

Model Building Algorithms & Routing

Tom Hinton

UOB

2008-03-11

Standard issue introductory slide

What is this presentation for?

- 1 Introduce optimisation by building probabilistic models
The biologically inspired computation part
- 2 Describe a simple model for routing problems and some results about same
- 3 Solicit suggestions about more appropriate classes of model

Standard issue introductory slide

What is this presentation for?

- 1 Introduce optimisation by building probabilistic models
The biologically inspired computation part
- 2 Describe a simple model for routing problems and some results about same
- 3 Solicit suggestions about more appropriate classes of model

Standard issue introductory slide

What is this presentation for?

- 1 Introduce optimisation by building probabilistic models
The biologically inspired computation part
- 2 Describe a simple model for routing problems and some results about same
- 3 Solicit suggestions about more appropriate classes of model

- **Optimisation**

Given $f : \mathcal{X} \rightarrow \mathcal{R}$ and an ordering on \mathcal{R} , find an $x \in \mathcal{X}$ which minimises f . The value $f(x)$ is known as OPT.

- **α -approximation**

Given f etc, find an $x \in \mathcal{X}$ such that $f(x) \leq \alpha \text{OPT}$

- **Differential α -approximation**

Given f etc, find an $x \in \mathcal{X}$ such that $|\{y \in \mathcal{X} : f(x) > y\}| \geq \alpha |\mathcal{X}|$
(i.e. find a solution better than α th of the space).

- **Hand-waving Optimisation**

Given f etc, wave your hands and find a solution that's OK.

- **Optimisation**

Given $f : \mathcal{X} \rightarrow \mathcal{R}$ and an ordering on \mathcal{R} , find an $x \in \mathcal{X}$ which minimises f . The value $f(x)$ is known as OPT.

- **α -approximation**

Given f etc, find an $x \in \mathcal{X}$ such that $f(x) \leq \alpha \text{OPT}$

- **Differential α -approximation**

Given f etc, find an $x \in \mathcal{X}$ such that $|\{y \in \mathcal{X} : f(x) > y\}| \geq \alpha |\mathcal{X}|$ (i.e. find a solution better than α th of the space).

- **Hand-waving Optimisation**

Given f etc, wave your hands and find a solution that's OK.

- **Optimisation**

Given $f : \mathcal{X} \rightarrow \mathcal{R}$ and an ordering on \mathcal{R} , find an $x \in \mathcal{X}$ which minimises f . The value $f(x)$ is known as OPT.

- **α -approximation**

Given f etc, find an $x \in \mathcal{X}$ such that $f(x) \leq \alpha \text{OPT}$

- **Differential α -approximation**

Given f etc, find an $x \in \mathcal{X}$ such that $|\{y \in \mathcal{X} : f(x) > y\}| \geq \alpha |\mathcal{X}|$ (i.e. find a solution better than α th of the space).

- **Hand-waving Optimisation**

Given f etc, wave your hands and find a solution that's OK.

- **Optimisation**

Given $f : \mathcal{X} \rightarrow \mathcal{R}$ and an ordering on \mathcal{R} , find an $x \in \mathcal{X}$ which minimises f . The value $f(x)$ is known as OPT.

- **α -approximation**

Given f etc, find an $x \in \mathcal{X}$ such that $f(x) \leq \alpha \text{OPT}$

- **Differential α -approximation**

Given f etc, find an $x \in \mathcal{X}$ such that $|\{y \in \mathcal{X} : f(x) > y\}| \geq \alpha |\mathcal{X}|$ (i.e. find a solution better than α th of the space).

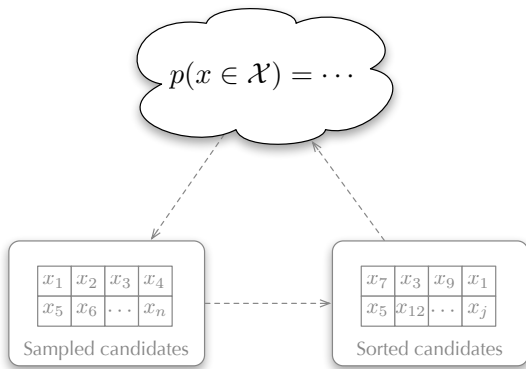
- **Hand-waving Optimisation**

Given f etc, wave your hands and find a solution that's OK.

Optimisation by iterative model building

AKA iterated density distribution, cross-entropy method, estimation of distribution, ...

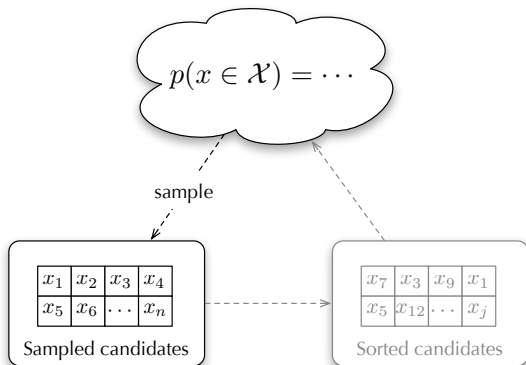
- Has a lot of names because it's been invented by a lot of people.
- Devil is as usual in the details (choice of model, here)



Optimisation by iterative model building

AKA iterated density distribution, cross-entropy method, estimation of distribution, ...

