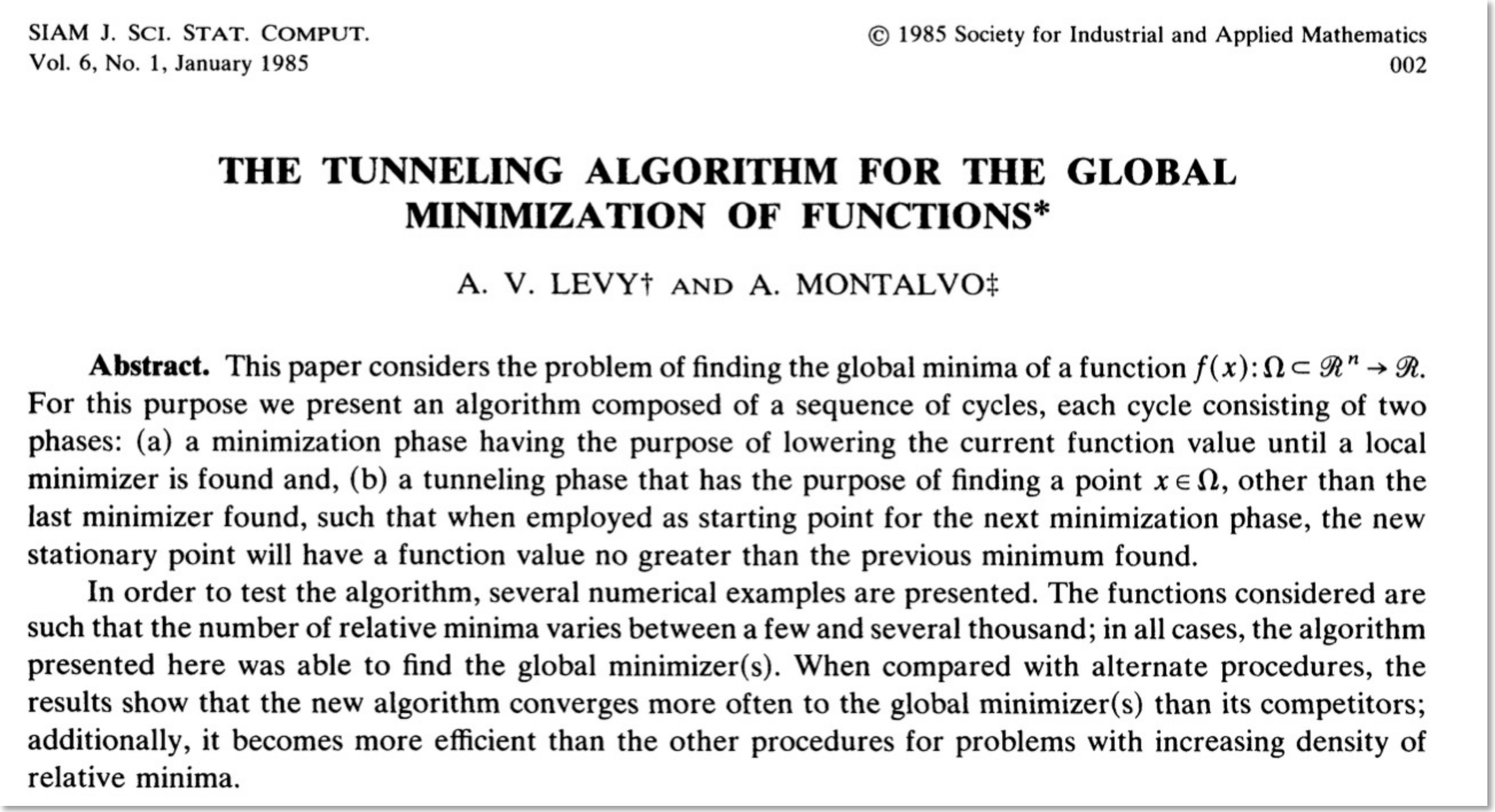
Tunneling Algorithm for Global Minimization

