Cooperative bare bones optimisation

May 29, 2013

Tim Blackwell, Goldsmiths, University of London

1 In the beginning

. . . there was SPECT

2 Particle swarm optimisation

PSO

n component vectors $x_i, p_i, u_{1,2}$
scalars $w, \phi_{1,2}$
N_i social neighbourhood of particle i

$$u_1, u_2 \sim U(0,1)^n$$
$$x_i \leftarrow x_i + wv_i + \frac{\phi_1}{2} u_1 \circ (p_i - x_i) + \frac{\phi_2}{2} u_2 \circ (g_i - x_i)$$

$$p_i \leftarrow BEST(p_i, x_i)$$
$$g_i \leftarrow BEST(p_j \in N_i)$$

Kennedy's bare bones version of PSO

$$
\begin{aligned}
\mu &= \frac{g_i + p_i}{2} \\
\delta &= |g_i - p_i| \\
x &= \mu + \delta N(0,1) \\
p_i &\leftarrow BEST(p_i, x)
\end{aligned}
$$

Bare bones with jumps

Two neighbourhoods, one for the focus determination and one for the spread.

A hidden scalar α multiplying the spread parameter δ is revealed. Tune α to the edge of collapse for maximum convergence rate.

A small probability of jumping in each component - a tail broadening measure, but unlike heavy tailed distributions such as Cauchy or Lévy, the broadening does not scale with the distribution parameters.

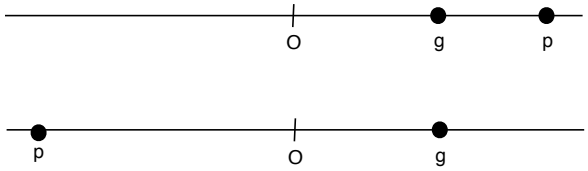
Bare bones with jumps: the workings

$$
\begin{aligned}
\theta &= (u \sim U(0,1) < p_J) \ ? \ 1 : 0 \\
\mu &= BEST(p_i \in N_i) \\
\delta &= DIFF(p_j, p_k \in M_i) \\
x &= (1-\theta)(\mu + \alpha\delta N(0,1)) + \theta U(-X, X)
\end{aligned}
$$

$N(\mu, \sigma^2)$ is the normal distribution with density $\rho_{\mu,\sigma^2}(x) = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}$.

Bare bones with jumps: derivation of α_c

Replace $N-1$ particles by a mean field p. Considering updates from g in the top diagram, either g moves and δ increases or p moves and δ decreases.



A 'no-collapse' condition is $\mathbb{E}\delta(t+1) = \delta(t)$.

Bare bones with jumps: derivation of α_c

Assuming that if x falls to the left of g then g will certainly move, and if x falls to the right then p will certainly move yields

$$\begin{aligned}\mathbb{E}\delta &= \int_{-\infty}^{0} (|x| + \delta)\rho_{g,\sigma^2} dx + \int_{0}^{\infty} x\rho_{g,\sigma^2} dx \\ &= \delta\end{aligned}$$

which is easily solved to give $\alpha_c = 0.63$. A more refined calculation taking into account the lower diagram gives $\alpha_c = 0.65$.

Bare bones with jumps: parameter values

α_c is the minimum scaling to ensure no collapse, and corresponds to the fastest convergence.

Experiments in $30D$ hint that $\alpha = \alpha_c$ is optimal, and indeed that a global focus neighbourhood ($BEST(p_i \in N) = g$) and a local, ring, spread neighbourhood ($DIFF(p_i, p_j \in M) = |p_{i_1} - p_{i+1}|$) are the preferred configurations, and that the jump probability should be set to 0.01.

3 Easy functions

Objective functions $f : [-X, X]^n \to \mathbb{R}$. The aim is to find the global minimum f^* of f.

f is optimised by $\{x^* : f^* = f(x^*)\}$.

We are interested in non-convex, multi-modal optimisation.

Heuristic algorithms such as PSO and GA attempt to find an approximate and adequate solution.

Easy functions

PSO does not depend on gradient information. This means that it optimises equivalence classes of functions,

$f \sim g$ if $f(x) < f(y) \Rightarrow g(x) < g(y)$.

The performance on g will be identical to that on f (given the same initial configuration and sequence of pseudo random numbers). This is a small class.

We might expect that PSO performs well on separable functions due to its component-by-component update.

Easy functions: separability

A function is separable if

$$x_i^* = \arg\min_{x_i} f(x_1, x_2, \ldots x_n).$$

Separability is important because an optimiser could optimise in any subspace, holding the rest of x fixed.

Separable functions are easy?

Easy functions: separability

However, separable functions are rare. (Take a separable function, dilate if necessary to break any spherical symmetry, and rotate by any angle.)