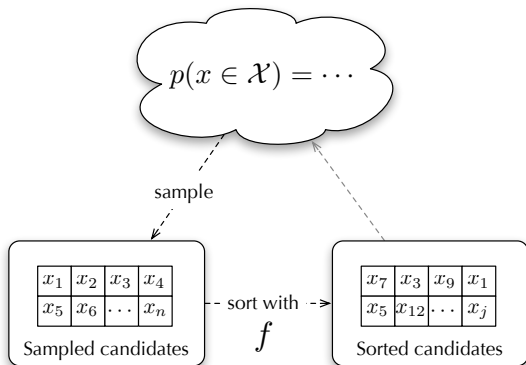
- Has a lot of names because it's been invented by a lot of people.
- Devil is as usual in the details (choice of model, here)



Optimisation by iterative model building

AKA iterated density distribution, cross-entropy method, estimation of distribution, ...

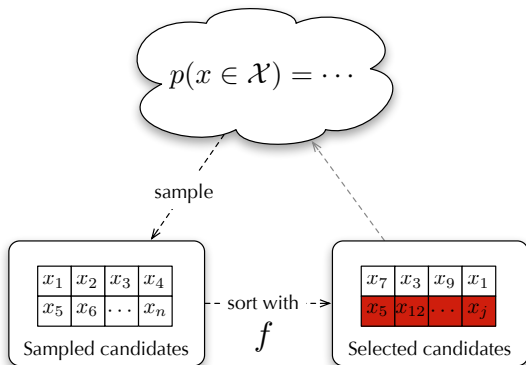
- Has a lot of names because it's been invented by a lot of people.
- Devil is as usual in the details (choice of model, here)



Optimisation by iterative model building

AKA iterated density distribution, cross-entropy method, estimation of distribution, ...

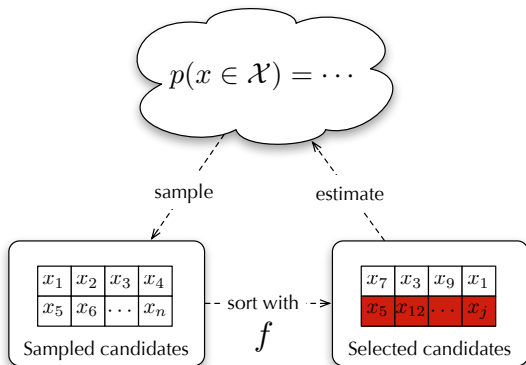
- Has a lot of names because it's been invented by a lot of people.
- Devil is as usual in the details (choice of model, here)



Optimisation by iterative model building

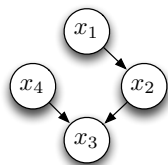
AKA iterated density distribution, cross-entropy method, estimation of distribution, ...

- Has a lot of names because it's been invented by a lot of people.
- Devil is as usual in the details (choice of model, here)



Models for unconstrained variable assignment problems

- Almost all model-building optimisation algorithms work has been done using problems where $\mathcal{X} = A^n$ for some alphabet A
 - Valid choices for variable i don't depend on choice for variable j
(Not saying the problem is linear, just that all strings in A^n are valid)
- State of the art techniques use k -degree acyclic graphs and statistical tests for conditional dependence to model f .

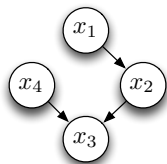


$$p(\vec{x}) = p(x_3|x_2, x_4) \cdot p(x_2|x_1) \cdot p(x_1) \cdot p(x_4)$$

- These can provably solve k -decomposable problems over n variables to optimality in $O(n^2)$ time and polynomial spaces.

Models for unconstrained variable assignment problems

- Almost all model-building optimisation algorithms work has been done using problems where $\mathcal{X} = A^n$ for some alphabet A
 - Valid choices for variable i don't depend on choice for variable j
(Not saying the problem is linear, just that all strings in A^n are valid)
- State of the art techniques use k -degree acyclic graphs and statistical tests for conditional dependence to model f .

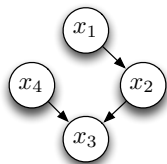


$$p(\vec{x}) = p(x_3|x_2, x_4) \cdot p(x_2|x_1) \cdot p(x_1) \cdot p(x_4)$$

- These can provably solve k -decomposable problems over n variables to optimality in $O(n^2)$ time and polynomial spaces.

Models for unconstrained variable assignment problems

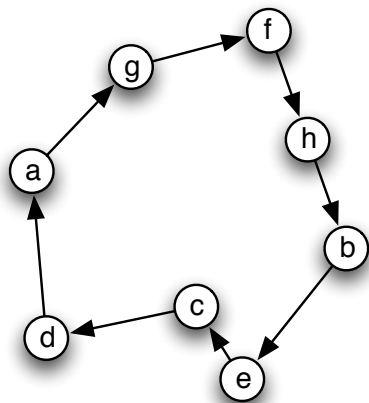
- Almost all model-building optimisation algorithms work has been done using problems where $\mathcal{X} = A^n$ for some alphabet A
 - Valid choices for variable i don't depend on choice for variable j
(Not saying the problem is linear, just that all strings in A^n are valid)
- State of the art techniques use k -degree acyclic graphs and statistical tests for conditional dependence to model f .