Marcus Nørgaard Weng, MSc Nanoscience

KØBENHAVNS UNIVERSITET

Journal Article


SIAM J. SCI. STAT. COMPUT.
Vol. 6, No. 1, January 1985

THE TUNNELING ALGORITHM FOR THE GLOBAL MINIMIZATION OF FUNCTIONS*

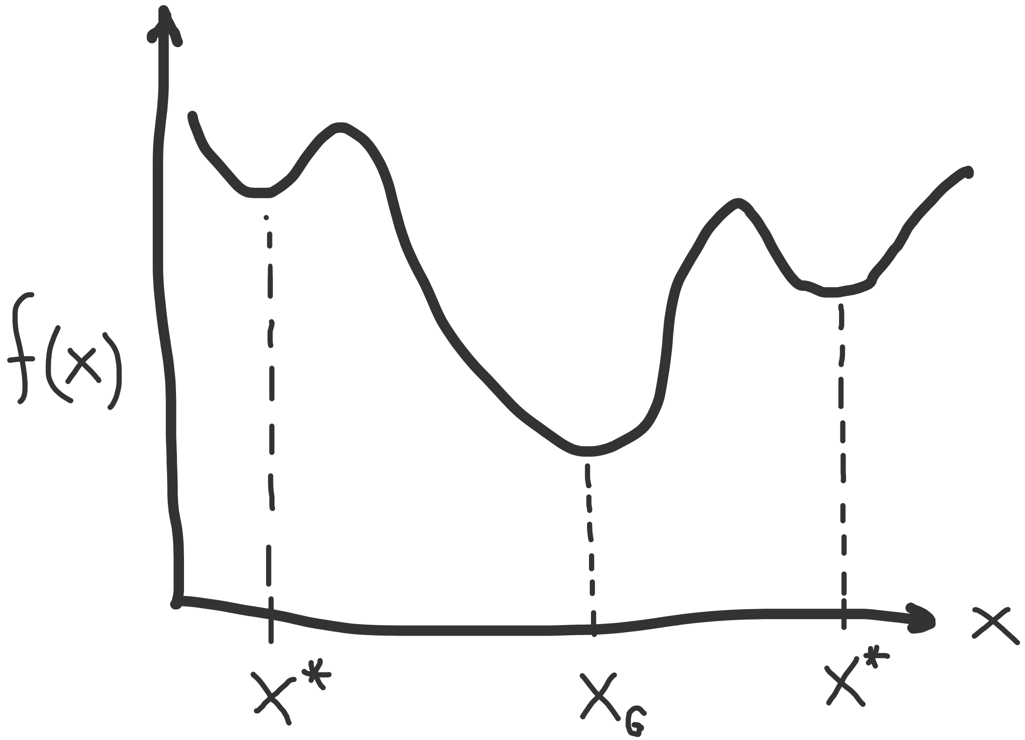
A. V. LEVY† AND A. MONTALVO‡

Abstract. This paper considers the problem of finding the global minima of a function $f(x): \Omega \subset \mathcal{R}^n \to \mathcal{R}$. For this purpose we present an algorithm composed of a sequence of cycles, each cycle consisting of two phases: (a) a minimization phase having the purpose of lowering the current function value until a local minimizer is found and, (b) a tunneling phase that has the purpose of finding a point $x \in \Omega$, other than the last minimizer found, such that when employed as starting point for the next minimization phase, the new stationary point will have a function value no greater than the previous minimum found.

In order to test the algorithm, several numerical examples are presented. The functions considered are such that the number of relative minima varies between a few and several thousand; in all cases, the algorithm presented here was able to find the global minimizer(s). When compared with alternate procedures, the results show that the new algorithm converges more often to the global minimizer(s) than its competitors; additionally, it becomes more efficient than the other procedures for problems with increasing density of relative minima.