Although many of the popular benchmarks are (surprisingly) separable, heterogeneous problems (problems that are the direct product of subproblems, for example a composition of a continuous and a discrete subproblem) are separable in the respective subspaces.

Easy functions: the length of a vector in various spaces

Consider the family of functions $||\cdot||_p : [-1,1]^n \to \mathbb{R},\ p \geq 0$ that measure the separation of x from O. These are the L^p norms, $p > 0$:

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

(Added 23 April.) The triangle inequality does not hold for $0 < p < 1$ so the L^p norm is defined only for $p \geq 1$. However for $0 < p < 1$ we get a concave function - see later empirical results.)

Easy functions: the length of a vector in various spaces

The L^1 norm is the Manhattan distance, and the L^2 is the Euclidean distance.

$||x||_2^2$ is also known as the Sphere function.

$||x||_\infty$ is Shwefel's 2.21 function, the 'Box' function,

$$f(x) = ||x||_\infty = \sup |x_i| : i = 1 \dots n, x \in [-1, 1]$$

The $f = ||\cdot||_p$ family is unimodal and, apart from $||x||_\infty$, it is separable.

(Added 16/4/13.) Notice that f is convex for $p \geq 1$. Convex functions are, in principle, optimised efficiently (ref?). In particular, if a convex function has a local minimum then this minimum must be a global minimum.

However f is concave (but still unimodal) for $p < 1$.

Easy functions: the length of a vector in various spaces

We shall use easy functions in the following to gain insight into high dimensional optimisation and to theoretically investigate algorithm behaviour. They should also be included in any benchmark trial since not all algorithms find them that easy!

4 Surprises in high dimensions

Where the initial points land

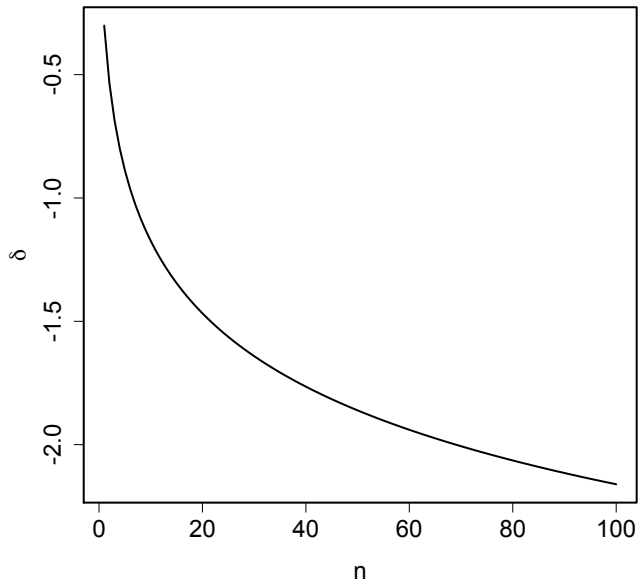
Uniform random initialisation in $[-1, 1]^n$. Consider the inner box $[-1 + \delta, 1 - \delta]^n$ that has half the available volume of 2^n.

$$\begin{aligned} vol([-1+\delta, 1-\delta]^n) &= 2^n(1-\delta)^n \\ &= 2^{n-1} \\ \Rightarrow \delta &= 1 - 2^{-1/n} \end{aligned}$$

In particular $\delta(30) \approx 0.023$, $\delta(100) \approx 0.0069$, $\delta(1000) = 0.00069$

Where the initial points land: box initialisation

Half of the points are within δ of a face. (The vertical axis in plot below is $\log_{10}(\delta)$.)



Where the initial points land: more general search volume

If $vol \sim cr^n$ for some constant c and length scale r then consider the volume of the shape of dimension $r - \epsilon$ that has half the volume:

$$\begin{aligned} vol(r-\epsilon) &= cr^n(1-\epsilon/r)^n \\ &= \frac{1}{2}cr^n \\ \Rightarrow \epsilon/r &= 1 - 2^{-1/n}. \end{aligned}$$

Where the initial points land: even more general search volume

Consider any compact set C. By the isoperimetric principle, it has surface area at least the surface area of a ball with the same volume. An ϵ extension that doubles the volume of C will have a smaller ϵ than for the corresponding ball. For the unit ball B, $vol(B) = c_n$, and the vol of its ϵ extension is $c_n(1+\epsilon)^n (= 2c_n)$.

Hence $\epsilon' \leq 2^{1/n} - 1$ for any compact search volume.

Where the initial points land: using the Brunn-Minkowski inequality

A quantitative expression of the isoperimetric principle is the Brunn-Minkowski inequality

$$vol(C+D)^{1/n} \geq vol(C)^{1/n} + vol(D)^{1/n}$$

where the Minkowski sum of two sets is $C + D = \{c + d : c \in C, d \in D\}$. The dilatate rC is defined $\{rc : c \in C\}$.