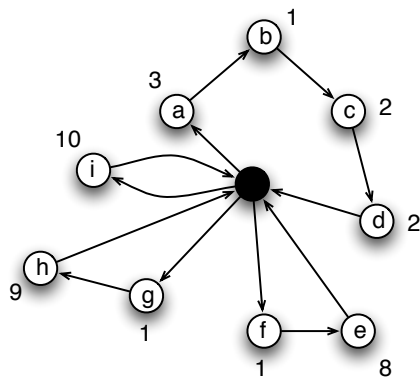
$$p(\vec{x}) = p(x_3|x_2, x_4) \cdot p(x_2|x_1) \cdot p(x_1) \cdot p(x_4)$$

- These can provably solve k -decomposable problems over n variables to optimality in $O(n^2)$ time and polynomial spaces.

A couple of routing problems

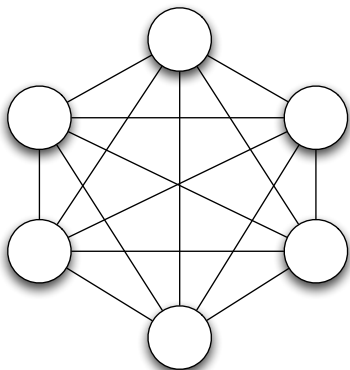


$$\text{TSP: } f(x) = \sum_i m(x_i, x_{i \oplus 1})$$



$$\text{CVRP: } f(x) = \sum_{c \in x} f(c) \text{ subject to weight constraint}$$

Why graphical models aren't (immediately) usable



Dependencies between variables in a permutation problem of size 6

Transition matrices for $p(x \in \pi(V))$

- **Idea:** presume that edges contribute independently to the quality of a tour (blatantly false).
- Distribution over high quality solutions can be expressed as a transition matrix:

$$\begin{bmatrix} p(v_0 \rightarrow v_0) & p(v_1 \rightarrow v_0) & p(v_2 \rightarrow v_0) & \cdots \\ p(v_0 \rightarrow v_1) & p(v_1 \rightarrow v_1) & p(v_2 \rightarrow v_1) & \cdots \\ p(v_0 \rightarrow v_2) & p(v_1 \rightarrow v_2) & p(v_2 \rightarrow v_2) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- **Sample:** random walk, enforcing permutation constraint by ignoring repeated visits
- **Estimate:** use (Laplace corrected) marginal frequencies of each edge as new transition probability

Transition matrices for $p(x \in \pi(V))$

- **Idea:** presume that edges contribute independently to the quality of a tour (blatantly false).
- Distribution over high quality solutions can be expressed as a transition matrix:

$$\begin{bmatrix} p(v_0 \rightarrow v_0) & p(v_1 \rightarrow v_0) & p(v_2 \rightarrow v_0) & \cdots \\ p(v_0 \rightarrow v_1) & p(v_1 \rightarrow v_1) & p(v_2 \rightarrow v_1) & \cdots \\ p(v_0 \rightarrow v_2) & p(v_1 \rightarrow v_2) & p(v_2 \rightarrow v_2) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- **Sample:** random walk, enforcing permutation constraint by ignoring repeated visits
- **Estimate:** use (Laplace corrected) marginal frequencies of each edge as new transition probability

Transition matrices for $p(x \in \pi(V))$

- **Idea:** presume that edges contribute independently to the quality of a tour (blatantly false).
- Distribution over high quality solutions can be expressed as a transition matrix:

$$\begin{bmatrix} p(v_0 \rightarrow v_0) & p(v_1 \rightarrow v_0) & p(v_2 \rightarrow v_0) & \cdots \\ p(v_0 \rightarrow v_1) & p(v_1 \rightarrow v_1) & p(v_2 \rightarrow v_1) & \cdots \\ p(v_0 \rightarrow v_2) & p(v_1 \rightarrow v_2) & p(v_2 \rightarrow v_2) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- **Sample:** random walk, enforcing permutation constraint by ignoring repeated visits
- **Estimate:** use (Laplace corrected) marginal frequencies of each edge as new transition probability

Transition matrices for $p(x \in \pi(V))$

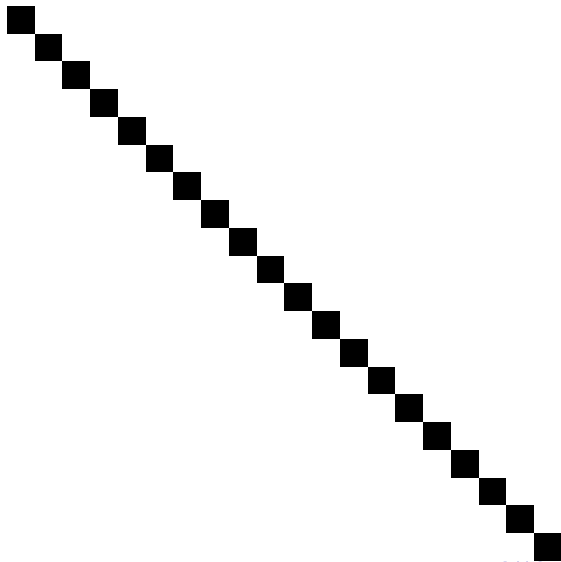
- **Idea:** presume that edges contribute independently to the quality of a tour (blatantly false).
- Distribution over high quality solutions can be expressed as a transition matrix:

$$\begin{bmatrix} p(v_0 \rightarrow v_0) & p(v_1 \rightarrow v_0) & p(v_2 \rightarrow v_0) & \cdots \\ p(v_0 \rightarrow v_1) & p(v_1 \rightarrow v_1) & p(v_2 \rightarrow v_1) & \cdots \\ p(v_0 \rightarrow v_2) & p(v_1 \rightarrow v_2) & p(v_2 \rightarrow v_2) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- **Sample:** random walk, enforcing permutation constraint by ignoring repeated visits
- **Estimate:** use (Laplace corrected) marginal frequencies of each edge as new transition probability

