Partition Crossover can Linearize Local Optima Lattices of k-bounded Pseudo-Boolean Functions

Darrell Whitley
Colorado State University
Fort Collins, Colorado, USA
whitley@colostate.edu

Gabriela Ochoa
University of Stirling
Stirling, Scotland, UK
gabriela.ochoa@stir.ac.uk

Francisco Chicano
ITIS Software, Universidad de Málaga
Málaga, Spain
chicano@uma.es

ABSTRACT

When Partition Crossover is used to recombine two parents which are local optima, the offspring are all local optima in the smallest hyperplane subspace that contains the two parents. The offspring can also be organized into a non-planar hypercube "lattice." Furthermore, all of the offspring can be evaluated using a simple linear equation. When a child of Partition Crossover is a local optimum in the full search space, the linear equation exactly determines its evaluation. When a child of Partition Crossover can be improved by local search, the linear equation is an upper bound on the evaluation of the associated local optimum when minimizing. This theoretical result holds for all k-bounded Pseudo-Boolean optimization problems, including MAX-kSAT, QUBO problems, as well as random and adjacent NK landscapes. These linear equations provide a stronger explanation as to why the "Big Valley" distribution of local optima exists. We fully enumerate a sample of NK landscapes to collect frequency information to complement our theoretical results. We also introduce new algorithmic contributions that can 1) expand smaller lattices in order to find larger lattices that contain additional local optima, and 2) introduce an efficient method to find new improving moves in lattices using *score vectors*.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; *Combinatorial algorithms*; • **Theory of computation** → **Evolutionary algorithms**.

KEYWORDS

NK landscapes, Big Valley Hypothesis, Partition Crossover, Iterated Local Search

ACM Reference Format:
Darrell Whitley, Gabriela Ochoa, and Francisco Chicano. 2023. Partition Crossover can Linearize Local Optima Lattices of k-bounded Pseudo-Boolean Functions. In *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA '23), August 30-September 1, 2023, Potsdam, Germany*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3594805.3607129

FOGA '23, August 30-September 1, 2023, Potsdam, Germany

ACM ISBN 979-8-4007-0202-0/23/08…$15.00
https://doi.org/10.1145/3594805.3607129

1 INTRODUCTION

In this paper, we consider the set of all k-bounded pseudo-Boolean optimization problems, $f : \mathbb{B}^n \rightarrow \mathbb{R}$. Thus, the objective function $f(x)$ acts on a search space of Boolean inputs, $x \in \mathbb{B}^n$, and the output can be any real-valued or Boolean value. This set of problems include MAX-kSAT, Quadratic Unconstrained Boolean Optimiziation (QUBO) problems, as well as random and adjacent NK landscapes. Some of our proofs will also generalize to the Traveling Salesman Problem.

Partition Crossover is a deterministic form of recombination that has been successfully applied to k-bounded Pseudo-Boolean optimization problems. Partition Crossover [19] combined with iterated local search has been able to solve adjacent NK landscapes to globally optimal solutions for problems with one million variables [6]. Partition Crossover was also able to produce better results than state-of-the-art inexact MAX-kSAT problems when tested specifically on very hard problem instances taken from the SAT Competitions [5]. Finally, Partition Crossover operators (which include both Iterated Partial Transcription and Generalized Partition Crossover) are used by the highly successful Lin-Kernighan-Helsgaun [12] algorithm for the Traveling Salesman Problem [18].

One of the beneficial features of Partition Crossover is that it is deterministic. When Partition Crossover is successful, it produces a localized decomposition of the evaluation function into q linearly separable components, and it returns the best of 2^q offspring in $O(n)$ time. Given two parents that are locally optimal, the offspring are all locally optimal in the smallest hyperplane subspace (aka, the "largest order" hyperplane) that contains the two parents. This hyperplane is found by fixing all of the bits that the two parents share in common, thus creating a lower-dimensional subspace in which all of the offspring are locally optimal. A formal proof is given by Tinós et al. [19].

This paper differs from previous studies of Partition Crossover because it focuses on all of the offspring that can be produced by Partition Crossover, rather than the best possible offspring. This can provide new insights into the distribution of local optima in k-bounded pseudo-Boolean functions.

All of the children produced by Partition Crossover can be organized into hypercube structures of dimension q under Partition Crossover. Given a hypercube, every hyperplane of the same order has identical structure and the hypercube graph forms a non-planar lattice [15]. Thus, we can say that the children produced by Partition Crossover are organized into a "lattice." Whitley and Ochoa proved that these lattices can be exponentially large for the Traveling Salesman Problem [22].

For problem instances where data has been collected, the majority of offspring produced by Partition Crossover are also local

optima in the full search space when generated from two parents that are also locally optimal. Obviously, this is an empirical observation.

However, occasionally a specific offspring is not locally optimal in the full search space. In this case, an improving move can only come from a bit that the parents share in common; this immediately follows from the fact that all children are locally optimal in the smallest hyperplane that contains the two parents. Thus any improved string must lie outside that hyperplane. A formal proof is given by Tinós [19].

First, consider the case where all of the offspring produced by Partition Crossover are locally optimal in the full search space. In this paper, we show that all of these local optima can be evaluated using a simple linear equation of the form:

$$f(x) = \alpha_0 + \sum_{i=1}^{q} \alpha_i b_i \quad (1)$$

where b_i is a Boolean decision variable which simultaneously acts on disjoint subsets of variables in x. When b_i is flipped, a specific subset of variables selected from x also flip. This is less mysterious than it might sound because the variable b_i is deterministically selecting a subset of bits from either Parent 1 or Parent 2. Under Partition Crossover, the bits that are swapped between Parent 1 and Parent 2 are always complements.

What implications does this have? Let the vector

$$< 0, 1, 2, 3, 4, 5, 6, 7 >$$

denote the corners of a 3-d cube over 3 Boolean variables. (Each integer maps to a 3 bit string, e.g. 5 = 101.) Assume the numbers from 0 to 7 are indices mapping to 8 different local optima generated by Partition Crossover.

The evaluations of these local optima are determined by the values $\alpha_1, \alpha_2, \alpha_3$ in EQN 1. And the set of local optima indexed by <0, 1, 4, 5> can be shifted by a constant to yield a new set of local optima indexed by <2, 3, 6, 7> by changing α_2. Thus, the evaluations of children in one half of the lattice are related to evaluations of children in another half of the lattice by a single constant in a linear equation. Local optima are not just "randomly scattered" across the search space in some arbitrary fashion.

This happens no matter the size of the lattice (i.e. the hypercube of dimension q). Applications of Partition Crossover to real-world industrial MAX-kSAT problems have produced decompositions where $q > 1000$ and the lattice is larger than 2^{1000} [4].

Our empirical results also show that one local optimum can appear in hundreds of lattices, even when $n = 30$. The number of lattices can grow exponentially with n [22]. Thus, the fitness of one local optimum can be simultaneously constrained by multiple linear equations, one for every Partition Crossover event where that local optimum appears as an offspring.

Second, consider the offspring produced by Partition Crossover that are **not** locally optimal. The linear equation is now a bound on the evaluation of the associated local optimum, which is found by improving the child produced by Partition Crossover. But each offspring produced by Partition Crossover must be in a *different basin of attraction* associated with a different local optimum. This is also independent of how a local optimum is defined, as long as there exists a well defined neighborhood structure.

A local optimum that is not an offspring might be reached from offspring that appear in multiple non-overlapping lattices. In that case, the evaluation of the local optimum must be bounded by all of the linear equations that are associated will all of these lattices, but of course the tightest bound would seem to be the most relevant.

