Visualisation of Cost Landscapes

Adam Prügel-Bennett
ISIS Research Group
Electronics and Computer Science
University of Southampton
Visualisation of Cost Landscapes

Adam Prügel-Bennett
ISIS Research Group
Electronics and Computer Science
University of Southampton

Work in collaboration with Jonathan Hallam and Will Benfold
1. Optimisation Problems

2. Cost Landscapes

3. Barrier Trees

4. Example Instances

5. Mapping Configurations

6. Modelling the Problem
Subtext of Talk

Complex Objects
Subtext of Talk

Complex Objects

Informal Pictures

Mathematical Analysis

Visualisation
Subtext of Talk

Mathematical Analysis

Informal Pictures

Complex Objects

Visualisation
Combinatorial Optimisation Problems

- Set view: $S$—‘the search space’
Combinatorial Optimisation Problems

- Set view: $S$—‘the search space’

- Consider discrete optimisation problems where $S$ is a finite countable set
Combinatorial Optimisation Problems

- Set view: $S$—‘the search space’

- Consider discrete optimisation problems where $S$ is a finite countable set

- Cost function: $f : S \mapsto \mathbb{R}$
Combinatorial Optimisation Problems

- Set view: \( S \)—‘the search space’

- Consider discrete optimisation problems where \( S \) is a finite countable set

- Cost function: \( f : S \mapsto \mathbb{R} \)

- Objective is to find \( x^* \in S : \forall y, f(x^*) \leq f(y) \)
Combinatorial Optimisation Problems

- Set view: $S$—‘the search space’

- Consider discrete optimisation problems where $S$ is a finite countable set

- Cost function: $f : S \rightarrow \mathbb{R}$

- Objective is to find $x^* \in S : \forall y, f(x^*) \leq f(y)$

- Usually satisfied with finding $x^* \in S : \text{for most } y, f(x^*) \leq f(y)$
Combinatorial Optimisation Problems

- Set view: $S$—‘the search space’

- Consider discrete optimisation problems where $S$ is a finite countable set

- Cost function: $f : S \mapsto \mathbb{R}$

- Objective is to find $x^* \in S : \forall y, f(x^*) \leq f(y)$

- Usually satisfied with finding $x^* \in S : \text{for most } y, f(x^*) \leq f(y)$

- Consider mainly degenerate problems $|f(S)| \ll |S|$
Example: Max-Sat

- Given $n$ Boolean variables $X_i \in \{T, F\}$

- $M$ disjunctive clauses, e.g.

  $$c_1 = x_1 \lor \neg x_2 \lor x_3$$
  $$c_2 = \neg x_2 \lor x_3 \lor x_5$$
  $$\vdots$$
  $$c_M = x_2 \lor \neg x_4 \lor \neg x_5$$

- Find an assignment, $X \in \{T, F\}^n$ which satisfies the most clauses
Example: Max-Sat

- Given $n$ Boolean variables $X_i \in \{T, F\}$

- $M$ disjunctive clauses, e.g.

  $$c_1 = x_1 \lor \neg x_2 \lor x_3$$
  $$c_2 = \neg x_2 \lor x_3 \lor x_5$$
  $$\vdots$$
  $$c_M = x_2 \lor \neg x_4 \lor \neg x_5$$

- Find an assignment, $X \in \{T, F\}^n$ which satisfies the most clauses
Example: Max-Sat

• Given \( n \) Boolean variables \( X_i \in \{T, F\} \)

• \( M \) disjunctive clauses, e.g.

\[
\begin{align*}
c_1 &= x_1 \lor \neg x_2 \lor x_3 \\
c_2 &= \neg x_2 \lor x_3 \lor x_5 \\
& \vdots \\
c_M &= x_2 \lor \neg x_4 \lor \neg x_5
\end{align*}
\]

• Find an assignment, \( X \in \{T, F\}^n \) which satisfies the most clauses
Max-Sat Problems

- Arises naturally in digital circuits
- Many planning problems reduce to Max-Sat
- Any non-deterministic Turing machine can be reduced to a SAT problem—Cook’s theorem
- Factorisation can be solved using a SAT solver
Max-Sat Problems

- Arises naturally in digital circuits
- Many planning problems reduce to Max-Sat
- Any non-deterministic Turing machine can be reduced to a SAT problem—Cook’s theorem
- Factorisation can be solved using a SAT solver
Max-Sat Problems

- Arises naturally in digital circuits
- Many planning problems reduce to Max-Sat
- Any non-deterministic Turing machine can be reduced to a SAT problem—Cook’s theorem
- Factorisation can be solved using a SAT solver
Max-Sat Problems