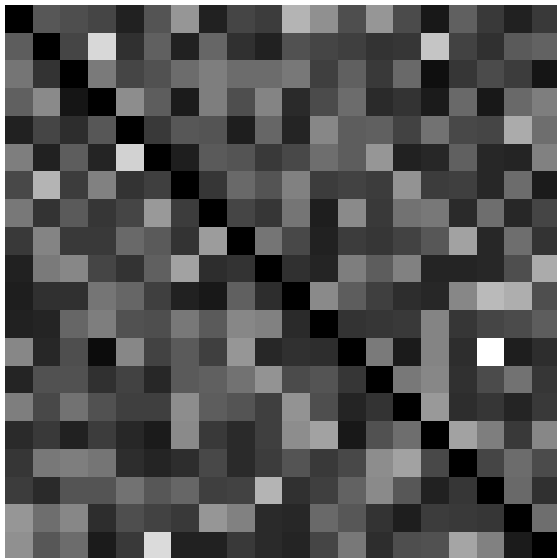
Animated example

Does anyone know why it moves sideways?



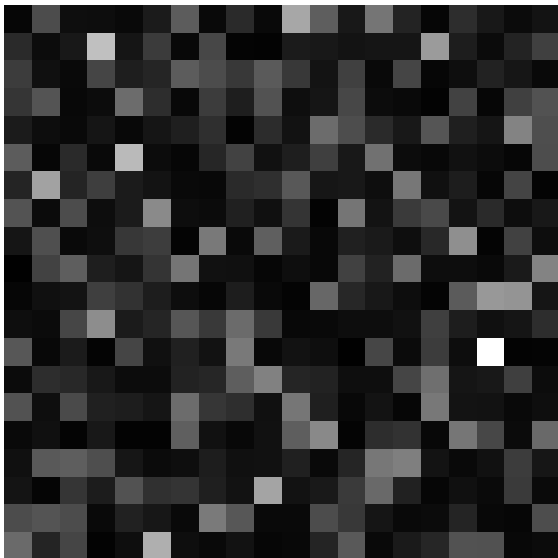
Animated example

Does anyone know why it moves sideways?



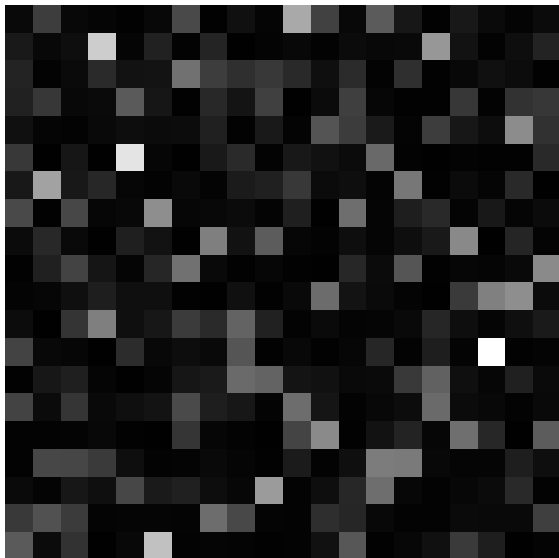
Animated example

Does anyone know why it moves sideways?



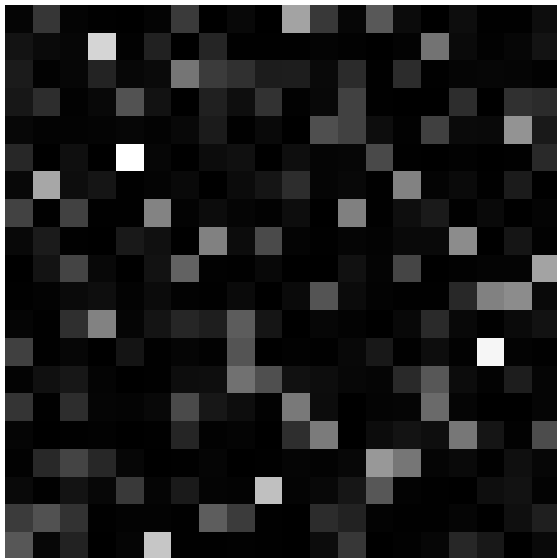
Animated example

Does anyone know why it moves sideways?



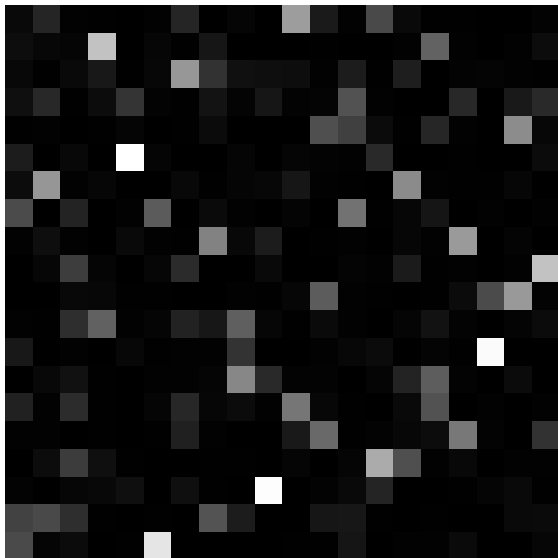
Animated example

Does anyone know why it moves sideways?