This paper makes two other contributions. Section 4 fully enumerates both random and adjacent NK landscapes of size $n = 30$ and $n = 40$. We do this to ask the following questions: How often is a child produced by Partition Crossover also a local optimum in the full search space? As one might expect, the probability is higher for adjacent NK landscapes than random NK landscapes. In addition, we asked if local optima that are close to the global optimum display a higher frequency of membership in lattices than local optima that are farther away from the global optimum. We found that Partition Crossover is more productive when combining parents that are closer to the global optimum, and the children of parents that are closer to the global optimum are also more frequently local optima.

Section 5 presents algorithmic contributions. Section 5.1 presents a simple way to discover larger lattices starting from smaller lattices. Sections 5.2 and 5.3 demonstrate how *score vectors* can be used to efficiently track improving moves associated with offspring that are not locally optimal. We prove that it is possible to evaluate q offspring while making only 1 call to the evaluation function because each offspring only updates a partial fragment of the evaluation function. While evaluating offspring, we can also update and maintain the *score vector* which can be used to identify both improving moves and any local optimum in $O(1)$ time after every bit flip.

2 K-BOUNDED BOOLEAN FUNCTIONS

Let $f(x)$ denote a k-bounded Boolean function

$$f(x) = \sum_{i=1}^{m} f_i(x)$$

where each subfunction $f_i(x)$ extracts and uses at most k Boolean variables from x. Each subfunction $f_i(x)$ knows which Boolean variables to use.

Boolean functions have binary inputs and outputs. Functions are pseudo-Boolean if the domain (the input) is binary but the co-domain (the output) can be real-valued. Thus, MAX-kSAT functions are Boolean but NK landscapes are pseudo-Boolean. Both the class of MAX-kSAT functions and random NK landscapes are NP-Complete.

It should noted that these k-bounded functions have very sparse nonlinearity. Since there are m subfunctions, and each subfunction has k variables, then in expectation every variable appears in $k\frac{m}{n}$ subfunctions. This would be the case for random MAX-3SAT instances and random NK landscapes.

One way to formally measure the nonlinearity of a function is to convert that function into a polynomial form. For example let $W(.)$ denote a Walsh Transform (AKA: a Hadamard Transform; AKA: a discrete Fourier Transform using a square wave).

$$W(f(x)) = \sum_{i=1}^{m} W(f_i(x))$$

Note that $W(f_i(x))$ produces at most $2^k - k - 1$ nonlinear coefficients. There are 2^k total coefficients in the polynomial, but k terms are linear and 1 term is a constant. For example, for MAX-3SAT, there are only 4 nonlinear terms. Thus, if $m = O(n)$ then the number of nonlinear coefficients in a k-bounded pseudo-Boolean function is also $O(n)$.

Real-world problems empirically have much lower levels of nonlinearity than randomly generated problems [9]. One might speculate that this is true because real-world problems have some form of regular structure. However, one can also prove that the reductions and transforms which are used to generate k-bounded pseudo-Boolean functions also result in low levels of nonlinearity.

For example, Cormen's "Introduction to Algorithms" textbook [8] reduces the following SAT expression into a MAX-3SAT expression.

$$(y1 \iff (y2 \wedge \neg x2))$$

results in the following four clauses in Conjunctive Normal Form:

$$(\neg y1 \vee \neg y2 \vee \neg x2) \wedge (\neg y1 \vee y2 \vee \neg x2)$$
$$\wedge (\neg y1 \vee y2 \vee x2) \wedge (y1 \vee \neg y2 \vee x2)$$

Note that these four clauses can also be combined into a single Boolean subfunction $f_i(y1, y2, x2)$. It follows that these four clauses yield at most 4 nonlinear terms, or 1 per clause. This is consistent with the findings of Hains et al. [9], who found that the median number of total terms per clause in 14 real-world industrial MAXSAT problems was 1.17, which yields approximately 1 nonlinear term per clause if m/n = 4.2735.

Consider a slightly different set of four MAX-3SAT clauses.

$$(\neg y1 \vee \neg y2 \vee x2) \wedge (\neg y1 \vee y2 \vee \neg x2)$$
$$\wedge (\neg y1 \vee y2 \vee x2) \wedge (y1 \vee \neg y2 \vee x2)$$

Only the first clause changed. When these four clauses are converted into a single subfunction, that subfunction is quadratic (it is a QUBO subfunction). This is because the 3-way interactions cancel. When four clauses are generate randomly using the same 3 variables, a QUBO will result with probability 36/70 > 0.51. There are 4 ways for one of 4 clauses to be false, and (2^3 choose 4) = 70. Enumerating all 70 cases yields the stated result.

Transforms [3] also exist that convert pseudo-Boolean functions into k-bounded or quadratic pseudo-Boolean functions. The use of transforms and NP-Completeness reductions can also yield problem instances with very low levels of nonlinearity, even when the original problem instance is random.

In effect, transforms and reductions add additional variables in order to achieve very low levels of bounded nonlinearity.

3 PARTITION CROSSOVER (PX)

Converting problems into k-bounded pseudo-Boolean functions allows us to use Graybox crossover operators that also exploit bounded nonlinearity. In this paper, we use the Partition Crossover operator to induce q-dimensional lattices over subsets of local optima. The following example, labeled F1, illustrates key ideas behind Partition Crossover. F1 has $n = 24$ variables and $m = 18$ subfunctions. Every subfunction of F1 is labeled using the boldface letters **"a"** to **"r"** and takes in 3 variables. The variables are labeled with integers from 1 to 24, where integer i denotes x_i. This table shows the membership of variables in subfunctions.

F1:	**a**: 1 2 3	**f**: 6 7 23	**k**: 11 13 22	**p**: 15 16 17
	b: 2 3 4	**g**: 8 9 10	**l**: 11 20 21	**q**: 16 17 20
	c: 3 4 5	**h**: 8 9 22	**m**: 11 21 22	**r**: 18 19 21
	d: 4 5 6	**i**: 8 10 23	**n**: 14 15 24	
	e: 5 6 7	**j**: 11 12 13	**o**: 14 16 17	

In this case, a boldface **f** denotes a subfunction and $\mathbf{f}(x) \neq f(x)$. In a MAX-3SAT problem, each subfunction would denote a clause in Conjunctive Normal Form, and we would evaluate each clause. In an NK landscape, we would generate a random lookup table for the binary assignments to the variables in each subfunction.

Partition Crossover constructs the Variable Interaction Graph (VIG) of the k-bounded function. Figure 1a presents an image of the VIG for example F1. If $f(x)$ has n variables and m subfunctions, the VIG has n nodes and $O(k^2m)$ edges when k is bounded. If $m = O(n)$ the VIG can be computed $exactly$ in $O(n)$ time using the Fourier Transform. The VIG can also be defined heuristically: create a vertex for each variable in $f(x)$, and create an edge between two variables if they appear together in a subfunction of $f(x)$. Often, the heuristic VIG is only slightly larger than the exact VIG. The VIG is only computed once.

Assume that the parents P1 and P2 are also local optima under a single bit-flip neighborhood. Partition Crossover decomposes the VIG by fixing common variable assignments in P1 and P2. A "common assignment" means that both parents have the same assignment (0 or 1) for a particular variable x_i. After variables with common assignments are removed, the VIG is often decomposed into q connected subgraphs. Decomposing the VIG also decomposes $f(x)$ into linearly separable subfunctions. In our example F1, let the two parent strings P1 and P2 be:

$$P1 \;=\; 01000\ 01100\ 10000\ 01101\ 1111$$

$$P2 \;=\; 10111\ 10011\ 01111\ 10011\ 1111$$

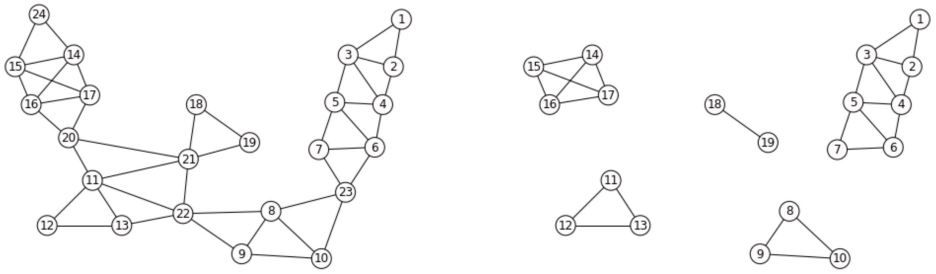
These two solutions share ***** ***** ***** ****1 1111 as common bits, where * represents bits that have complementary assignments in the parent solutions. When the VIG is decomposed, vertices 20 through 24 (x_{20} to x_{24}) are deleted, leaving five recombining components. Figure 1b shows the recombination graph associated with the VIG for example F1.