- Arises naturally in digital circuits
- Many planning problems reduce to Max-Sat
- Any non-deterministic Turing machine can be reduced to a SAT problem—Cook’s theorem
- Factorisation can be solved using a SAT solver
Max-Sat Problems

- Arises naturally in digital circuits
- Many planning problems reduce to Max-Sat
- Any non-deterministic Turing machine can be reduced to a SAT problem—Cook’s theorem
- Factorisation can be solved using a SAT solver
Machine Learning Example

- Perceptron is a classic learning machine for performing binary classification

- Given a data set $\mathcal{D} = \{ \mathbf{x}_i, c_i \}_{i=1}^P$
  - $\mathbf{x}_i$—$n$-dimensional input pattern
  - $c_i \in \{-1, 1\}$—class label

- Find a weight vector $\mathbf{w}$ such that

  $$c_i \mathbf{w}^T \mathbf{x}_i > 0$$

  for as many patterns as possible
Machine Learning Example

• Perceptron is a classic learning machine for performing binary classification

• Given a data set \( D = \{ \mathbf{x}_i, c_i \}_{i=1}^{P} \)
  * \( \mathbf{x}_i \)—\( n \)-dimensional input pattern
  * \( c_i \in \{-1, 1\} \)—class label

• Find a weight vector \( \mathbf{w} \) such that

\[
c_i \mathbf{w}^T \mathbf{x}_i > 0
\]

for as many patterns as possible
Machine Learning Example

- Perceptron is a classic learning machine for performing binary classification

- Given a data set $\mathcal{D} = \{x_i, c_i\}_{i=1}^P$
  
  - $x_i$—n-dimensional input pattern
  - $c_i \in \{-1, 1\}$—class label

- Find a weight vector $w$ such that

  $$c_i \mathbf{w}^T \mathbf{x}_i > 0$$

  for as many patterns as possible
Separating Plane

[Diagram of red and green points indicating a separating plane]
Separating Plane
Separating Plane
Ising Perceptron

- Weight vector is strongly constrained $w = \{-1, 1\}^n$
- Problem becomes NP-Hard (integer programming problem)
- Both Max-Sat and the Ising perceptron have a lot of structure not captured by the set view of optimisation problems
Ising Perceptron

- Weight vector is strongly constrained $\mathbf{w} = \{-1, 1\}^n$

- Problem becomes NP-Hard (integer programming problem)

- Both Max-Sat and the Ising perceptron have a lot of structure not captured by the set view of optimisation problems
Ising Perceptron

- Weight vector is strongly constrained \( \mathbf{w} = \{-1, 1\}^n \)
- Problem becomes NP-Hard (integer programming problem)
- Both Max-Sat and the Ising perceptron have a lot of structure not captured by the set view of optimisation problems
1. Optimisation Problems
2. Cost Landscapes
3. Barrier Trees
4. Example Instances
5. Mapping Configurations
6. Modelling the Problem
Neighbourhood

- We can introduce a topology to a search space by defining a neighbourhood function $\mathcal{N} : S \rightarrow 2^S$.

- Neighbourhoods are often connected with local search operators.

  $\mathcal{N}(x)$ is the set of configurations we can move to from $x$.

- Key Heuristic: Neighbouring configurations are more likely to have a similar cost than random configurations.
Neighbourhood

• We can introduce a topology to a search space by defining a neighbourhood function $\mathcal{N} : S \mapsto 2^S$

• Neighbourhoods are often connected with local search operators

  $\mathcal{N}(x)$ is the set of configurations we can move to from $x$

• Key Heuristic: Neighbouring configurations are more likely to have a similar cost than random configurations
Neighbourhood

• We can introduce a topology to a search space by defining a neighbourhood function \( \mathcal{N} : \mathcal{S} \mapsto 2^\mathcal{S} \)

• Neighbourhoods are often connected with local search operators

\[ \mathcal{N}(x) \text{ is the set of configurations we can move to from } x \]

• Key Heuristic: Neighbouring configurations are more likely to have a similar cost than random configurations
Neighbourhood

- We can introduce a topology to a search space by defining a neighbourhood function $\mathcal{N} : S \mapsto 2^S$

- Neighbourhoods are often connected with local search operators

  $\mathcal{N}(x)$ is the set of configurations we can move to from $x$

- Key Heuristic: Neighbouring configurations are more likely to have a similar cost than random configurations
A natural neighbourhood for binary strings is the Hamming neighbourhood