Introduction

- Assume f(x), $x_{min} < x < x_{max}$
- f(x) is real and has gradient ∇f(x)
- We cannot solve x in ∇f(x) = 0
- We want to find the input x corresponding to the minimum f(x)
- f(x) could be a – LLH fit of a model with parameters x to experimental data
- For just 1 parameter we could 1D raster scan
- Not feasible for co-dependant parameters in 8 dimentions



Introduction

- This is attempted with minimization algorithms

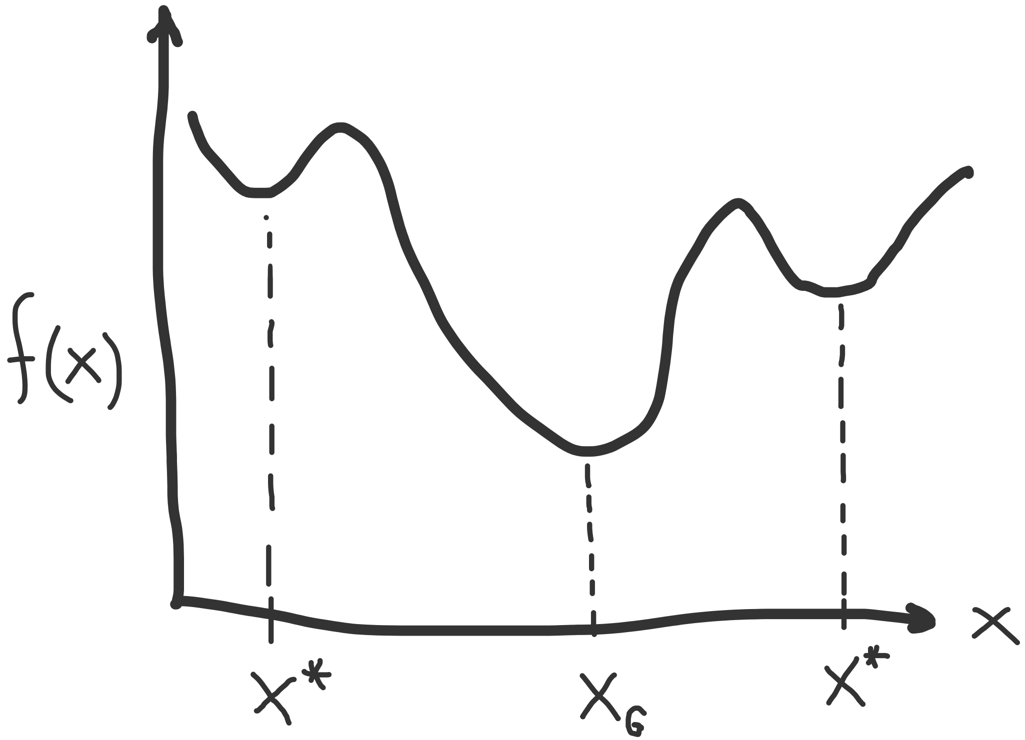
Newton's Method:

- $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

Gradient Descent:

- $x_{n+1} = x_n - \gamma_n \nabla f(x_n)$

Works if: All starting points x_0 lead to the global minimum without passing by local minima