When any two random strings of length n are generated, the number of matching bits (both 0, or both 1) at any position i has a Binomial distribution with mean $n/2$. Thus, half the bits in two random binary strings are expected to match. We might expect two local optima to be closer in Hamming distance than random bitstrings, but this may not be true for all problem domains.

The q components of the decomposed VIG represent 2^q candidate solutions of $f(x)$ because each component can either take assignments from P1 or P2. Mathematically, P1 and P2 are included in the set of 2^q possible children.

Let x' denote any string where the shared fixed bits in P1 and P2 are removed. Some subfunctions in $f(x)$ may only be associated with fixed variables, and thus these subfunctions only contribute to some constant c. When the VIG is decomposed, $f(x)$ is also decomposed into a set of new subfunctions $g_1(x')\ldots g_q(x')$. Then

$$f(x) = g(x') + c = \sum_{i=1}^{q} g_i(x') + c \tag{2}$$



(a) The VIG of F1 (b) The Recombination Graph

Figure 1: Variable interaction graph (VIG) for example F1 as well as the decomposed recombination graph for P1 and P2.

In Figure 1 we have the following specific example, where the variables and indexes of $g(x')$ are arbitrarily sorted by index:

$$g(x') = g_1(x_1, x_2, x_3, x_4, x_5, x_6, x_7) + g_2(x_8, x_9, x_{10}) + g_3(x_{11}, x_{12}, x_{13}) + g_4(x_{14}, x_{15}, x_{16}, x_{17}) + g_5(x_{18}, x_{19})$$

Note that the function $g(x')$ is **separable.** The set of subfunctions $g_1(x')\ldots g_q(x')$ are disconnected subgraphs of the recombination graph. Thus, for any i and j ($i \neq j$) the subfunctions $g_i(x')$ and $g_j(x')$ do not have any shared variables. Each $g_i(x')$ also maps to a unique subset of the subfunctions as well. Recall that x' denotes a string where the variables x_{20} to x_{24} are fixed (to 1 in this case). Also recall that boldface is use to represent subfunctions (for example **f, g, i, j, k, n, m**). Then we can see that:

$$g_2(x_8, x_9, x_{10}) = \mathbf{g}(x_8, x_9, x_{10}) + \mathbf{h}(x_8, x_9) + \mathbf{i}(x_9, x_{10})$$
$$g_3(x_{11}, x_{12}, x_{13}) = \mathbf{j}(x_{11}, x_{12}, x_{13}) + \mathbf{k}(x_{11}, x_{13}) + \mathbf{l}(x_{11}) + \mathbf{m}(x_{11})$$
$$g_5(x_{18}, x_{19}) = \mathbf{r}(x_{18}, x_{19})$$

and more generally for x':

$$g_1(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = \mathbf{a}(x') + \mathbf{b}(x') + \mathbf{c}(x') + \mathbf{d}(x') + \mathbf{e}(x') + \mathbf{f}(x')$$
$$g_4(x_{14}, x_{15}, x_{16}, x_{17}) = \mathbf{n}(x') + \mathbf{o}(x') + \mathbf{p}(x') + \mathbf{q}(x')$$

Thus, the recombining components decompose both the set of subfunctions as well as the set of variables.

Exactly the same function $g(x')$ can be constructed for Partition Crossover when applied to the Traveling Salesman Problem [18].

3.1 A Linear Equation for Children

We next show how to convert Equation 2 into a simple linear equation that can be used to evaluate all of the children of Parents P1 and P2.

For each recombining component from $i = 1$ to q we define components of a weight vector α as follows:

$$\alpha_i = g_i(P1) - g_i(P2)$$

where α_i represents the change in evaluation when changing only the bits in the i^{th} recombining component. The subfunction g_i selects the correct bits to evaluate the i^{th} recombining component.

Let b denote an auxiliary binary string (a vector of variables) of length q. Let $b_i = 0$ denote that the child inherits the bits from Parent 2 in the i^{th} recombining component. By symmetry, $b_i = 1$ denotes that the child inherits the bits from Parent 1 in the i^{th} recombining component. Inheritance from P1 is represented by $b = 1^q$ and inheritance from P2 is represented by $b = 0^q$.

Theorem 3.1. *Under Partition Crossover, all of the 2^q children of parents P1 and P2 can be evaluated using the following linear equation and the auxiliary bit function b.*

$$f(x) = g(x') + c = \alpha_0 + \sum_{i=1}^{q} \alpha_i b_i \tag{3}$$

where $\alpha_0 = f(P2)$.

Proof. By construction, the set of subfunctions $g_1(x)\ldots g_q(x)$ are linearly separable. The coefficient α_i computes the change which occurs when the bits in a single recombining component (indexed by i) all flip simultaneously from the assignment in P2 to the assignment in P1. By definition, when $b_i = 0$ the evaluation of $g_i(x')$ from P2 are already includes in α_0. Therefore, when $b_i = 1$, the change in the evaluation of $g_i(x'_1)$ is computed by $\alpha_i = g_i(P1) - g_i(P2)$. □

Corollary 3.2. *If all of the children of Partition Crossover are local optima, then all of those local optima can be evaluated using linear equation (2). If a child of Partition Crossover is not a local optimum, linear equation (2) provides an upper bound (when minimizing) on the fitness of all local optima that can be reached by taking at least one improving move starting from that child.*

Obviously, linear equation (2) provides a bound on the fitness of any local optimum that can be reached from a child of Partition Crossover since local search can only improve on the evaluation of the original child. Theorem 3.1 holds for all k-bounded pseudo-Boolean functions as well as the Traveling Salesman Problem which uses the same function $g(x')$.

We next show that local optima can sometimes be grouped into pairs, such that each pair has the same combined fitness.

Theorem 3.3. *For any child C_i in the lattice produced by Partition Crossover, denote its complement by $\bar{C}_i$. The following equalities hold:*

$$f(P1) + f(P2) = f(C_i) + f(\bar{C}_i) = \frac{1}{2^{q-1}} \sum_{i=1}^{2^q} f(C_i) \tag{4}$$

Proof. First consider $f(P1) + f(P2) = f(C_i) + f(\bar{C}_i)$. In Equation 3 inheritance from P1 is represented by $b = 1^q$ and from P2

is represented by $b = 0^q$. By definition, if string b generates child C under PX, then $\bar{b}$ generates $\bar{C}$. Since $g(x')$ is linearly separable with respect to representation b it follows that $f(C) + f(\bar{C}) = f(P1) + f(P2)$. And on average $f(P1) + f(P2)/2 = \frac{1}{2^q} \sum_{i=1}^{2^q} f(C_i)$ and $f(P1) + f(P2) = \frac{1}{2^{q-1}} \sum_{i=1}^{2^q} f(C_i)$. □

Clearly, if all of the children produced by Partition Crossover are locally optimal, then all of the associated pairs of local optima will have the combined fitness of $f(P1) + f(P2)$.

The following are some additional consequences of Theorem 3.3. Let C_b denote the best child and C_w the worst child. Because C_b and C_w must be complements $f(P1) + f(P2) = f(C_b) + f(C_w)$.