Configurations differing at a single site are neighbours

\[ X = (T, T, F, T, F, T, T) \quad w = (+1, -1, -1, +1, +1, +1) \]
\[ X = (T, F, F, T, F, T, T) \quad w = (+1, -1, -1, +1, -1, +1) \]

Changing one variable in Max-Sat shouldn’t affect too many clauses

Changing a weight in the Ising perceptron causes a minimum shift in the separating plane
Hamming Neighbourhood

• A natural neighbourhood for binary strings is the Hamming neighbourhood

• Configurations differing at a single site are neighbours

\[ X = (T, T, F, T, F, T, T) \quad w = (+1, -1, -1, +1, +1, +1) \]

\[ X = (T, F, F, T, F, T, T) \quad w = (+1, -1, -1, +1, -1, +1) \]

• Changing one variable in Max-Sat shouldn’t affect too many clauses

• Changing a weight in the Ising perceptron causes a minimum shift in the separating plane
Hamming Neighbourhood

- A natural neighbourhood for binary strings is the Hamming neighbourhood

- Configurations differing at a single site are neighbours

\[ X = (T, T, F, T, F, T, T) \quad w = (+1, -1, -1, +1, +1, +1) \]
\[ X = (T, F, F, T, F, T, T) \quad w = (+1, -1, -1, +1, -1, +1) \]

- Changing one variable in Max-Sat shouldn’t affect too many clauses

- Changing a weight in the Ising perceptron causes a minimum shift in the separating plane
Hamming Neighbourhood

- A natural neighbourhood for binary strings is the Hamming neighbourhood

- Configurations differing at a single site are neighbours

\[
X = (T, T, F, T, F, T, T) \quad w = (+1, -1, -1, +1, +1, +1) \\
X = (T, F, F, T, F, T, T) \quad w = (+1, -1, -1, +1, -1, +1)
\]

- Changing one variable in Max-Sat shouldn’t affect too many clauses

- Changing a weight in the Ising perceptron causes a minimum shift in the separating plane
Hypercube Topology

- The Hamming neighbourhood induces a hypercube topology on the search space of binary strings

3D Binary String Search Space
Other Neighbourhoods

- For binary string problems the Hamming neighbourhood is so natural it is difficult to image anything else
Other Neighbourhoods

• For binary string problems the Hamming neighbourhood is so natural it is difficult to image anything else

• But what is the neighbourhood for TSP?
Other Neighbourhoods

- For binary string problems the Hamming neighbourhood is so natural it is difficult to image anything else
- But what is the neighbourhood for TSP?
Other Neighbourhoods

- For binary string problems the Hamming neighbourhood is so natural it is difficult to image anything else
- But what is the neighbourhood for TSP?

2-Opt neighbours
Cost Landscape

- Cost function + Neighbourhood = Landscape
Cost Landscape

- Cost function + Neighbourhood = Landscape
Cost Landscape

- Cost function + Neighbourhood = Landscape
Cost Landscape

- Cost function + Neighbourhood = Landscape

Unfortunately difficult to visualise in high dimensions
Costs in the Ising Perceptron

Each input divides the search space in two
Costs in the Ising Perceptron

Each input divides the search space in two
Correctly Classified Region

The set of inputs defines a cone on the hypersphere which correctly classifies all inputs
Correctly Classified Region

The set of inputs defines a cone on the hypersphere which correctly classifies all inputs.
Increasing the Problem Difficulty

- The problem difficulty is increased as we increase \( \alpha = \frac{P}{n} \)
The problem difficulty is increased as we increase $\alpha = P/n$.
Increasing the Problem Difficulty

• The problem difficulty is increased as we increase $\alpha = P/n$

Typically means that the exceptions become exponentially rare with $n$
Schematic representation of low $\alpha$

Note that the real search space is high dimensional
Schematic representation at higher $\alpha$
Schematic representation at higher $\alpha$

Solutions are typically disconnected
Replica Symmetry Breaking

- The problem becomes hard just as the solutions become disconnected

- The solutions can be a long way apart

- The solution space “shatters” into many local optima rather like a piece of glass
  - shattering is random
  - there are (exponentially) many local optima
  - the sizes of the local optima differ dramatically

- In calculations this shattering causes “replica-symmetry breaking”
Replica Symmetry Breaking

• The problem becomes hard just as the solutions become disconnected

• The solutions can be a long way apart

• The solution space “shatters” into many local optima rather like a piece of glass
  ★ shattering is random
  ★ there are (exponentially) many local optima
  ★ the sizes of the local optima differ dramatically

• In calculations this shattering causes “replica-symmetry breaking”
Replica Symmetry Breaking

- The problem becomes hard just as the solutions become disconnected.
- The solutions can be a long way apart.
- The solution space “shatters” into many local optima rather like a piece of glass.
  - Shattering is random.
  - There are (exponentially) many local optima.
  - The sizes of the local optima differ dramatically.
- In calculations this shattering causes “replica-symmetry breaking.”
Replica Symmetry Breaking

- The problem becomes hard just as the solutions become disconnected.