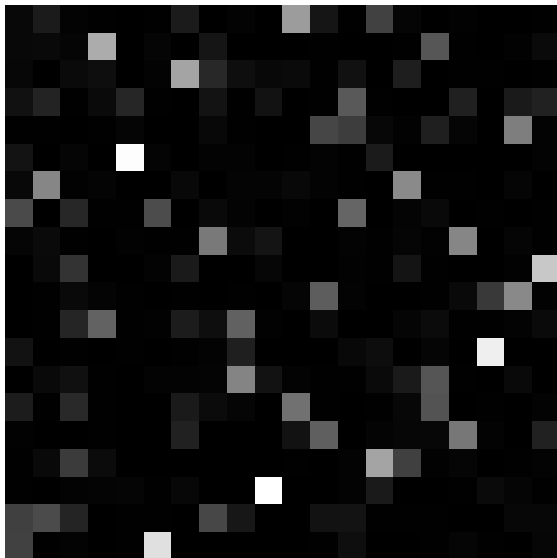
Animated example

Does anyone know why it moves sideways?



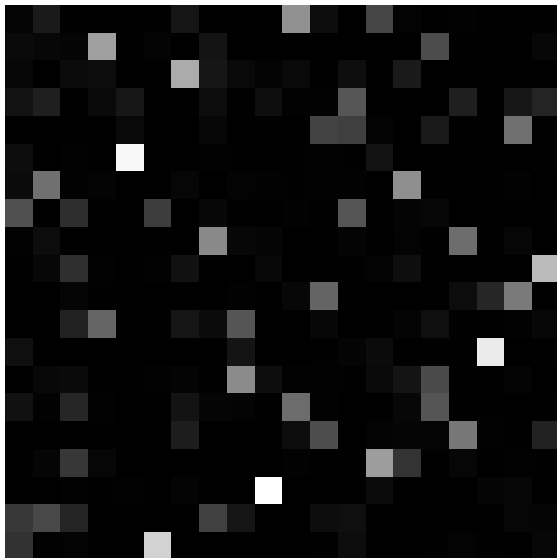
Animated example

Does anyone know why it moves sideways?



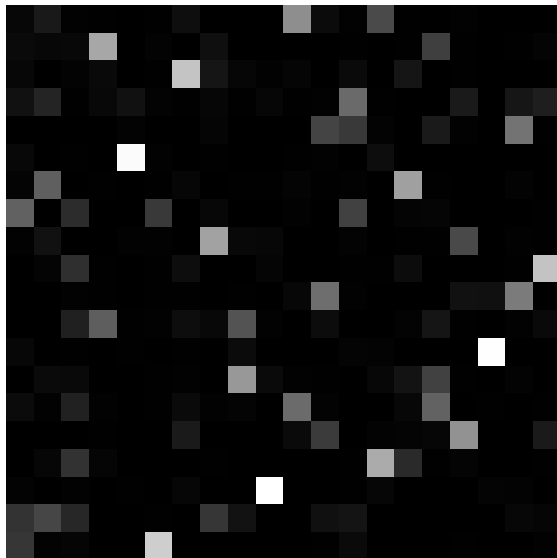
Animated example

Does anyone know why it moves sideways?



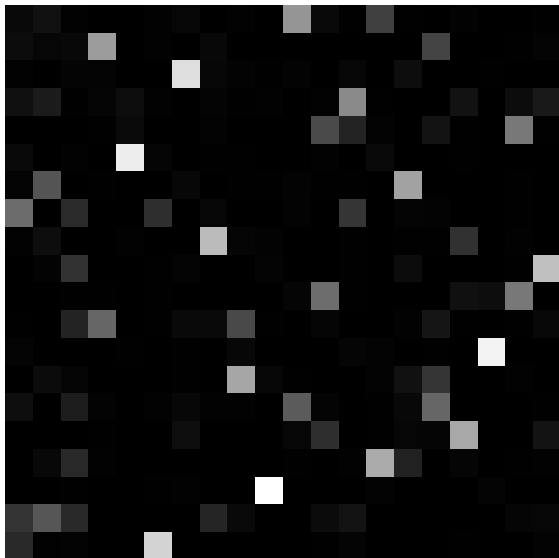
Animated example

Does anyone know why it moves sideways?



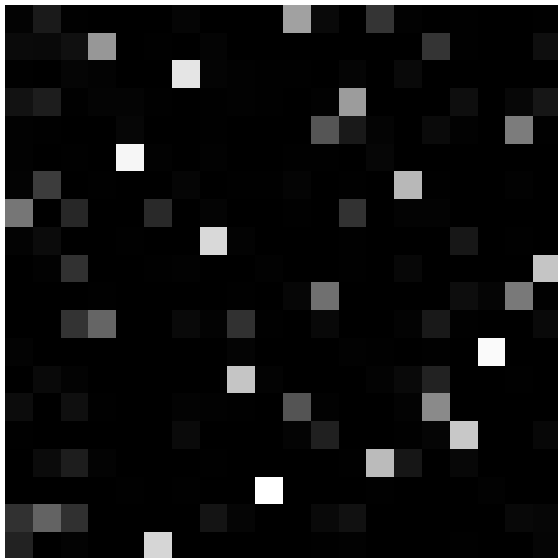
Animated example

Does anyone know why it moves sideways?