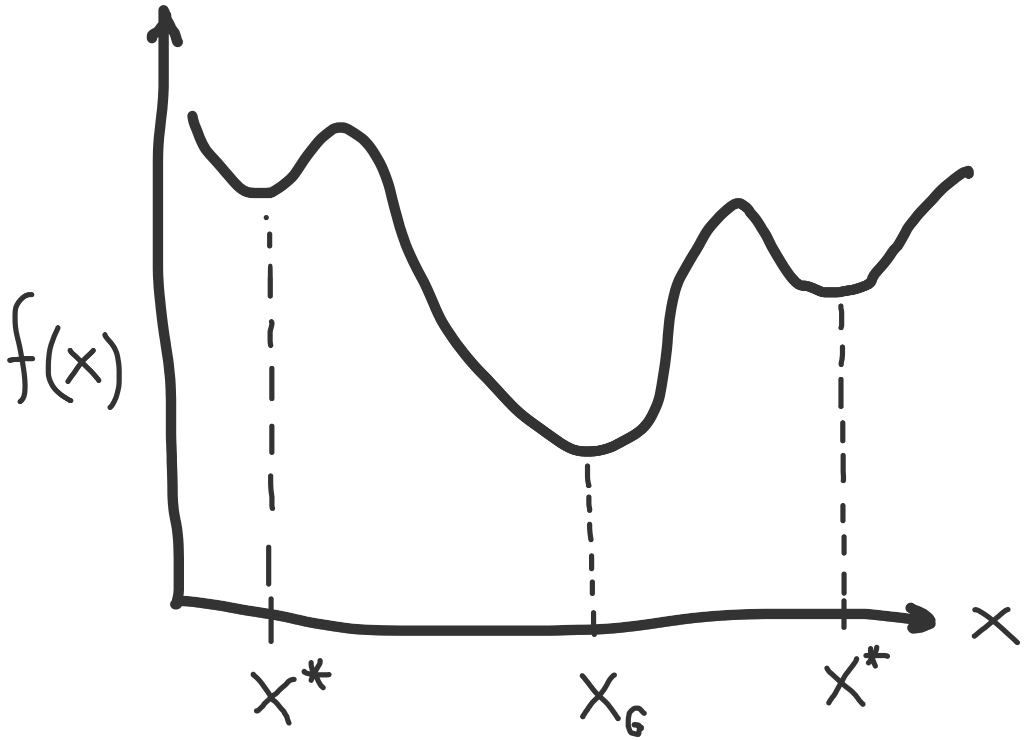
Introduction

Momentum:

- SGD with momentum to decrease oscillations
- $x_{n+1} = x_n - \eta \nabla f(x_n) + \alpha \nabla f(x_{n-1})$

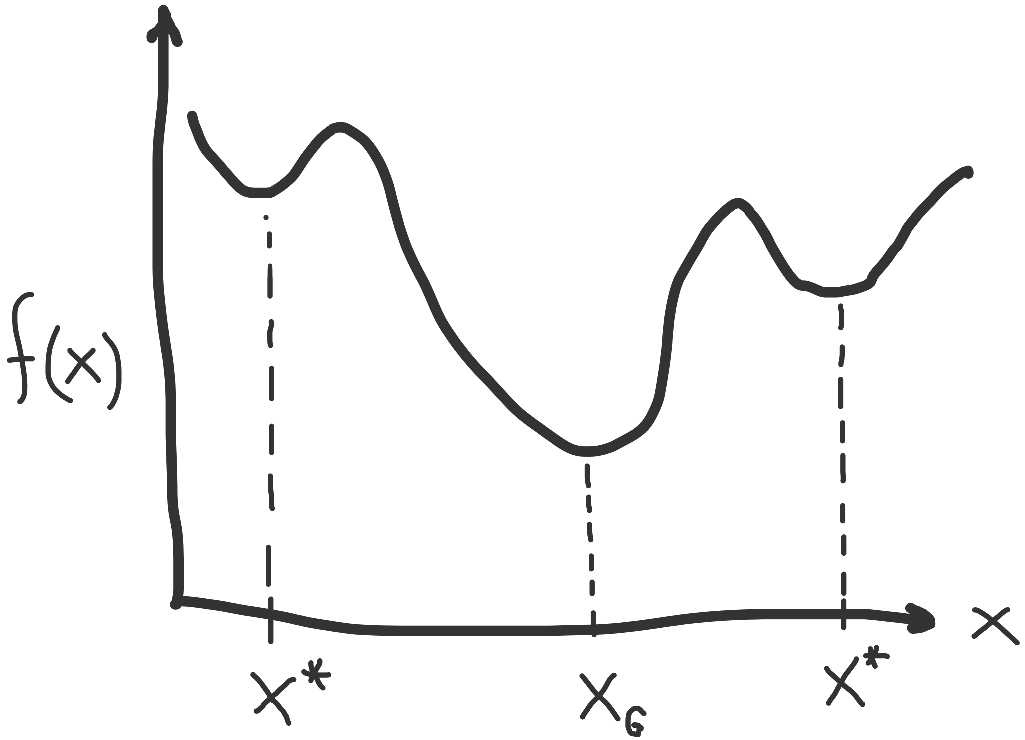
Adaptive Gradient (AdaGrad):

- SGD with per-parameter scalar γ_n
- Maybe some parameters are on a much larger scale than others
- $G = \sum g_\tau \cdot g_\tau^t$
- $x_{n+1,j} = x_{n,j} - \frac{\eta}{\sqrt{G_{i,j}}} \cdot g_j$



The problem

- Gradient-based minimization algorithms are prone to local minima
- The global minimum is the desired minimum