- The solutions can be a long way apart.

- The solution space “shatters” into many local optima rather like a piece of glass:
  - Shattering is random.
  - There are (exponentially) many local optima.
  - The sizes of the local optima differ dramatically.

- In calculations this shattering causes “replica-symmetry breaking.”
Easy Phase

\[ \alpha < \alpha_d \]

‘Gradient’ leads to global optima
Hard Phase

\[ \alpha > \alpha_d \]

Ambiguous ‘gradients’
Is this an Accurate Picture?

- Interested in high dimensional discrete problems
- Usually there are exponential number of local optima
- Phase transition can be more complicated (e.g. number partitioning problem)
Is this an Accurate Picture?

- Interested in high dimensional discrete problems
- Usually there are exponential number of local optima
- Phase transition can be more complicated (e.g. number partitioning problem)
• Interested in high dimensional discrete problems

• Usually there are exponential number of local optima

• Phase transition can be more complicated (e.g. number partitioning problem)
Is this an Accurate Picture?

• Interested in high dimensional discrete problems

• Usually there are exponential number of local optima

• Phase transition can be more complicated (e.g. number partitioning problem)
1. Optimisation Problems
2. Cost Landscapes
3. Barrier Trees
4. Example Instances
5. Mapping Configurations
6. Modelling the Problem
Barrier Trees

• To help visualise the search space we can build a ‘Barrier Tree’

• Leaves of the tree are local minima

• Merging vertices correspond to lowest cost saddle-points

• Height of vertices indicate fitness
Barrier Trees

- To help visualise the search space we can build a ‘Barrier Tree’
- Leaves of the tree are local minima
- Merging vertices correspond to lowest cost saddle-points
- Height of vertices indicate fitness
Barrier Trees

- To help visualise the search space we can build a ‘Barrier Tree’
- Leaves of the tree are local minima
- Merging vertices correspond to lowest cost saddle-points
- Height of vertices indicate fitness
Barrier Trees

- To help visualise the search space we can build a ‘Barrier Tree’
- Leaves of the tree are local minima
- Merging vertices correspond to lowest cost saddle-points
- Height of vertices indicate fitness
Example
Discrete Search Spaces

• To formalise these notion we define a path

\[ \pi(x, y) = (x_i \mid x_1 = x \land x_n = y \land x_{i+1} \in N(x_i)) \]

• A sequence of Neighbouring configurations from \( x \) to \( y \)

• Assume that the Neighbourhood relation is symmetric
Discrete Search Spaces

- To formalise these notion we define a path
  \[ \pi(x, y) = (x_i | x_1 = x \land x_n = y \land x_{i+1} \in \mathcal{N}(x_i)) \]
  A sequence of Neighbouring configurations from \( x \) to \( y \)

- Assume that the Neighbourhood relation is symmetric
Level Connected Sets

- We define the level connectedness by the equivalence relation

\[ \mathcal{LC} = \{(x, y) | \exists \pi(x, y) \land \forall x_i \in \pi(x, y), f(x_i) = f(x)\} \]
Limitation of Level Connectedness

- There are often a very large number of level-connected sets
- Consider the ‘ones-max’ problem where

\[ f(x) = \text{number of ones in } x \]

- Every configuration is a level-connected set with a Hamming neighbourhood
Limitation of Level Connectedness

- There are often a very large number of level-connected sets

- Consider the ‘ones-max’ problem where

  \[ f(x) = \text{number of ones in } x \]

- Every configuration is a level-connected set with a Hamming neighbourhood
Limitation of Level Connectedness

- There are often a very large number of level-connected sets.
- Consider the ‘ones-max’ problem where $f(x) =$ number of ones in $x$.
- Every configuration is a level-connected set with a Hamming neighbourhood.
Level Accessible

• We define accessibility as the relation

\[ A = \{ (x, y) | \exists \pi(x, y) \land \forall x_i \in \pi(x, y), f(x_i) \leq f(x) \} \]

i.e. there exists a path with no configuration exceeding the cost of the initial configuration

• Level accessibility is the equivalence relation that

\[ LA = \{ (x, y) | (x, y) \in A \land f(x) = f(y) \} \]