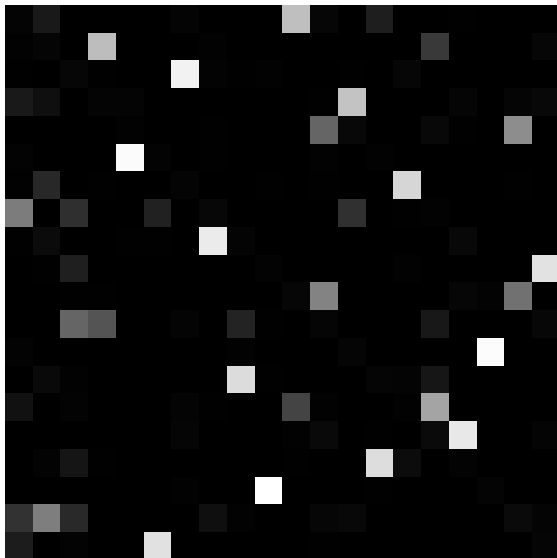
Animated example

Does anyone know why it moves sideways?



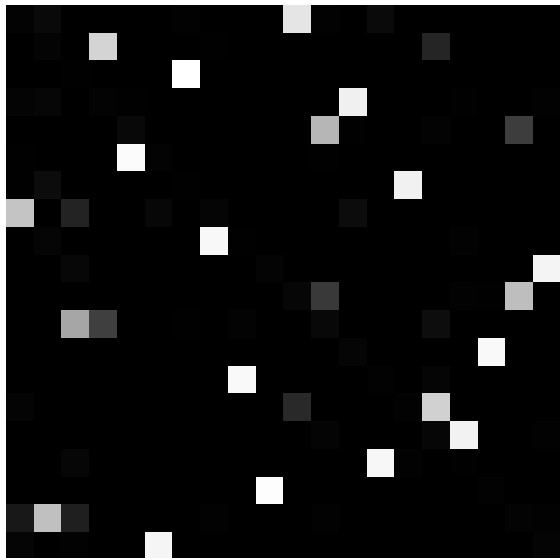
Animated example

Does anyone know why it moves sideways?



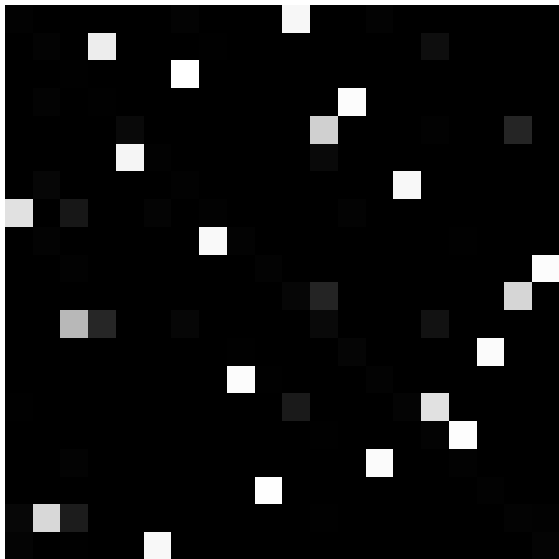
Animated example

Does anyone know why it moves sideways?



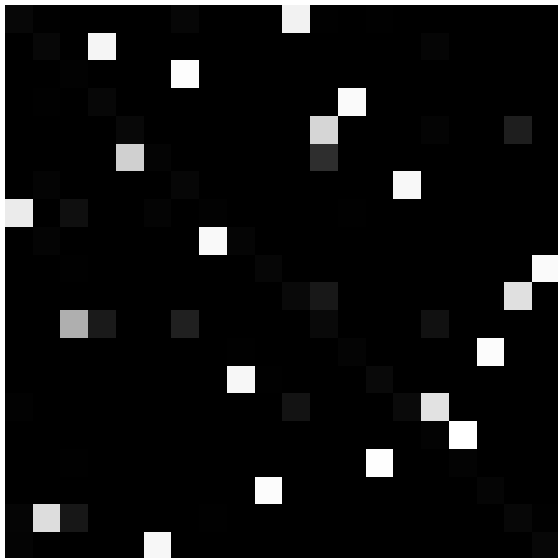
Animated example

Does anyone know why it moves sideways?



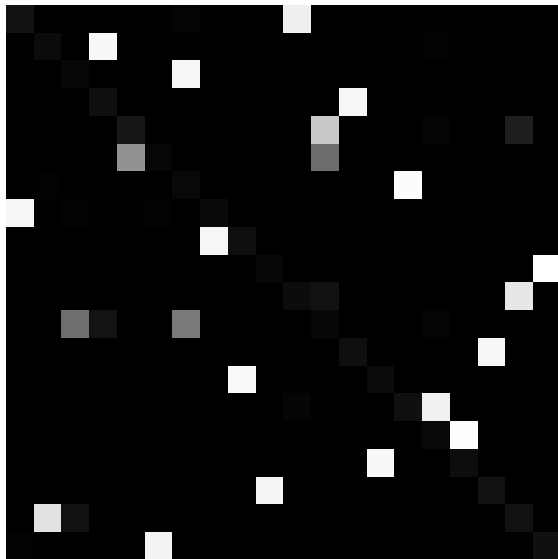
Animated example

Does anyone know why it moves sideways?