Where the initial points land: using the Brunn-Minkowski inequality

The same argument then proceeds via the BM inequality: consider when $vol(C) = vol(B)(= c_n)$ and extend C by adding balls ϵB to the surface (centre of the ball is at each point $c \in \partial C$). Then

$$\begin{aligned} vol(C + \epsilon B)^{1/n} &\geq vol(C)^{1/n} + \epsilon\ vol(B)^{1/n} \\ \Rightarrow vol(C + \epsilon B) &\geq (1 + \epsilon)^n vol(B) \\ \Rightarrow 2vol(C) &\geq (1 + \epsilon)^n vol(B) \\ 2^{1/n} - 1 &\geq \epsilon. \end{aligned}$$

Where the initial points land: intuition

Alternatively, consider the components of x: $x_1, x_2, \ldots x_n$. If one or more of the components is within δ of the edge then x will lie close to the surface.

The probability that none of the components is within δ of the edge (± 1) is $(2-2\delta)^n/2^n = (1-\delta)^n \to 0$ for any $1 \geq \delta > 0$ as $n \to \infty$.

Hence search points $(p_i) = (p_1, p_2, \ldots p_N)$ uniform randomly positioned in C will concentrate (as $n \to \infty$) on the surface!

Where the initial points land: summary

1. high dimensional spaces are poorly sampled by uniform distributions
2. search launched from uniformly distribute initial points will quite likely ($prob \approx \frac{1}{2}$) leave the search volume
3. 30 is a large number

(Added 16/4/13.) The estimation $prob \approx \frac{1}{2}$ is a consequence of the search spread at initialisation for any reasonable algorithm will of the order of the size of the box i.e. something like 2.

Effective search volume is negligible

Take one of the initial points g, and launch a search with g as a focus and with search spread determined by the separation of initial points (p_i).

Assume a normal search and generate trial points

$$x \sim N(\mu, \sigma^2)$$

where $\mu = g$ and $\sigma = \sigma(|p_i - p_j|)$.

Effective search volume is negligible

Actually, in BBJ, $\sigma = \alpha|p_{i-1} - p_{i+1}|$, and the trial position is formed component by component, $x_d \sim N(g_d, \alpha|p_{i-1,d} - p_{i_1,d}|)$. In this case set σ to $\max(\sigma_d)$.

For two points x, y uniform randomly in $[-1, 1]^n$ then $E(|x_d - y_d|) = 2/3$.

Effective search volume is negligible

99.7% of the trial points will lie in a ball of radius $3\sigma = 6/3 = 2$. Since g is very likely at the boundary, half of these will land inside $[-1, 1]^n$. They will land in a volume

$$\frac{1}{2} c_n 2^2 = \frac{1}{2} \left(\frac{\pi^{n/2}}{\Gamma(n/2+1)} \right) 2^2$$

or,

$$\begin{aligned} \frac{\text{eff search vol}}{\text{vol of search space}} &= \frac{1}{2} \left(\frac{\pi^{n/2}}{\Gamma(n/2+1)} \right) \\ &\to \frac{1}{2} \left(\frac{2\pi e}{n} \right)^{n/2} \\ &\approx \frac{1}{2} \left(\frac{17}{n} \right)^{n/2} \end{aligned}$$

Effective search volume is negligible

As an example, $\frac{\text{eff search vol}}{\text{vol of search space}} \approx 1.7 \times 10^{-39}$ $(n = 100)$.

At $n = 30$ we find $\frac{\text{eff search vol}}{\text{vol of search space}} \approx 10^{-5}$ (without Stirling's approx).

30 is a large number and 100 is impossibly big.

Difficulty of finding a better value: Box function

$f(x) = ||x||_\infty = \max(|x_1|, |x_2|, \ldots, |x_n|)$, $x \in \mathbb{R}^n \cap [-1, 1]^n$.

Suppose that $\max_d |x_d| = |x_1|$ and $x_1 > 0$.

$x_1 \sim U(-1, 1), x_2 \sim U(-1, 1), \ldots$

The update probability P_u is

$$P_u = \left(\frac{2|x_1|}{2}\right)^n = |x_1|^n.$$

Difficulty of finding a better value: Box function

We can estimate x_1 from the expectation of $\max(x_1, x_2, \ldots x_n)$ where $x_i \sim U(-1, 1)$.

This can be calculated from the probability density $\frac{dx}{2}$, or just consider distributing n points evenly along $[-1, 1]$ i.e. $(n+1)\Delta = 2 \Rightarrow \max|n| = 1 - \Delta = \frac{n-1}{n+1}$. So

$$P_u = \left(\frac{n-1}{n+1}\right)^n.$$

An initial update is increasingly certain as $n \to \infty$.