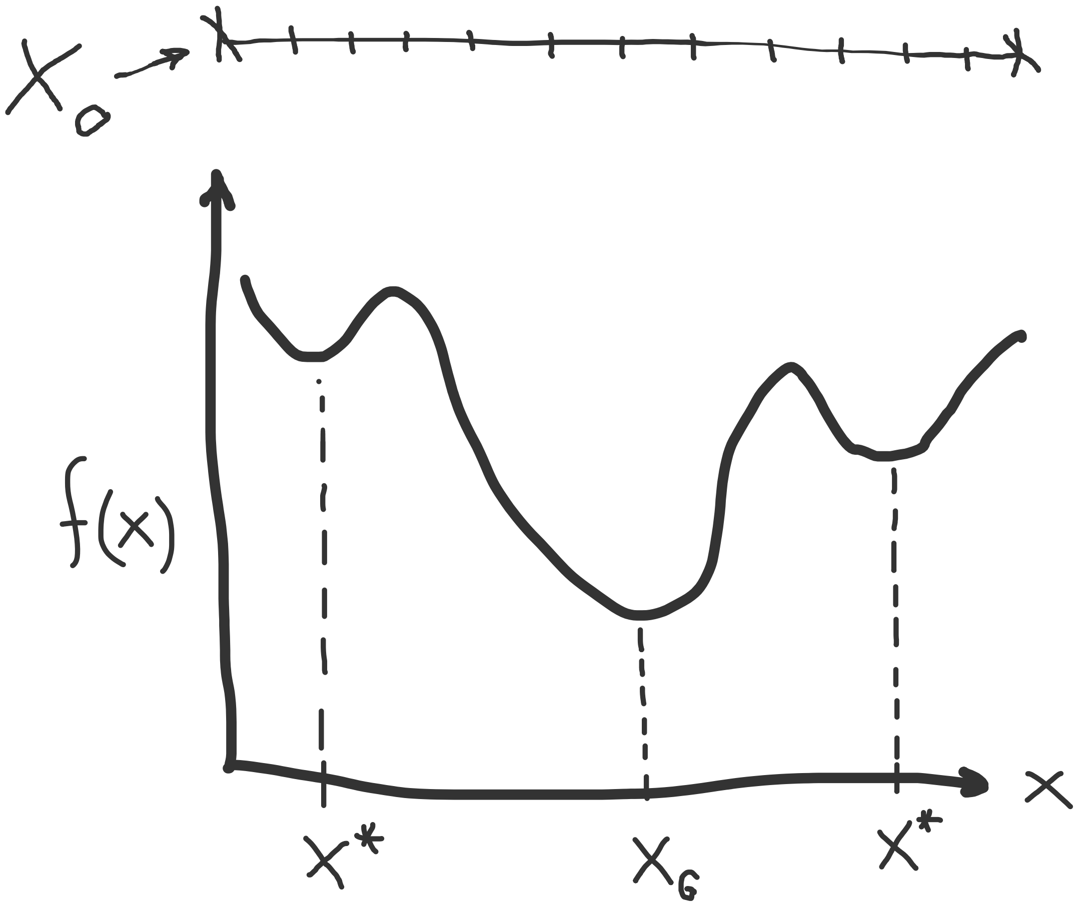
The Tunneling Algorithm

- **Goal**: Finding the global minimum of a function f – avoiding local minima
- An iterative algorithm in two phases:

1: **Minimizing phase**

A gradient-based minimization algorithm is started from a range of x_0

The found local minima x* are saved



The Tunneling Algorithm

2: **Tunneling phase**

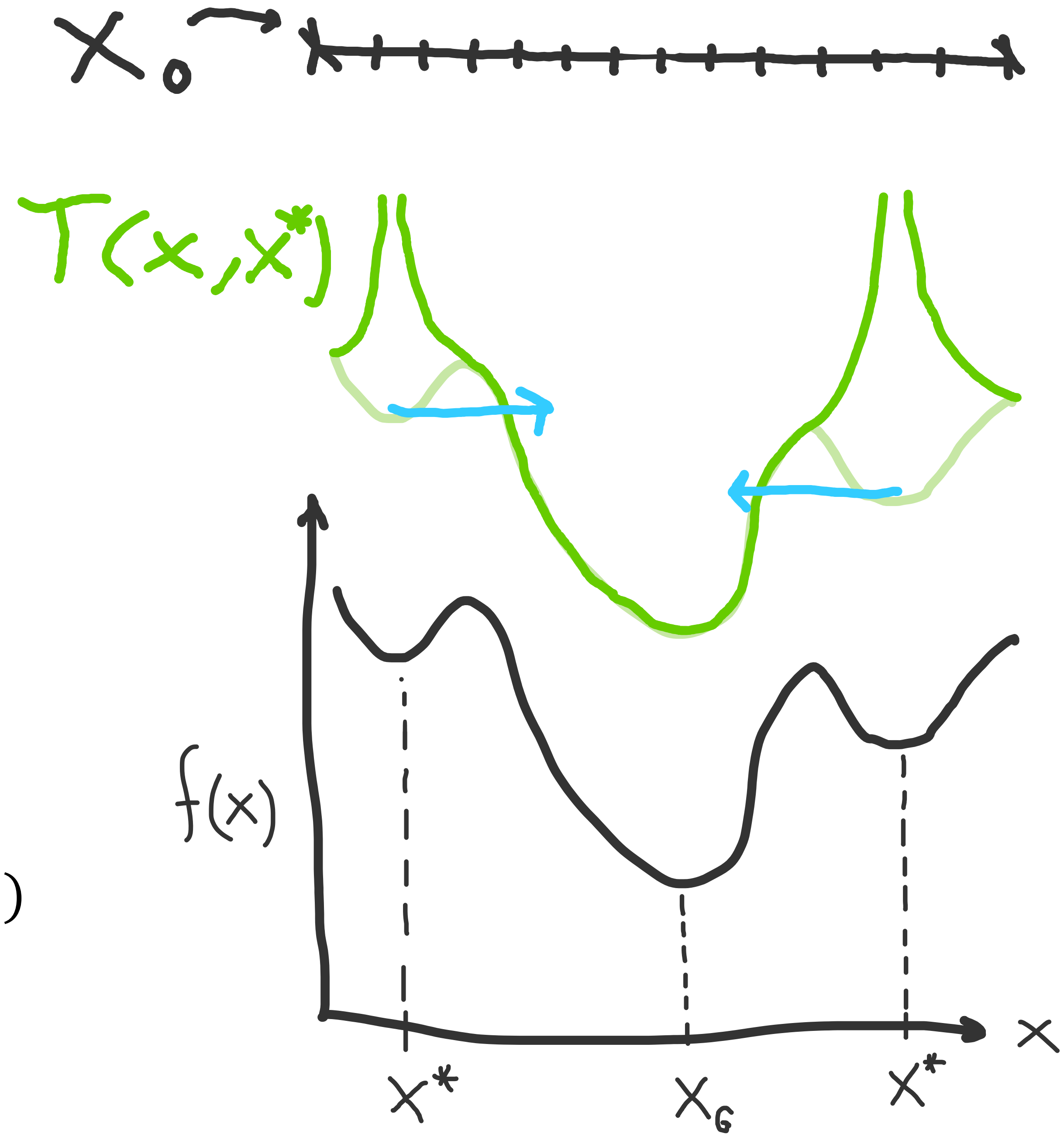
$$T(x, x^*) = \frac{f(x) - f'(x^*)}{[(x - x^*)' \cdot (x - x^*)]^{\eta}}$$

A new function T is constructed by altering $f(x)$, adding *poles* at x^*.

$$\frac{1}{x - x^*} \rightarrow \infty \; when \; x \rightarrow x^*$$

This discourages the gradient-based minimization algorithm from approaching the local minima on $f(x)$

Minimization algorithm is now used on $T(x, x^*)$



Measurement of success

How do we know if the minimization algorithm is better than SGD?