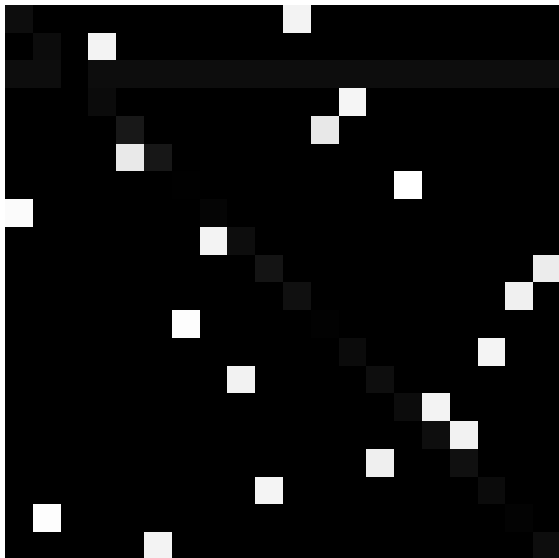
Animated example

Does anyone know why it moves sideways?



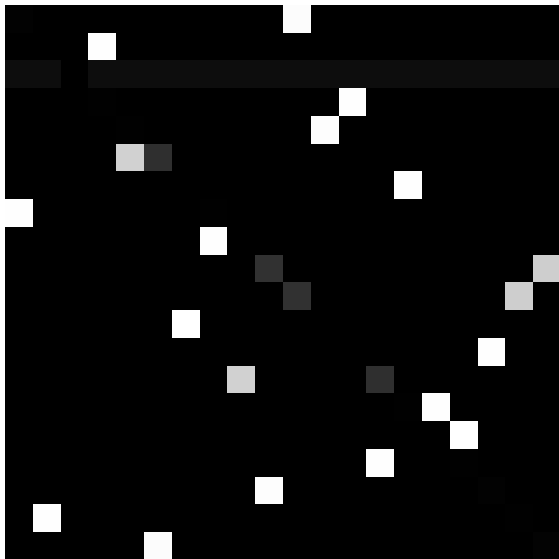
Animated example

Does anyone know why it moves sideways?



Animated example

Does anyone know why it moves sideways?



Using a wider locale

- This technique works surprisingly well for metric instances
 - More a reflection of the strong locality in almost all metric TSP instances than anything else
- Relaxing the triangle inequality impairs algorithm effectiveness
- A possible solution: extend the matrix & model to have some fixed “memory”

$$\begin{bmatrix} p(v_0 \rightarrow v_0 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_0 | x_{i-1}, x_{i-2}, \dots) & \dots \\ p(v_0 \rightarrow v_1 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_1 | x_{i-1}, x_{i-2}, \dots) & \dots \\ p(v_0 \rightarrow v_2 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_2 | x_{i-1}, x_{i-2}, \dots) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

- Unfortunately requires n^k variables for k edges of “memory”, (and still supposes locality on a larger scale)
- Does provide moderate improvement (about 1 standard deviation) for some degree of perturbation.

Using a wider locale

- This technique works surprisingly well for metric instances
 - More a reflection of the strong locality in almost all metric TSP instances than anything else
- Relaxing the triangle inequality impairs algorithm effectiveness
- A possible solution: extend the matrix & model to have some fixed “memory”

$$\begin{bmatrix} p(v_0 \rightarrow v_0 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_0 | x_{i-1}, x_{i-2}, \dots) & \dots \\ p(v_0 \rightarrow v_1 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_1 | x_{i-1}, x_{i-2}, \dots) & \dots \\ p(v_0 \rightarrow v_2 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_2 | x_{i-1}, x_{i-2}, \dots) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

- Unfortunately requires n^k variables for k edges of “memory”, (and still supposes locality on a larger scale)
- Does provide moderate improvement (about 1 standard deviation) for some degree of perturbation.

Using a wider locale

- This technique works surprisingly well for metric instances
 - More a reflection of the strong locality in almost all metric TSP instances than anything else
- Relaxing the triangle inequality impairs algorithm effectiveness
- A possible solution: extend the matrix & model to have some fixed “memory”

$$\begin{bmatrix} p(v_0 \rightarrow v_0 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_0 | x_{i-1}, x_{i-2}, \dots) & \dots \\ p(v_0 \rightarrow v_1 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_1 | x_{i-1}, x_{i-2}, \dots) & \dots \\ p(v_0 \rightarrow v_2 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_2 | x_{i-1}, x_{i-2}, \dots) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

- Unfortunately requires n^k variables for k edges of “memory”, (and still supposes locality on a larger scale)
- Does provide moderate improvement (about 1 standard deviation) for some degree of perturbation.

Using a wider locale

- This technique works surprisingly well for metric instances
 - More a reflection of the strong locality in almost all metric TSP instances than anything else
- Relaxing the triangle inequality impairs algorithm effectiveness
- A possible solution: extend the matrix & model to have some fixed “memory”

$$\begin{bmatrix} p(v_0 \rightarrow v_0 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_0 | x_{i-1}, x_{i-2}, \dots) & \dots \\ p(v_0 \rightarrow v_1 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_1 | x_{i-1}, x_{i-2}, \dots) & \dots \\ p(v_0 \rightarrow v_2 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_2 | x_{i-1}, x_{i-2}, \dots) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

- Unfortunately requires n^k variables for k edges of “memory”, (and still supposes locality on a larger scale)
- Does provide moderate improvement (about 1 standard deviation) for some degree of perturbation.