Next, assume there are 3 recombining components and that we define 8 possible children relative to a 3-bit auxiliary function $\mathcal{G}$ over the recombining components.

Let $\hat{g}(b)$ be a proxy function for $g(x') = g_1(x') + g_2(x') + g_3(x')$ where b again determines inheritance of recombining components from Parent P1 or P2. If all 8 children are local optima in the full search space, then Theorem 3.3 tells us:

$$\hat{g}(111) + \hat{g}(000) = \hat{g}(001) + \hat{g}(110) = \hat{g}(010) + \hat{g}(101) = \hat{g}(100) + \hat{g}(011)$$

These same equalities must also hold for function $f(x)$ when evaluating the children of Partition Crossover.

There is also a recursive decomposition based on hyperplane slices of the hypercube lattice. Thus, it is true (for example) that if we fix $b_1 = 0$:

$$\hat{g}(011) + \hat{g}(000) = \hat{g}(001) + \hat{g}(010)$$

This recursive decomposition must hold for any linear function.

Why does this matter? These equalities strongly constrain the intervals between the evaluations of local optima in the lattice. These local optima do not and cannot have arbitrary evaluations.

3.2 The Hamming Distance Linear Equation

Let the function $HD(x, y)$ measure the Hamming distance between bit strings x and y (where string length is n). By convention, a global optimum is represented by string x^*. Generalize the function $HD_{g_i}(x', x^*)$ so it also computes Hamming Distance only over the bits in the recombining component represented by $g_i(x')$.

We again use the auxiliary string b and will compute the coefficients (denoted by β_i) of a linear equation. Let $\beta_0 = HD(P2, x*)$. Let $\beta_i = HD_{g_i}(P1, x^*) - HD_{g_i}(P2, x^*)$.

Theorem 3.4. *Under Partition Crossover, the Hamming distance between a global optimum x^* all of possible 2^q children of parents P1 and P2 can be evaluated using the following linear equation and the auxiliary bit function b.*

$$HD(x, x^*) = \beta_0 + \sum_{i=1}^{q} \beta_i b_i. \quad (5)$$

Proof. By definition, the calculation of Hamming distance is already a linear function. When $b_i = 0$ the calculation of $HD_{g_i}(P2, x^*)$ is already included in β_0. When $b_i = 1$, the expression $\beta_i = HD_{g_i}(P1, x^*) - HD_{g_i}(P2, x^*)$ computes the change in Hamming distance when the bits in a single recombining component (indexed by i) all flip simultaneously from the assignment in P2 to the assignment in P1. □

It follows that the Hamming function $H(x, x^*)$ is also symmetric and

$$HD(C) + HD(\bar{C}) = HD(P1) + HD(P2)$$

when evaluating the children of Partition Crossover. Therefore, lattices (and sub-lattices) are symmetric both in terms of fitness, but also in terms of Hamming distance.

3.3 An Example Lattice

In Figure 2, 16 different local optima are shown, taken from an adjacent NK landscape with N=40 and k=3. All of these 16 local optima were generated by one application of Partition Crossover: **all of the children are also local optima.** Note the strong symmetries which exist in the set of local optima. Let μ denote the average of all of these locally optimal children; we can think of μ as being a "centroid" of these fitness values. The average of all pairs of complements in the lattice of offspring is also μ.

The coefficients for the linear equation in this case are given by

$$f(x) = f(Parent2) + \alpha_1 b_1 + \alpha_2 b_2 + \alpha_3 b_3 + \alpha_4 b_4$$

$$f(x) = 2754 + 6b_1 + 19b_2 + 64b_3 + 71b_4 \quad (6)$$

In Figures 2 and 3 the constant ($f(P2) = 2754$) has been dropped.

In Figure 2, the linear equation (2) also tells us that all of the local optima in hyperplanes 00** and 11** and 01** and 10** must be identical in fitness, except shifted by a constant in both the y axis (the evaluation function) and the x axis (Hamming distance). The constant must be the sum of one or more coefficients in linear equation 3 (the α terms) and linear equation 5 (the β terms). Fixing the same bits induces a lower dimensional space in each case. This can be seen in Figure 3 where 3 of $6 = \binom{4}{2}$ subspaces are highlighted. The ellipses show different shifts of the evaluation function and Hamming distance in different hyperplane subspaces of the lattice in Figure 2.

Note that there is nothing special about this particular example (except that all of the children are also local optima). This same kind of shifting must occur in all lattices that might be found in any arbitrary k-bounded pseudo-Boolean problem instance. We also sampled a random NK landscape (n=40, k=3) with 4 recombining components where all of the children were also local optima. In this case, $f(P2) = 2735$ and the α coefficients (i.e. to compute fitness) are

$$f(x) = 2735 - 60b_1 + 88b_2 + 80b_3 - 19b_4. \quad (7)$$

The same shifting must also occur for the Traveling Salesman Problem when using Partition Crossover or Helsgaun's IPT operator [18].

4 LATTICES AND THE BIG VALLEY HYPOTHESIS

The "Big Valley" hypothesis asserts that local optima which are closer to the global optimum in evaluation are generally also closer to the global optimum in the representation space [1, 2]. The Big Valley hypothesis conjectures that heuristic search methods are able to escape one local optimum by moving a short distance into the basin of another local optimum. By repeatedly "hopping" from local optimum to local optimum, iterated local search can (often) make progress toward a global optimum.

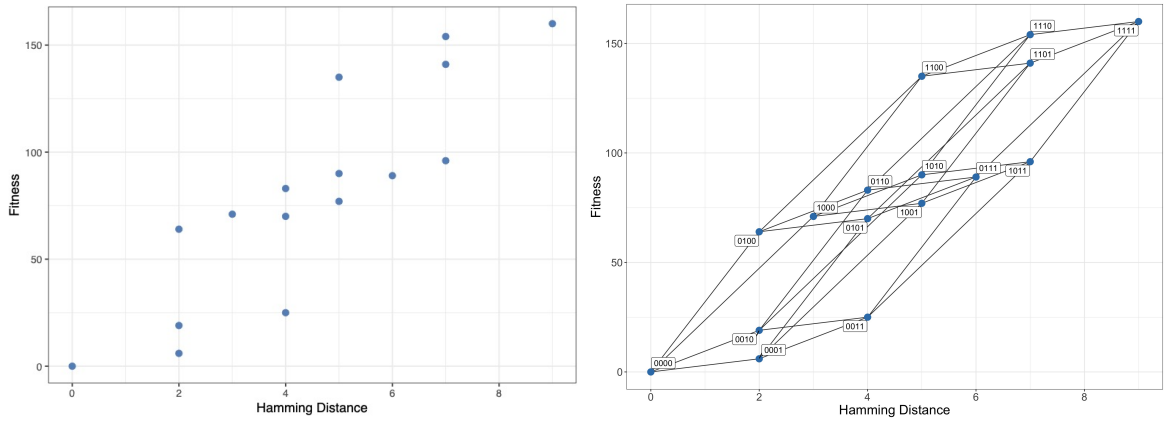


Figure 2: A sample of 16 optima from a Big Valley distribution of local optima (leftmost). These optima were generated under one application of Partition Crossover. The hypercube recombination graph is also shown (rightmost).

