutionary dynamics of populations with different sizes and structures [11]. Since the details of these population structures, and thus the precise evolutionary dynamics, are often unknown (or even potentially unknowable in historic cases), it is very helpful to be able to reason over wide classes of local search algorithms, as we do here.

In settings where locally optimal assignments cannot be efficiently found by any local search algorithm, the computational complexity and the combinatorial structure of the fitness graph can be viewed as an ultimate constraint, that prevents evolution from stabilizing at a local fitness peak [11]; such cases will give rise to *open-ended evolution*. By identifying which families of constraint graphs lead to intractable local search problems, we can therefore classify which forms of gene-interaction network enable open-ended evolution.

Beyond the context of biological evolution, we also believe that the tools for classifying fitness landscapes that we have begun to develop here will allow considerable further progress, and may eventually help to shed more light on the question of why local search algorithms can be extremely effective in practice. Another possible research direction is to use the analysis of constraint graphs and encouragement graphs to design more effective local search algorithms.

Acknowledgments

References

1. Aaronson, S.: Lower bounds for local search by quantum arguments. SIAM J. Comput. 35(4), 804–824 (2006), http://dx.doi.org/10.1137/S0097539704447237
2. Carbonnel, C., Romero, M., Zivny, S.: The complexity of general-valued CSPs seen from the other side. In: 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018. pp. 236–246 (2018)
3. Chapdelaine, P., Creignou, N.: The complexity of Boolean constraint satisfaction local search problems. Annals of Mathematics and Artificial Intelligence 43(1-4), 51–63 (2005), http://dx.doi.org/10.1007/s10472-004-9419-y
4. Cohen, D.A., Cooper, M.C., Creed, P., Jeavons, P.G., Zivny, S.: An algebraic theory of complexity for discrete optimization. SIAM J. Comput. 42(5), 1915–1939 (2013)
5. Cooper, M.C., De Givry, S., Schiex, T.: Optimal soft arc consistency. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence. pp. 68–73. IJCAI'07 (2007)
6. Crona, K., Greene, D., Barlow, M.: The peaks and geometry of fitness landscapes. Journal of Theoretical Biology 317, 1–10 (2013)
7. de Visser, J., Park, S., Krug, J.: Exploring the effect of sex on empirical fitness landscapes. The American Naturalist (2009)
8. Färnqvist, T.: Constraint optimization problems and bounded tree-width revisited. In: Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems CPAIOR. Lecture Notes in Computer Science, vol. 7298, pp. 163–179. Springer (2012)
9. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, CA. (1979)
10. Johnson, D., Papadimitriou, C., Yannakakis, M.: How easy is local search? Journal of Computer and System Sciences 37, 79–100 (1988)
11. Kaznatcheev, A.: Computational complexity as an ultimate constraint on evolution. Genetics 212(1), 245–265 (2019)
12. Kolmogorov, V., Zivny, S.: The complexity of conservative valued CSPs. J. ACM 60(2), 10:1–10:38 (2013), https://doi.org/10.1145/2450142.2450146
13. Llewellyn, D.C., Tovey, C.A., Trick, M.A.: Local optimization on graphs. Discrete Applied Mathematics 23(2), 157–178 (1989)
14. Malan, K.M., Engelbrecht, A.P.: A survey of techniques for characterising fitness landscapes and some possible ways forward. Information Sciences 241, 148 – 163 (2013), http://www.sciencedirect.com/science/article/pii/S0020025513003125
15. Monien, B., Tscheuschner, T.: On the power of nodes of degree four in the local max-cut problem. In: Calamoneri, T., Diaz, J. (eds.) Algorithms and Complexity. pp. 264–275. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
16. Ochoa, G., Veerapen, N.: Mapping the global structure of TSP fitness landscapes. Journal of Heuristics 24(3), 265–294 (Jun 2018)
17. Poelwijk, F., Kiviet, D., Weinreich, D., Tans, S.: Empirical fitness landscapes reveal accessible evolutionary paths. Nature 445, 383–386 (2007)
18. Poelwijk, F., Sorin, T.N., Kiviet, D., Tans, S.: Reciprocal sign epistasis is a necessary condition for multi-peaked fitness landscapes. Journal of Theoretical Biology 272, 141 – 144 (2011)
19. Schaffer, A., Yannakakis, M.: Simple local search problems that are hard to solve. SIAM Journal on Computing 20(1), 56–87 (1991)
20. Tayarani-Najaran, M., Prügel-Bennett, A.: On the landscape of combinatorial optimization problems. IEEE Trans. Evolutionary Computation 18(3), 420–434 (2014), https://doi.org/10.1109/TEVC.2013.2281502

21. Thapper, J., Zivny, S.: Necessary conditions for tractability of valued CSPs. SIAM J. Discrete Math. 29(4), 2361–2384 (2015), https://doi.org/10.1137/140990346
22. Thapper, J., Zivny, S.: The complexity of finite-valued CSPs. J. ACM 63(4), 37:1–37:33 (2016), https://doi.org/10.1145/2974019
23. Wright, S.: The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In: Proceedings of the Sixth International Congress on Genetics. pp. 355–366 (1932)