1. Choose test functions
   i. 16 functions from 2 to 10 dimensions with known global minima
2. Choose minimizers for comparison
   i. Gradient Descent
3. Attempt minimization of test functions with all minimizers from many starting points
4. Measure $n_{iterations}$ and t_{CPU} for each minimization

5. Measure success P:

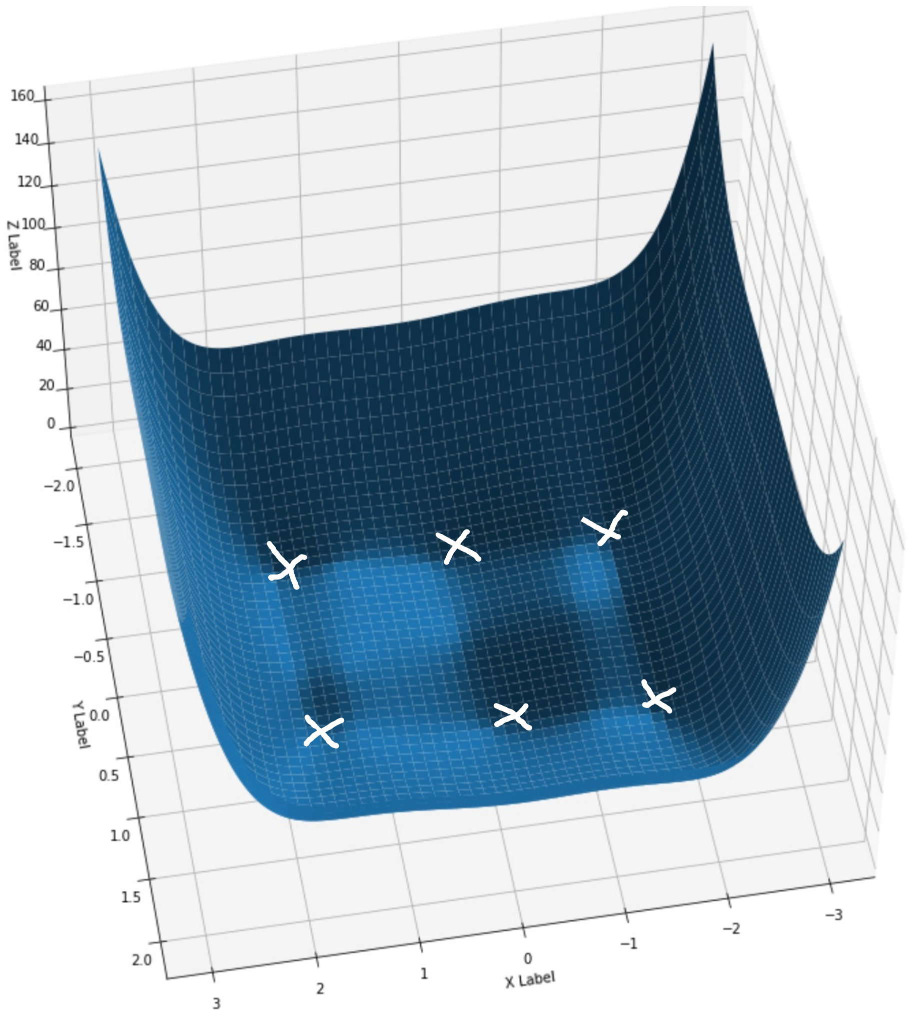
$$P = \frac{\sum_{i=1}^{N_r} M_i}{N_G \cdot N_r}$$

N_r: Number of starting points
M_i: Number of found global minima
N_g: Number of global minima

Test-function: The six-hump camelback function

$$f(x_1, x_2) = [4 - 2.1x_1^2 + \tfrac{1}{3}x_1^4]x_1^2 + x_1x_2 + [-4 + 4x_2^2]x_2^2, \qquad -3 \leqq x_1 \leqq 3, \ -2 \leqq x_2 \leqq 2$$

6 local minima
2 global minima



Results

TABLE 1
Summary of numerical results for the tunneling, MRS, and MMRS methods.