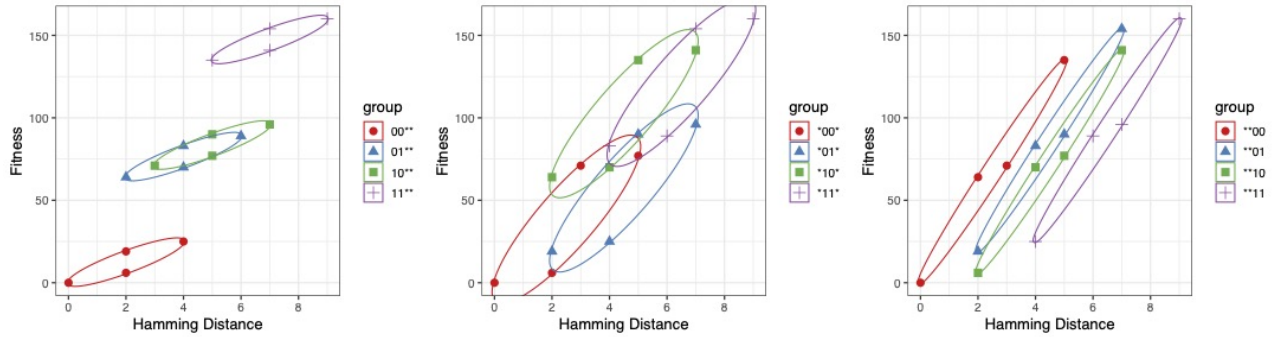


Figure 3: The same local optima shown in Figure 2 are shown here grouped as lower dimensional hyperplanes of the larger lattice. These are 3 of 6 configurations. Each frame shows how lower dimensional lattices are identical but shifted by a constant. All of the children are local optima, and each constant can be exactly derived from the coefficients of linear equations for Hamming distance (Eq. 5) and fitness (Eq. 6). Lower dimensional lattices can also be used to find higher dimensional lattices.

This Big Valley Hypothesis has been used to explain the effectiveness of heuristic search methods for combinatorial optimization. Intuitively, heuristic search methods often escape one local optimum by moving a short distance into the basin of another (hopefully better) local optimum. This implies some global structure. However, no general mechanism has been proposed that can adequately explain why the Big Valley structure should exist, or why it should occur so commonly in instances of NP-Hard problems. And it should be noted, there are also examples of landscapes with multiple funnels where the Big Valley (or largest funnel) will lead one away from the global optimum [10]; however, these cases are often associated with continuous optimization problems, and some hard combinatorial problems such as the quadratic assignment problem (QAP) [17].

The linear equations introduced in this paper provide a stronger explanation as to why the Big Valley distribution of local optima exists, and why both genetic recombination and iterated local search are able to exploit the Big Valley distribution of local optima. Local optima which are children of Partition Crossover have fitness values that are determined by a relatively small number of linear coefficients. Thus, local optima that co-occur together in lattices must also display similar fitness trends relative to their distance from the global optimum.

Consider the case where all of the children of Partition Crossover (including the parents) are local optima. All of the children belong to a lattice and the orientation of the lattice is determined by linear equations 3 and 5. If the change in Hamming distance and in fitness evaluation are positively correlated (assuming minimization), the entire lattice of local optima must contribute to a Big Valley distribution.

4.1 Exploring the Big Valley Distribution

The previous section starts to explain the distribution of many local optima that are part of Big Valley distributions. Our empirical data was collected from NK landscapes, where N=n and K=k-1. The NK look-up tables (i.e., the subfunctions) were populated with random integers between 1 and 100 (see [16]). In those lattices that are composed entirely of local optima, an improvement in fitness

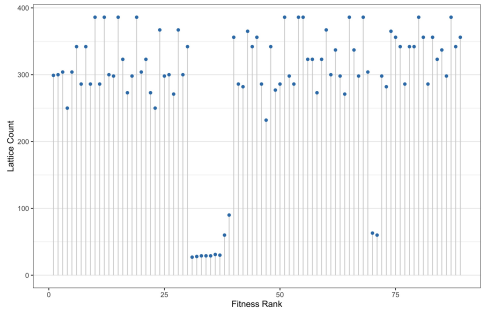


Figure 4: All of the local optima of an adjacent NK landscape (N=30, k=3) are plotted with fitness rank on the x-axis. The y-axis counts how often that optimum appears in a lattice. The minimum membership is 27 lattices; the average is 288.

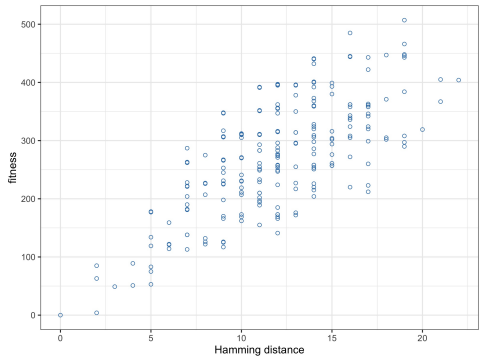


Figure 5: This is a classic example of the Big Valley distribution of local optima. The fitness evaluation is on the y-axis and the Hamming distance from the global optimum is on the x-axis. The global optimum was captured by coincidence.

is often matched by a reduction in Hamming distance across the entire lattice.

Of course, it is possible that a local optimum is not part of any lattice. But the empirical data suggests this might be rare. In Figure 4, we enumerated all of the local optima in an adjacent NK landscape for n=30 and k=3. There were 89 local optima in the search space. On average every local optimum appears in 288 lattices. Every local optimum appears in a minimum of 27 lattices. Figure 4 also displays locality effects: local optima with similar fitness may also appear in a similar number of lattices. This may be because these local optima share some (but not all) of the same recombining components.

In Figure 5, we present a classic illustration of the Big Valley distribution sampled from the same NK landscape (N=30 and k=3). Local optima that are nearer to the global optimum in Hamming distance are also nearer to the global optimum in fitness evaluation. This cloud of local optima looks as if the local optima might be randomly distributed. But they are not. In this case, the samples were filtered so that all of the children of Partition Crossover are also local optima, and we filtered to select a variety of lattice sizes. A lattice containing the global optimum was sampled by coincidence.

In Figure 6 we present the distinct lattices that make up the cloud of local optima seen in Figure 5. All of the distinct lattices are symmetric, with shifted fitness values and shifted Hamming distance relative to the global optimum. We can see that most individual lattices also show a Big Valley bias. But not all lattices are aligned with the Big Valley distribution. Lattice L9 is anti-correlated with the Big Valley distribution. However, for most lattices, the best child is also closest to the global optimum. Taken together the lattices yield the Big Valley distribution found in Figure 5.

5 THE NK LANDSCAPE EMPIRICAL DATA

Following the convention of NK landscapes, N=n. We generated 10 adjacent and 10 random NK landscapes with N=40 and N=30 and K+1 = k = 3. We then enumerated all of the local optima in the search space. We then recombined **every** local optimum with every other local optimum. This represents all possible recombinations of parents that are locally optimal.

Our first question: What percentage of the application of Partition Crossover produced children and lattices?

```
N=30 Adjacent NK:  88.93%      N=30 Random NK:  21.21%
N=40 Adjacent NK:  96.84%      N=40 Random NK:  32.40%
```

We already see that random and adjacent NK landscapes are very different (also see [20]). Partition Crossover is much more successful when recombining adjacent NK landscapes. However, the success rate of Partition Crossover appears to increase with problem size for both random and adjacent NK landscapes. That is logical: longer strings provide more opportunities for decomposition.

Our second question: What percentage of the children of Partition Crossover are also local optima?

```
N=30 Adjacent NK: 83.04%      N=30,  Random NK:  74.43%
N=40 Adjacent NK: 79.91%      N=40,  Random NK:  71.29%
```

There is again a difference between random and adjacent NK landscapes, but here the difference is less extreme. And now the percentage of children that are also local optima seems to decrease with problem size.

This also means that in 20% to 30% or more applications of Partition Crossover, there is an opportunity to improve one of the children using local search. Of course, modern local search methods for k-bound pseudo-Boolean functions do not randomly flip bits and do not enumerate the neighborhood. But it is worth pursuing these improving moves and what is the best way to do that? We address this question in Sections 6.2 and 6.3

All of our empirical data is presented in Table 1. Table 1 summarizes data over **all** of the local optima, and also over **the best half** of the local optima (all of those with above median fitness).