• Level accessibility induces a partitioning of the search space into level-accessible sets

\[ S = \bigcup_{i=1}^{N+1} V_i \]

• In ones-max there are \( N + 1 \) level accessible sets
Level Accessible

• We define accessibility as the relation

\[ A = \{(x, y)|\exists \pi(x, y) \land \forall x_i \in \pi(x, y), f(x_i) \leq f(x)\}\]

i.e. there exists a path with no configuration exceeding the cost of the initial configuration

• Level accessibility is the equivalence relation that

\[ \mathcal{L}A = \{(x, y)|(x, y) \in A \land f(x) = f(y)\}\]

• Level accessibility induces a partitioning of the search space into level-accessible sets

\[ S = \bigcup_{i=1}^{N+1} V_i \]

• In ones-max there are \(N + 1\) level accessible sets
We define accessibility as the relation

\[ A = \{ (x, y) | \exists \pi(x, y) \land \forall x_i \in \pi(x, y), f(x_i) \leq f(x) \} \]

i.e. there exists a path with no configuration exceeding the cost of the initial configuration.

Level accessibility is the equivalence relation that

\[ LA = \{ (x, y) | (x, y) \in A \land f(x) = f(y) \} \]

Level accessibility induces a partitioning of the search space into level-accessible sets \( S = \bigcup_{i=1}^{N} V_i \).

In ones-max there are \( N + 1 \) level accessible sets.
Level Accessible

- We define **accessibility** as the relation

  \[ A = \{(x, y) | \exists \pi(x, y) \land \forall x_i \in \pi(x, y), f(x_i) \leq f(x)\} \]

  i.e. there exists a path with no configuration exceeding the cost of the initial configuration

- **Level accessibility** is the equivalence relation that

  \[ \mathcal{L}A = \{(x, y) | (x, y) \in A \land f(x) = f(y)\} \]

- Level accessibility induces a partitioning of the search space into level-accessible sets \( S = \bigcup_{i=1}^{N} V_i \)

- In ones-max there are \( N + 1 \) level accessible sets
Level Accessible Sets
Barrier Trees

- We can define accessibility between level-accessible sets

\[ \mathcal{A}_{LAS} = \{ (\mathcal{V}_i, \mathcal{V}_j) | \exists x \in \mathcal{V}_i, \exists y \in \mathcal{V}_j, (x, y) \in \mathcal{A} \} \]
Barrier Trees

- We can define accessibility between level-accessible sets

\[ A_{LAS} = \{(V_i, V_j) \mid \exists x \in V_i, \exists y \in V_j, (x, y) \in A\} \]

- Accessibility between level-accessible sets defines a partial ordering
Barrier Trees

- We can define accessibility between level-accessible sets

\[ \mathcal{A}_{LAS} = \{(V_i, V_j) \mid \exists x \in V_i, \exists y \in V_j, (x, y) \in A\} \]

- Accessibility between level-accessible sets defines a partial ordering

- The Barrier tree is the diagram of the partial ordering
Barrier Trees

• We can define accessibility between level-accessible sets

\[ \mathcal{A}_{LAS} = \{(V_i, V_j) \mid \exists x \in V_i, \exists y \in V_j, (x, y) \in A\} \]

• Accessibility between level-accessible sets defines a partial ordering

• The Barrier tree is the diagram of the partial ordering i.e. graph of direct predecessor
Barrier Trees

• We can define accessibility between level-accessible sets

\[ A_{LAS} = \{(V_i, V_j) \mid \exists x \in V_i, \exists y \in V_j, (x, y) \in A\}\]

• Accessibility between level-accessible sets defines a partial ordering

• The Barrier tree is the diagram of the partial ordering i.e. graph of direct predecessor
Barrier Trees

- We can define accessibility between level-accessible sets

\[ A_{LAS} = \{ (V_i, V_j) | \exists x \in V_i, \exists y \in V_j, (x, y) \in A \} \]

- Accessibility between level-accessible sets defines a partial ordering

- The Barrier tree is the diagram of the partial ordering i.e. graph of direct predecessor
Barrier Trees

• We can define accessibility between level-accessible sets

\[ \mathcal{A}_{LAS} = \{ (V_i, V_j) | \exists x \in V_i, \exists y \in V_j, (x, y) \in A \} \]

• Accessibility between level-accessible sets defines a partial ordering

• The Barrier tree is the diagram of the partial ordering i.e. graph of direct predecessor
Computing Barrier Trees

- Finding the level-accessible sets and Barrier trees can be computed efficiently, $O(|S| \times |N|)$, using a flooding algorithm

- For larger problems we can compute the low-cost part of the Barrier tree using a modified branch and bound algorithm

- The statistical properties of the rest of the tree can be estimated using sampling techniques
Computing Barrier Trees

- Finding the level-accessible sets and Barrier trees can be computed efficiently, $O(|S| \times |N|)$, using a flooding algorithm.