Ex/dim.	Method	Total time (CPU-secs)	Total evaluations: Functions	Total evaluations: Gradient	Minimization time (CPU-secs)	Evaluations during minimization phase: Functions	Evaluations during minimization phase: Gradient	Number of minimizations	p
1/2	Tunnel	87.045	12,160	1,731	2.113	1,047	97	17	0.9445
	MRS	88.089	35,479	9,572	87.845	35,479	9,572	244	0.5
	MMRS	87.066	50,462	7	0.148	75	7	1	0.0555
2/2	Tunnel	8.478	2,912	390	1.045	525	53	3	1
	MRS	5.197	*	*	*	*	*	10	0
	MMRS	4.313	*	*	*	*	*	2	0
3/2	Tunnel	5.984	2,180	274	1.409	710	69	3	1
	MRS	2.094	*	*	*	*	*	4	0
	MMRS	6.018	3,350	19	0.292	136	19	1	0
4/2	Tunnel	1.984	1,496	148	0.033	57	17	2	1
	MRS	0.036	61	19	0.034	61	19	2	1
	MMRS	2.036	6,000	10	0.016	32	10	1	0.5
5/2	Tunnel	3.238	2,443	416	0.947	1,116	273	2.75	1
	MRS	64.0	*	*	*	*	*	1	0
	MMRS	1.062	1,241	287	0.997	1,195	287	3	1
6/3	Tunnel	12.915	7,325	1,328	4.435	3,823	848	3.25	1
	MRS	3.516	2,861	784	3.485	2,861	784	3	1
	MMRS	4.024	3,443	726	3.231	2,647	726	3	1
7/4	Tunnel	20.450	4,881	1,371	1.91	1,126	376	4.25	1
	MRS	3.391	*	*	*	*	*	6	0
	MMRS	4.335	3,076	457	2.289	1,369	457	2	1
8/5	Tunnel	11.885	7,540	1,122	9.32	6,471	881	2	1
	MRS	8.107	*	*	*	*	*	1	0
	MMRS	11.925	9,458	171	2.112	1,506	171	1	0
9/8	Tunnel	45.474	19,366	2,370	35.644	16,138	2,011	2.5	1
	MRS	38.091	17,229	2,143	38.091	17,229	2,143	2	1
	MMRS	45.535	22,193	1,771	30.757	14,126	1,771	1	0
10/10	Tunnel	68.22	23,982	3,272	67.283	22,191	3,135	2.5	1
	MRS	192.0	*	*	*	*	*	1	0
	MMRS	68.26	25,966	2,913	62.47	23,093	2,913	1	0
11/2	Tunnel	4.364	2,613	322	0.762	736	158	2.5	0.5
	MRS	6.308	6,851	876	6.308	6,851	876	5	0
	MMRS	1.792	1,867	250	1.406	1,441	250	4	1
12/3	Tunnel	12.378	6,955	754	3.927	3,142	479	4.25	0.75
	MRS	13.291	10,566	1,652	13.227	10,566	1,652	9	0
	MMRS	2.975	2,316	359	2.139	2,316	359	3	1
13/4	Tunnel	8.35	3,861	588	3.076	1,863	390	3.75	0.75
	MRS	9.851	6,659	740	9.821	6,659	740	2	0
	MMRS	8.376	6,234	273	2.294	1,419	273	2	0
14/5	Tunnel	28.33	10,715	1,507	7.249	3,565	797	5.5	0.75
	MRS	51.707	28,347	4,002	51.712	28,347	4,002	7	0
	MMRS	28.362	17,339	1,098	11.150	5,746	1,098	3	0
15/6	Tunnel	33.173	12,786	1,777	17.282	7,839	1,329	3.5	1
	MRS	41.065	19,301	2,784	41.028	19,301	2,784	2	0
	MMRS	33.231	18,985	132	1.263	479	132	2	0
16/7	Tunnel	71.981	16,063	2,792	15.350	6,142	1,013	7.5	0.75
	MRS	92.615	38,483	5,411	92.546	38,483	5,411	8	0
	MMRS	72.027	36,195	435	5.977	2,132	435	2	0

* Failure in convergence.

Tunneling

Often higher P than SGD

Conclusion

The tunneling algorithm is better at finding the global minima than the compared minimization algorithms.

Purpose of the tunneling algorithm: Find the global minimizer, avoid local minima

The results indicate that it succeeds much more often than simple gradient-descent based minimisers – **Success!**

Nota bene:

- Optimized for finding the global minimum – not always needed!
- Results highly dependant on chosen test functions
- Extremely cool name, much cooler than Gradient Descent