The Table counts the average number of local optima for each function. It also counts the number of lattices of size 4, size 8, size 16, size 32, size 64, and size 128. Recall that with q recombining components, the size of the lattice is 2^q.

The Table reports the Partition Crossover success rate, which means that an application of Partition Crossover resulted in recombining components and that children were produced. The success

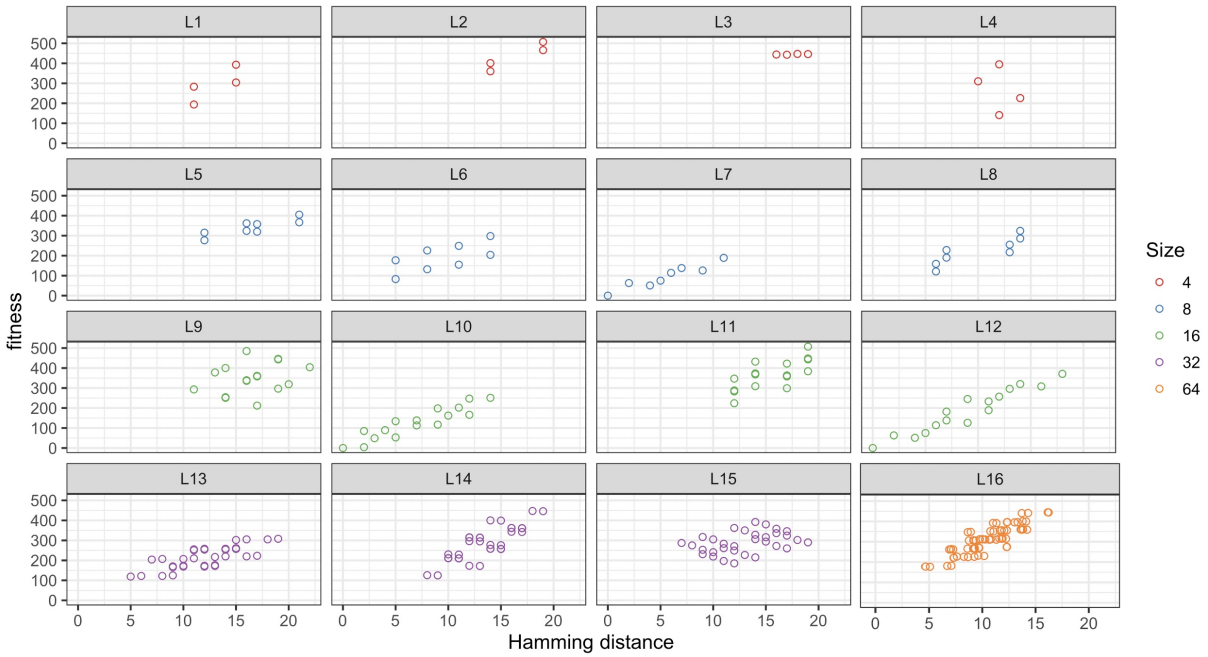


Figure 6: Lattices over the set of local optima seen in the Big Valley distribution shown in Figure 4. Every point is a local optimum. The constraints and symmetries governing the fitness evaluations are obvious in these individual lattice distributions. Some jitter was added to L16 of size 64 in order to visualize points that have very similar evaluations.

rate for adjacent NK landscapes is extremely high (> 78% for N = 40). The success rate for random NK landscapes is lower, but still substantial (>30 % for N=40).

The number of children of Partition Crossover that are also locally optimal is high. For adjacent NK landscapes and N=40, 80% of the children of Partition Crossover are also locally optimal. For random NK landscapes and N=40, 71% of the children of Partition Crossover are also locally optimal. These percentages are higher for local optima in the upper 50 percentile; these are the local optima that are closer to the global optimum in fitness.

Lattices produced by recombining local optima in the upper 50 percentile are also approximately 3 times more likely to contain the global optimum. The probability of a lattice containing the global optimum was greater than 0.5% for N=30 and N=40, for both adjacent and random NK landscapes when the parents are sampled from the upper 50 percentile.

6 ALGORITHMIC CONTRIBUTIONS

When a large lattice does exist where all of the children are local optima, we should expect to find smaller sub-lattices of this lattice with high frequency. In general, if there are q recombining components in a larger lattice, there are $2^q - (q+1)$ sub-lattices with some dimension less than q but greater than 1 or 0. If the children of the large lattice are all local optima, then (obviously) all of the children of the approximately 2^q sub-lattices are also all local optima. It follows that the recombination of parents which are local optima is much more likely to discover a smaller sub-lattice of a larger lattice than it is to discover the larger lattice itself.

We also motivate this section by referring to Figure 3. The lattice in this case is 4 dimensional. However, the decompositions that are shown are 2 dimensional. Assume that we have discovered some (but not all) of the 2-dimensional lattices that make up the 4-dimensional lattice. Can we use this information to find the higher dimensional lattice? In some cases, the answer is immediately "yes". In other cases, there are strategies that will improve the chances of finding higher dimensional lattices.

Consider Figure 3 again. Assume we have not found the full 4-dimensional lattice, but we have found two of the associated 3-dimensional lattices.

For example, if one 3-dimensional lattice has coefficients

$$f(x) = c1 + 6b_1 + 19b_2 + 64b_3$$

and another 3-dimensional lattice has coefficients

$$f(x) = c2 + 6b_1 + 19b_2 + 71b_4$$

then recovering the 4-dimensional lattice is trivial.

What precisely happens in this case? Consider 3 parents: P2, P1, P3. In the 4-dimensional hypercube for the auxiliary function b let P2 be represented by 0000, P1 is represented by 0111 and P3 is represented by 1011. Then P2 $\otimes$ P1 has coefficients $\alpha_1, \alpha_2, \alpha_3$ and P2 $\otimes$ P3 has coefficients $\alpha_1, \alpha_2, \alpha_4$.

Of course this case is special. Because P2 is involved in both applications of Partition Crossover and is the basis of the constant, the α_i values overlap. In general, this rarely happens.

However, more generally we can look for any pair of successful applications of Partition Crossover that share at least one identical recombining component. The following *Lattice*

Table 1: Empirical data collected by enumerating all local optima for N=30 and N=40 (k=3) NK landscapes. All numbers are averages of 10 instances, except Adjacent N=40 where there were 8 instances. Recombination crossed every local optimum with every local optimum (columns labeled "ALL"); or Recombination crossed the best half of the local optima with each other (columns labeled "Best 1/2"). Over 70% of all children were also local optima, but this number decreased as N increased.

	N = 30 Adjacent		N = 30 Random		N = 40 Adjacent		N = 40 Random	
	ALL	Best 1/2	ALL	Best 1/2	ALL	Best 1/2	ALL	Best 1/2
Count of all Local Optima	865	432	485	243	5 448	2 729	3 468	1 734
Count of size 4 lattices	270 774	73 668	27 348	10 890	4 071 180	1 015 428	1 858 411	470 687
Count of size 8 lattices	255 536	69 574	4 280	2 753	7 718 967	1 983 148	461 645	124 739
Count of size 16 lattices	69 390	18 213	284	126	5 334 851	1 339 630	68 703	18 913
Count of size 32 lattices	4 725	1 190	1	1	1 553 133	316 041	4 384	1 482
Count of size 64 lattices	48	11	0	0	125 394	23 250	128	54
Count of size 128 lattices	0	0	0	0	2 673	137	0	0
Total Lattice Count	600 473	162 656	31 915	12 934	18 639 384	4 677 638	2 388 035	615 875
Crossover Success Rate	88%	96%	21%	30%	79%	96%	32%	33%
Locally Optimal Children	83%	86%	74%	84%	80%	84%	71 %	77 %
Global Opt. as Improved Child (Count)	441	247	360	233	9 438	5 375	2 027	2 621
Global Opt. as Direct Child (Count)	1 516	1 221	213	81	19 244	19 222	1 566	1 176
Lattices containing Global Opt.	0.32%	0.90%	1.79%	2.40%	0.15%	0.52%	0.15%	0.61%

Discovery Theorem provides a road-map that can help guide search toward the discovery of larger lattices.