- For larger problems we can compute the low-cost part of the Barrier tree using a modified branch and bound algorithm.

- The statistical properties of the rest of the tree can be estimated using sampling techniques.
Computing Barrier Trees

- Finding the level-accessible sets and Barrier trees can be computed efficiently, \( O(|S| \times |N|) \), using a flooding algorithm.

- For larger problems we can compute the low-cost part of the Barrier tree using a modified branch and bound algorithm.

- The statistical properties of the rest of the tree can be estimated using sampling techniques.
Outline

1. Optimisation Problems
2. Cost Landscapes
3. Barrier Trees
4. Example Instances
5. Mapping Configurations
6. Modelling the Problem
\textbf{MAX-3-SAT } $\alpha = \frac{M}{N} = 5 \ N = 10$
MAX-3-SAT $\alpha = \frac{M}{N} = 5 \ N = 20$
MAX-3-SAT $\alpha = \frac{M}{N} = 5$ $N = 30$
MAX-3-SAT $\alpha = \frac{M}{N} = 5 \ N = 40$
MAX-3-SAT $N = 40 \ \alpha = M/N = 4$
MAX-3-SAT $N = 40 \ \alpha = M/N = 6$
MAX-3-SAT  \( N = 40 \)  \( \alpha = M/N = 8 \)
MAX-3-SAT  \( N = 40 \)  \( \alpha = M/N = 10 \)
Phase Transition in Number Partitioning

- Are phase transitions evident in barrier trees?
- Studied by Stadler, Hordijk and Fontanari
- Very little evidence in shape
- Phase transition in difficulty $B/\delta$
Phase Transition in Number Partitioning

- Are phase transitions evident in barrier trees?

- Studied by Stadler, Hordijk and Fontanari

- Very little evidence in shape

- Phase transition in difficulty $B/\delta$
Phase Transition in Number Partitioning

- Are phase transitions evident in barrier trees?
- Studied by Stadler, Hordijk and Fontanari
- Very little evidence in shape
- Phase transition in difficulty $B/\delta$
Phase Transition in Number Partitioning

- Are phase transitions evident in barrier trees?
- Studied by Stadler, Hordijk and Fontanari
- Very little evidence in shape
- Phase transition in difficulty $B/\delta$
1. Optimisation Problems
2. Cost Landscapes
3. Barrier Trees
4. Example Instances
5. Mapping Configurations
6. Modelling the Problem
Mappings Configurations

The diagram depicts a cubic structure with labeled vertices. The vertices are numbered 1, 2, 3, 4, and are connected by lines to form a 3D cube. The labels indicate the mapping configurations as follows:

- Vertex 1: 3
- Vertex 2: 2
- Vertex 3: 3
- Vertex 4: 4

These labels suggest a specific mapping scheme across the vertices of the cube.
Mappings Configurations

![Diagram of mappings configurations]
Mappings Configurations
Mapping Configurations

- Every configuration is mapped to a node of the Barrier tree

- We can study statistical properties of the tree vertices
  - number of configurations in a state
  - Stringiness of local minima
  - Distance to global solution
Mapping Configurations

- Every configuration is mapped to a node of the Barrier tree

- We can study statistical properties of the tree vertices
  - number of configurations in a state
  - Stringiness of local minima
  - Distance to global solution
Max-3-Sat $N = 40 \ \alpha = M/N = 8$
## Statistics

<table>
<thead>
<tr>
<th>Cost</th>
<th># of configs</th>
<th># of minima</th>
<th>Mean basin size</th>
<th># minima depth &gt; 1</th>
<th>% configs local minima</th>
<th># of saddle points</th>
<th>% descents runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>210560</td>
<td>25</td>
<td>2.44</td>
<td>0</td>
<td>0.029</td>
<td>1</td>
<td>2.06</td>
</tr>
<tr>
<td>15</td>
<td>46518</td>
<td>37</td>
<td>4.29</td>
<td>0</td>
<td>0.34</td>
<td>1</td>
<td>9.09</td>
</tr>
<tr>
<td>14</td>
<td>8336</td>
<td>28</td>
<td>3.93</td>
<td>2</td>
<td>1.3</td>
<td>1</td>
<td>18.04</td>
</tr>
<tr>
<td>13</td>
<td>1041</td>
<td>12</td>
<td>10.92</td>
<td>4</td>
<td>12.6</td>
<td>2</td>
<td>18.45</td>
</tr>
<tr>
<td>12</td>
<td>47</td>
<td>11</td>
<td>4.27</td>
<td>0</td>
<td>100.0</td>
<td>0</td>
<td>50.71</td>
</tr>
</tbody>
</table>
Simulated Annealing 40 Variables

t=0

Diagram of a simulated annealing process with 40 variables.
Simulated Annealing 40 Variables