Using a wider locale

- This technique works surprisingly well for metric instances
 - More a reflection of the strong locality in almost all metric TSP instances than anything else
- Relaxing the triangle inequality impairs algorithm effectiveness
- A possible solution: extend the matrix & model to have some fixed “memory”

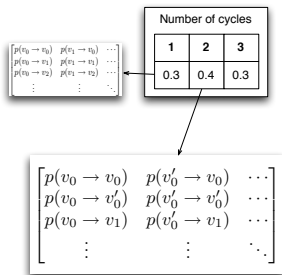
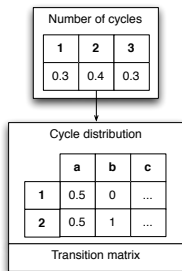
$$\begin{bmatrix} p(v_0 \rightarrow v_0 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_0 | x_{i-1}, x_{i-2}, \dots) & \dots \\ p(v_0 \rightarrow v_1 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_1 | x_{i-1}, x_{i-2}, \dots) & \dots \\ p(v_0 \rightarrow v_2 | x_{i-1}, x_{i-2}, \dots) & p(v_1 \rightarrow v_2 | x_{i-1}, x_{i-2}, \dots) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

- Unfortunately requires n^k variables for k edges of “memory”, (and still supposes locality on a larger scale)
- Does provide moderate improvement (about 1 standard deviation) for some degree of perturbation.

Further work, sticking points

Multiple cycle problems, ordering issues

- **Question:** How to model distribution over partitions + cycles?



Distribution over partitions + over edges

Partition count + dummy variables

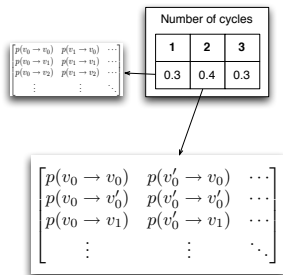
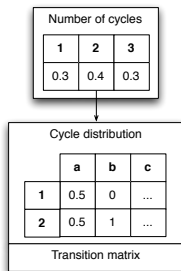
- All of these have two problems when sampling

- 1 Cycles have no identity, so how do we allocate the probability
- 2 Many variables \implies very big samples to keep estimator variance low

Further work, sticking points

Multiple cycle problems, ordering issues

- **Question:** How to model distribution over partitions + cycles?



Distribution over partitions + over edges

Partition count + dummy variables

- All of these have two problems when sampling
 - 1 Cycles have no identity, so how do we allocate the probability
 - 2 Many variables \implies very big samples to keep estimator variance low

Extending graphical models?

- The difficulty with graphical models was that permutation problems have n^2 constraints
- These constraints are very regular - can we do anything with that?
- Possibility: construct the graphical model in the normal way, and enforce permutation constraints when sampling, with some fix to deal with values that become illegal?

Extending graphical models?

- The difficulty with graphical models was that permutation problems have n^2 constraints
- These constraints are very regular - can we do anything with that?
- Possibility: construct the graphical model in the normal way, and enforce permutation constraints when sampling, with some fix to deal with values that become illegal?

Extending graphical models?

- The difficulty with graphical models was that permutation problems have n^2 constraints
- These constraints are very regular - can we do anything with that?
- Possibility: construct the graphical model in the normal way, and enforce permutation constraints when sampling, with some fix to deal with values that become illegal?

Extending graphical models?

- The difficulty with graphical models was that permutation problems have n^2 constraints
- These constraints are very regular - can we do anything with that?
- Possibility: construct the graphical model in the normal way, and enforce permutation constraints when sampling, with some fix to deal with values that become illegal?

Extending graphical models?

- The difficulty with graphical models was that permutation problems have n^2 constraints
- These constraints are very regular - can we do anything with that?
- Possibility: construct the graphical model in the normal way, and enforce permutation constraints when sampling, with some fix to deal with values that become illegal?

THE END – SUGGESTIONS?

Bonus puzzle slide!

Optimal strategy required!

You are given two sets \mathcal{X} and \mathcal{Y} , and some unknown function $f : \mathcal{X} \rightarrow \mathcal{Y}$. You are allowed to interactively make n calls to f , so the i th value you choose may depend on the previous $i - 1$ and their corresponding values under f . Repeated queries are allowed, and may help:

The objective is to maximise the sum of the *maximum* f -values you observe, so you may want to

- Sample n distinct values from \mathcal{X} , hoping for a really big win
- Sample (say) $n/2$ distinct values from \mathcal{X} , and then spend your remaining $n/2$ evaluations on revisiting the maximum point you've observed so far.
- Something in between? Something entirely different?

You do not know the distribution of $f - f(x_0)$ is completely uninformative about $f(x_1)$. \mathcal{Y} could be a big set, like \mathbb{N} .

Some relevant papers



M. Pelikan and D. E. Goldberg.

Hierarchical bayesian optimization algorithm = bayesian optimization algorithm + niching + local structures.

In *Optimization by Building and Using Probabilistic Models (OBUPM) 2001*, pages 217–221, San Francisco, California, USA, 7 July 2001.



M. Pelikan, D. E. Goldberg, and F. G. Lobo.

A survey of optimization by building and using probabilistic models.

Comput. Optim. Appl., 21(1):5–20, 2002.



R. Y. Rubinstein.

Optimization of computer simulation models with rare events.

European Journal of Operational Research, 99(1):89–112, May 1997.

available at

<http://ideas.repec.org/a/eee/ejores/v99y1997i1p89-112.html>.