6.1 The Lattice Discovery Theorem

Theorem (The Lattice Discovery Theorem): *Consider parents $P1 \otimes P2$ and $P3 \otimes P4$. Assume there is a recombining component rc1 that is shared by both sets of parents. With loss of generalization, assume there is at least one recombining component rc2 that is only found in $P1 \otimes P2$. If rc2 has no nonlinear interaction with the recombining components already produced by $P3 \otimes P4$ then the lattice associated with $P3 \otimes P4$ is a sub-lattice of a larger lattice.*

Proof: The proof immediately follows from the definition of what it means to be a recombining component. If rc2 has no nonlinear interaction with the recombining components already produced by $P3 \otimes P4$ then rc2 can only connect to bits that are shared in common by parents $P3$ and $P4$. Transplanting bit assignments from $P1$ to $P3$ and from $P2$ to $P4$ associated with recombining component rc2 will create a new set of parents which must also yield another recombining component □.

Corrollary: Under the Lattice Discovery theorem every recombining component that is only found in $P1 \otimes P2$ but not in $P3 \otimes P4$ represents an opportunity to find a larger lattice. This can also apply in reverse for recombining components found in $P3 \otimes P4$ but not in $P1 \otimes P2$.

This theorem provides an algorithmic strategy to search for larger lattices, and suggests other strategies as well. For example, assume that $P1 \otimes P2$ yields 32 recombining components, and $P3 \otimes P4$ yields 20 recombining components, but only 2 recombining components are shared. Hypothetically, it may be possible to merge these recombining components to obtain a lattice of dimension 50 (32+20-2).

Furthermore, even if we only know about $P1 \otimes P2$ but do not have $P3$ or $P4$, local search can fix the bits found in the known recombining components for $P1 \otimes P2$ and look for other local optima that share the known recombining components with $P1$ and $P2$. It is even possible that a library of recombining components could also be used to construct solutions by assembling strings made up of previously discovered non-overlapping recombining components. Thus, even when there are no shared recombining components, this strategy can be used to directly search for pairs of parents that do share recombining components. Finding larger lattices will improve the chances of finding an improved local optimum or even a global optimum.

The Lattice Discovery Theorem can be supported by data structures which efficiently store information about known recombining components. This includes information about which bits are involved, and the partial evaluation of the recombining component. This data structure would make it possible to determine if an application of Partition Crossover involves a previously discovered recombining component.

There are also situations which are not covered by the Lattice Discovery Theorem. For example, assume that there exists a lattice with 4 recombining components denoted by rc1, rc2, rc3 and rc4. There could exist an application of Partition Crossover which uses rc1 and rc2 and a second, different application of Partition Crossover that uses rc3 and rc4. In this case, there are no shared recombining components. However, this case might also be detected because disjoint subsets of bits are involved.

6.2 Lattices and Improving Moves

Consider a lattice with $q = 3$, and 8 children. We know the two parents are local optima. Let parent P1 be denoted by recombination

mask $b7 = 111$ and parent P2 by mask $b0 = 000$. We will associate children with their binary recombination masks. We will also group the children into pairs of binary complements according to their recombination masks and the associated integers.

b7 = 111	b1 = 001	b2 = 010	b4 = 100
b0 = 000	b6 = 110	b5 = 101	b3 = 011

Append the ! symbol to the children that are local optima, and append a # symbol to children that can be improved by local search.

b7 = 111!	b1 = 001#	b2 = 010!	b4 = 100#
b0 = 000!	b6 = 110#	b5 = 101!	b3 = 011!

Assuming that we always recombine local optima using PX, this lattice is reachable by recombining P1 and P2, and by recombining C2 and C5. But it is not reachable from any of the other children because one or both of the potential parents is not a local optimum. This assumes, of course, that we only recombine local optima (because we improve all strings that are not locally optimal).

The Lattice Expansion Theorem tells us that it is useful to store information about all of the local optima that are discovered during search as well as information about recombining components.

There also may be value in storing children that are members of a lattice, but which are not locally optimal. We sampled 30 strings which appear as a child under Partition Crossover but which were not locally optimal from an adjacent NK landscape with N=40, k=3. All of the lattices were of size 16. On average a child that was not locally optimal appeared in 1275 different lattices. Thus all children found in lattices are strongly associated with other local optima and could play a critical role in discovering other lattices and local optima.

Is it worth searching the lattice of children for improving moves? Partition Crossover already returns the best of 2^q offspring and testing all of the offspring is not feasible for large q.

We sampled 120 lattices of size 4 (q=2) to collect data, where N=30. Note that lattices of size 4 are embedded in larger lattices. So the results also partially transfer to larger lattices. We only consider cases where there was an improving move. Note that the best child can also be improved in some cases.

How often is the worst child improved to yield the best solution? For random NK landscapes 34.8 percent of the time the worst child is improved to be better than the best child. For adjacent NK landscapes, 30.0 percent of the time the worst child is improved to be better than the best child. But in order to efficiently locate and find improving moves in $O(1)$ time we must use a *score* vector.

6.3 Updating the Score Vector

Modern iterated local search algorithms, as well as intelligent evolutionary algorithms, use a *score* vector to track improving moves for a 1-bit flip, or even a multi-flip neighborhood [13] [14] [7]. The Hoos and Stützle textbook [13] explains how score vectors are used to track improving moves for MAX-kSAT problems.

Assume the current candidate solution is x_c. Let x_p be a neighbor of x_c generated by flipping bit p. The score vector stores the following information:

$$score(c, p) = f(x_c) - f(x_p)$$

The index c is typically omitted because c is the current solution. Thus the score vector is of dimension n. If we are minimizing and $f(x) > 0$ for all x, a positive score indicates an improving move. Updating the *score* vector is dramatically better than enumerating the neighborhood.

Whitley et al. [21] present a proof of $O(1)$ average time complexity for updating the score vector for all k-bounded pseudo-Boolean functions. For k-bounded functions, a bit flip affects $O(1)$ other bits on average and thus in expectation an arbitrary bit flip changes only $O(1)$ entries in the *score* vector, but in the worst case it changes $O(n)$ entries in the *score* vector. An average $O(1)$ complexity is achieved by bounding the frequency of repeatedly flipping the same bit [21]; this will amortize the worst case behavior.

So how do we combine score vectors with Partition Crossover? For example, we might not want to evaluate all of the children in the lattice when looking for children that are **not** local optima, and thus could be improved by local search. Evaluating all of the children requires determining the fitness of 2^q individuals. These evaluations can be grouped so that q children can be evaluated with one call to the evaluation function. But this still results in $2^q/q$ calls to the evaluation function. For large q checking all of the children for improving moves is not feasible. So we would need to sample, but we want to sample in a clever and efficient manner.

Theorem (The Score Update Theorem): *It is possible visit $q - 2$ child solutions along a path between Parent 1 and Parent 2 by changing every recombining component exactly once. This requires less than n updates to the score vector and can be done in $O(n)$ time with at most 1 call to the complete evaluation function.*

PROOF: Assume the parents are P1 and P2. We move from P1 or P2 by changing one recombining component at a time. Using the auxiliary binary string b let $b = 0^q$ correspond to P2 and let $b = 1^q$ correspond to P1. We will flip each bit in b one time, thus moving from P2 to P1 (or vice versa). This also means that every bit in string x also only flips at most one time, since every variable is found in only one recombining component. The score vector is updated after each bit flip in string x. But not every bit in x will flip because there must exist "shared variables" which separate the recombining components. The cost of bit flips and score vector updates combined is therefore $O(n)$. Every subfunction is only evaluated once because each subfunction can only be associated with one recombining component (which is associated with only 1 bit in b). Thus, evaluating $q - 2$ children in this way is equivalent to at most one call to the evaluation function $f(x)$. □

In practice, we can construct a path of children between P1 and P2 that includes the best child. Or we can also construct a second path of children between P1 and P2 that includes the worst child. The Score Update Theorem allows us to explore a constant number of paths between P1 and P2 with only 1 call to the complete evaluation function per path.