$t=16$
Simulated Annealing 40 Variables

\[ t=17 \]
Simulated Annealing 40 Variables

t=19
Simulated Annealing 40 Variables

$t=20$
Simulated Annealing 40 Variables

t=22
Simulated Annealing 40 Variables

$C$

5

4

3

2

1

$t=23$
Simulated Annealing 40 Variables
Simulated Annealing 40 Variables

t=30
Simulated Annealing 40 Variables

t=40
Simulated Annealing 40 Variables

\[ t=42 \]
Genetic Algorithms 30 Variables

\[ t = 0 \]
Genetic Algorithms 30 Variables

\[ t=16 \]
Genetic Algorithms 30 Variables

t=341
Genetic Algorithms 30 Variables

t=1805
Genetic Algorithms 30 Variables

t=2514
1. Optimisation Problems
2. Cost Landscapes
3. Barrier Trees
4. Example Instances
5. Mapping Configurations
6. Modelling the Problem
Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!
- Model has same structure of local minima and barriers as true problem
Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!
- Model has same structure of local minima and barriers as true problem
Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!
- Model has same structure of local minima and barriers as true problem
Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!
- Model has same structure of local minima and barriers as true problem
Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!
- Model has same structure of local minima and barriers as true problem
Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!

- Model has same structure of local minima and barriers as true problem
Schematic of Model Problem
Descent: Max-3-Sat $N = 20$, $\alpha = 6$
Comparison with True Problem

- Same qualitative behaviour
- Dynamics is slower—dynamics within state is not captured
- Model problems are systematically easier
- Configurations in state are not searched equally often
Comparison with True Problem

- Same qualitative behaviour
- Dynamics is slower—dynamics within state is not captured
- Model problems are systematically easier
- Configurations in state are not searched equally often
Comparison with True Problem

- Same qualitative behaviour
- Dynamics is slower—dynamics within state is not captured
- Model problems are systematically easier
- Configurations in state are not searched equally often
Comparison with True Problem

- Same qualitative behaviour
- Dynamics is slower—dynamics within state is not captured
- Model problems are systematically easier
- Configurations in state are not searched equally often
Comparison with True Problem

- Same qualitative behaviour
- Dynamics is slower—dynamics within state is not captured
- Model problems are systematically easier
- Configurations in state are not searched equally often
Frequency of Visiting Configurations

Configuration in a saddle-point level-connected set
Markov Models

• Many Heuristic search algorithms can be modelled as Markov processes

\[ p(t) = W(t)p(t-1) \]

• \( W(t) \) transition matrix

• \( W(t) \) depends on the connectivity matrix \( M \) and the heuristic

• For simulated annealing \( W(t) \) depends on the annealing temperature

• Because we have a small number of states this is tractable for moderate sized problems
Markov Models

- Many Heuristic search algorithms can be modelled as Markov processes
  \[ p(t) = W(t)p(t-1) \]

- \( W(t) \) transition matrix

- \( W(t) \) depends on the connectivity matrix \( M \) and the heuristic

- For simulated annealing \( W(t) \) depends on the annealing temperature

- Because we have a small number of states this is tractable for moderate sized problems
Markov Models

- Many Heuristic search algorithms can be modelled as Markov processes
  \[ p(t) = W(t)p(t - 1) \]

- \(W(t)\) transition matrix

- \(W(t)\) depends on the connectivity matrix \(M\) and the heuristic

- For simulated annealing \(W(t)\) depends on the annealing temperature

- Because we have a small number of states this is tractable for moderate sized problems
Markov Models

- Many Heuristic search algorithms can be modelled as Markov processes
  \[ p(t) = W(t)p(t - 1) \]

- \( W(t) \) transition matrix

- \( W(t) \) depends on the connectivity matrix \( M \) and the heuristic

- For simulated annealing \( W(t) \) depends on the annealing temperature

- Because we have a small number of states this is tractable for moderate sized problems
Markov Models

• Many Heuristic search algorithms can be modelled as Markov processes

\[ p(t) = W(t)p(t - 1) \]

• \( W(t) \) transition matrix

• \( W(t) \) depends on the connectivity matrix \( M \) and the heuristic

• For simulated annealing \( W(t) \) depends on the annealing temperature

• Because we have a small number of states this is tractable for moderate sized problems
Optimal Schedules

- We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems.