Consider the following example: assume that q=6. We can evaluate the following set of children by changing 1 bit at a time on the auxiliary binary string b; this will yield less than n updates to the score vector.

000000 → 000001 → 100001 → 100101 → 110101 → 110111 → 111111

Note that this updates the score vector while only flipping those bits in x that occur in a recombining component because they are the only bits that change along this path. However, if an improving move is identified, it must be a bit that is not in a recombining component.

If there is any child along this path that can be improved by local search, the score vector will identify it, calculate the change in evaluation and the score vector can be automatically updated if that move is accepted. Note that there could be multiple children along such a path that might be improved by local search.

7 CONCLUSIONS

We have used Partition Crossover to discover lattices. However, the local optima which appear in lattices are defined with respect to a specific local search operator. The lattices and constraints are inherent to the function itself. Partition Crossover is only a vehicle for discovering the lattices.

We have shown that Partition Crossover operators can be used to find subsets of local optima that are related to each other by a simple linear equation. When Partition Crossover identifies q recombining components, it also identifies a lattice (i.e., a hypercube) of children. All of the children in that lattice have fitness evaluations that can be determined using the following linear equation:

$$f(x) = g(x') + c = \alpha_0 + \sum_{i=1}^{q} \alpha_i b_i$$

where $\alpha_0 = f(P2)$ and b_i is associated with a recombining component.

If the parents are local optima, and all of the children are local optima, then local optima can also be evaluated using the same linear equation.

We also prove that the evaluation of local optima which appear in lattices are highly constrained, and for every solution C_i in a lattice: $f(P1) + f(P2) = f(C_i) + f(\bar{C}_i)$.

The Hamming distance between children which appear in a lattice and the global optimum is also determined by a simple linear equation. For any string x that appears in a lattice, and global optimum x*:

$$HD(x, x^*) = \beta_0 + \sum_{i=1}^{q} \beta_i b_i. \quad (8)$$

where $\beta_0 = f(P2)$ and b_i is associated with a recombining component.

The Lattice Expansion Theorem shows how it is possible to find higher dimensional lattices given a specific sample of lower dimensional (sub)lattices of local optima.

We enumerated all of the local optima in adjacent and random NK landscapes for N=30 and N=40 and k=3. This empirical data suggests that lattices occur with extremely high frequency and that this frequency appears to increase with problem size for k-bounded functions. Our results also show dramatic differences in adjacent and random NK landscapes, which might call into question comparisons of algorithms when using adjacent NK landscapes (which are known to have PTIME solutions) as benchmark test functions.

This work greatly improves our understanding of the Big Valley distribution, and lays a theoretical foundation which suggests new ways in which we could build more intelligent heuristic search algorithms by exploiting known properties of k-bounded pseudo-Boolean functions, including MAX-kSAT [11].

REFERENCES

[1] K. D. Boese, A. B. Kahng, and S. Muddu. 1993. *On the big valley and adaptive multi-start for discrete global optimizations.* Technical Report. UCLA CS Department.

[2] Kenneth D. Boese, Andrew B. Kahng, and Sudhakar Muddu. 1994. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters* 16 (1994), 101–113.

[3] E. Boros and P.L. Hammer. 2002. Pseudo-Boolean Optimization. *Discrete applied mathematics* 123, 1 (2002), 155–225.

[4] W. Chen and D. Whitley. 2017. Decomposing SAT instances with pseudo backbones. In *17th European Conference on Evolutionary Computation Combinatorial Optimization (EVOCOP).* Springer, LNCS.

[5] W. Chen, D. Whitley, F. Chicano, and R. Tinós. 2018. Tunneling between plateaus: improving on a state-of-the-art MAXSAT solver using partition crossover. In *Genetic and Evolutionary Computation Conference (GECCO).* ACM, 921–928.

[6] F. Chicano, D. Whitley, G. Ochoa, and R. Tinós. 2017. Optimizing one million variable NK landscapes by hybridizing deterministic recombination and local search. In *Genetic and Evolutionary Computation Conference (GECCO).* ACM, 753–760.

[7] F. Chicano, D. Whitley, and A. Sutton. 2014. Efficient Identification of Improving Moves in a Ball for Pseudo-Boolean Problems. In *Genetic and Evolutionary Computation Conference (GECCO).* ACM, 437–444.

[8] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. 2001. *Introduction to Algorithms, 2nd Edition.* MIT Press, New York.

[9] A Howe W Chen D Hains, D Whitley. 2013. Hyperplane initialized local search for MAXSAT. In *Genetic and Evolutionary Computation Conference (GECCO).* ACM, 437–444.

[10] J P K Doye, M A Miller, and D J Wales. 1999. The double-funnel energy landscape of the 38-atom Lennard-Jones cluster. *Journal of Chemical Physics* 110, 14 (1999), 6896–6906. https://doi.org/10.1063/1.478595

[11] P. Dunton and D. Whitley. 2022. Reducing the cost of partition crossover on large MAXSAT problems: the PX-preprocessor. In *Genetic and Evolutionary Computation Conference (GECCO).* ACM, 694–702.

[12] Keld Helsgaun. 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *Eur. J. Oper. Res.* 126, 1 (2000), 106–130. https://doi.org/10.1016/S0377-2217(99)00284-2

[13] H.H. Hoos and Th. Stützle. 2004. *Stochastic Local Search: Foundations and Applications.* Morgan Kaufman.

[14] H. H. Hoos. 1999. On the run-time behaviour of stochastic local search algorithms for SAT.. In *Proc of AAAI.* 661–666.

[15] M. Mieskolainen and R. Orava. 2017. Observables of QCD diffraction. In *AIP Conf Proceedings*, Vol. 1819(1). AIP Publishing.

[16] M. E. J. Newman and Robin Engelhardt. 1998. Effects of neutral selection on the evolution of molecular species. *Proc. R. Soc. London B* 256 (1998), 1333–1338.

[17] Gabriela Ochoa and Sebastian Herrmann. 2018. Perturbation Strength and the Global Structure of QAP Fitness Landscapes. In *PPSN (2) (Lecture Notes in Computer Science, Vol. 11102).* Springer, 245–256.

[18] Renato Tinós, Keld Helsgaun, and Darrell Whitley. 2018. Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic. In *PPSN (1) (Lecture Notes in Computer Science, Vol. 11101).* Springer, 95–107.

[19] R. Tinos, D. Whitley, and F. Chicano. 2015. Partition Crossover for Pseudo-Boolean Optimization. In *Foundations of Genetic Algorithms.* ACM Press, 137–149.

[20] E.D. Weinberger. 1996. *NP-Completeness of Kauffman's N-k model: a tunably rugged energy landscape.* Technical Report 96-02-003. Santa Fe Institute, Santa Fe, NM.

[21] D. Whitley, A. Howe, and D. Hains. 2013. Greedy or Not? Best Improving versus First Improving Stochastic Local Search for MAXSAT. In *The National Conference on Artificial Intelligence (AAAI).* 940–946.

[22] D. Whitley and G. Ochoa. 2022. Local optima organize into lattices under recombination: an example using the traveling salesman problem. In *Genetic and Evolutionary Computation Conference (GECCO).* ACM, 757–765.