- Schedule depends on optimisation criteria:
  - Where-you-are—optimise the final result
  - Best-so-far—optimise the best result obtained during run

- Can optimise for an ensemble of problems

- Can optimise other schedules such as the mutation rate for a descent algorithm
Optimal Schedules

- We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems.

- Schedule depends on optimisation criteria
  * Where-you-are—optimise the final result
  * Best-so-far—optimise the best result obtained during run

- Can optimise for an ensemble of problems

- Can optimise other schedules such as the mutation rate for a descent algorithm
Optimal Schedules

- We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems

- Schedule depends on optimisation criteria
  - Where-you-are—optimise the final result
  - Best-so-far—optimise the best result obtained during run

- Can optimise for an ensemble of problems

- Can optimise other schedules such as the mutation rate for a descent algorithm
Optimal Schedules

• We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems

• Schedule depends on optimisation criteria
  * Where-you-are—optimise the final result
  * Best-so-far—optimise the best result obtained during run

• Can optimise for an ensemble of problems

• Can optimise other schedules such as the mutation rate for a descent algorithm
Optimal Schedules

- We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems.

- Schedule depends on optimisation criteria
  - Where-you-are—optimise the final result
  - Best-so-far—optimise the best result obtained during run

- Can optimise for an ensemble of problems

- Can optimise other schedules such as the mutation rate for a descent algorithm
Optimal Schedules

• We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems

• Schedule depends on optimisation criteria
  * Where-you-are—optimise the final result
  * Best-so-far—optimise the best result obtained during run

• Can optimise for an ensemble of problems

• Can optimise other schedules such as the mutation rate for a descent algorithm
Optimal Schedules

• We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems

• Schedule depends on optimisation criteria
  ✤ Where-you-are—optimise the final result
  ✤ Best-so-far—optimise the best result obtained during run

• Can optimise for an ensemble of problems

• Can optimise other schedules such as the mutation rate for a descent algorithm
Max-Sat: Annealing Schedules with Where-You-Are

Adam Prügel-Bennett (ISIS ECS)

June 25, 2003
Max-Sat: Annealing Schedules with Best-So-Far

![Graph showing temperature over time for different temperatures T=100, T=200, T=300, T=400.](image)
Max-Sat: Mutation Schedule with Where-You-Are

![Graph showing mutation rate over time for different problem types.](image)

16-bit hurdle problem
20-variable Max-SAT problem
Conclusions

• Barrier trees provide a powerful tool for visualising cost landscapes

• By mapping configurations to nodes on the Barrier tree we can do much more
  ⋆ measure more statistical properties
  ⋆ visualise heuristic algorithms
  ⋆ construct model problems

• Model problems open new possibilities for studying heuristic search
Conclusions

• Barrier trees provide a powerful tool for visualising cost landscapes

• By mapping configurations to nodes on the Barrier tree we can do much more
  ✗ measure more statistical properties
  ✗ visualise heuristic algorithms
  ✗ construct model problems

• Model problems open new possibilities for studying heuristic search
Conclusions

• Barrier trees provide a powerful tool for visualising cost landscapes

• By mapping configurations to nodes on the Barrier tree we can do much more
  \* measure more statistical properties
  \* visualise heuristic algorithms
  \* construct model problems

• Model problems open new possibilities for studying heuristic search
Conclusions

• Barrier trees provide a powerful tool for visualising cost landscapes

• By mapping configurations to nodes on the Barrier tree we can do much more
  * measure more statistical properties
  * visualise heuristic algorithms
  * construct model problems

• Model problems open new possibilities for studying heuristic search
Conclusions

• Barrier trees provide a powerful tool for visualising cost landscapes

• By mapping configurations to nodes on the Barrier tree we can do much more
  ★ measure more statistical properties
  ★ visualise heuristic algorithms
  ★ construct model problems

• Model problems open new possibilities for studying heuristic search
Conclusions

- Barrier trees provide a powerful tool for visualising cost landscapes

- By mapping configurations to nodes on the Barrier tree we can do much more
  - measure more statistical properties
  - visualise heuristic algorithms
  - construct model problems

- Model problems open new possibilities for studying heuristic search
Any Questions?