But what if $x_1 = 0.1$ at some later stage of the optimisation? Then $P_u = 0.1^n$. Even at $n = 30$, the update probability for uniform search is 10^{-30}.

Difficulty of finding a better value: Box function, normal search

Search is launched from g and suppose that $f(g) = g_1$. Then the update probability P_u is

$$P_u = \int_{-g_1}^{g_1} e^{-\frac{1}{2}\left(\frac{x-g_1}{\sigma_1}\right)^2} \frac{dx}{\sqrt{2\pi\sigma_1^2}} \prod_{d=2}^{d=n} \int_{-g_1}^{g_1} e^{-\frac{1}{2}\left(\frac{x-g_d}{\sigma_d}\right)^2} \frac{dx}{\sqrt{2\pi\sigma_d^2}}$$

An upper estimate can be made by setting $g_{d>1} = 0$ and $\sigma_i = \max(\sigma_1, \sigma_2, \ldots \sigma_n)$:

$$P_u \leq \int_{-g_1}^{g_1} e^{-\frac{1}{2}\left(\frac{x-g_1}{\sigma}\right)^2} \frac{dx}{\sqrt{2\pi\sigma^2}} \left(\int_{-g_1}^{g_1} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \frac{dx}{\sqrt{2\pi\sigma^2}} \right)^{n-1}.$$

Difficulty of finding a better value: Box function, normal search

Once more suppose that $\max_d |x_d| = g_1 = \frac{n-1}{n-2}$.

Furthermore, we suppose that $\sigma = 2/3$.

$$\begin{aligned} P_u &\leq \int_{-(\frac{n-1}{n+1})}^{\frac{n-1}{n+1}} \rho_{\frac{n-1}{n+1},\frac{4}{9}}(x)dx \left(\int_{-(\frac{n-1}{n+1})}^{\frac{n-1}{n+1}} \rho_{0,\frac{4}{9}}(x)dx \right)^{n-1} \\ &= \int_{-2(\frac{n-1}{n+1})}^{0} \rho_{0,\frac{4}{9}}(x)dx \left(\int_{-(\frac{n-1}{n+1})}^{\frac{n-1}{n+1}} \rho_{0,\frac{4}{9}}(x)dx \right)^{n-1} \\ &< \int_{-2}^{0} \rho_{0,\frac{4}{9}}(x)dx \left(\int_{-1}^{1} \rho_{0,\frac{4}{9}}(x)dx \right)^{n-1} \\ &\approx 0.50 * (0.87)^{n-1} \end{aligned}$$

Clearly P_u vanishes exponentially in n. For example $P_u(100) \approx 3.4 \times 10^{-7}, P_u(1000) \approx 3.0 \times 10^{-63}$.

Difficulty of finding a better value: Box function, normal search

The difference in initial P_u between normal search and uniform search is because normal search concentrates in an n-ball of radius less than half the diameter of the search volume.

Consider a ball of radius 1, just fitting inside the box$[-1,1]^n$. The volume of the ball is $c_n = \frac{\pi^{n/1}}{\Gamma(\frac{n}{2}+1)}$. Using the large n approximation $\Gamma(n) \approx (\frac{n}{e})^n$ then

$$\frac{\text{vol ball}}{\text{vol of box}} \to \left(\frac{\pi e}{2n}\right)^{n/2} \approx \left(\frac{4.27}{n}\right)^{n/2} \approx 3 \times 10^{-69} \ (n = 100).$$

Difficulty of finding a better value: square root of sphere function

$f(x) = ||x||_2 = \left(|x_1|^2 + |x|^2 + \ldots + |x_n|^2\right)^{1/2}$, $x \in \mathbb{R}^n \cap [0,1]^n$.

Uniform search

A random point in the search space will lie on a level set - the $n-1$-sphere, or surface of the $n-$ball. Enclose the ball by a box of length twice the radius of the ball. The update probability is then at least the ratio of these volumes, $\left(\frac{\pi e}{2n}\right)^{n/2}$ $(n \to \infty)$.

Difficulty of finding a better value: square root of sphere function, normal search

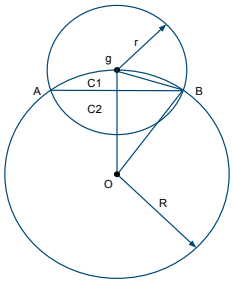


Figure 1: Diagram showing an update volume composed of two hyperspherical caps, C1 and C2. C1 has radius R, base AB and polar angle $\Phi = \cos^{-1}\left(1 - \frac{r^2}{2R^2}\right)$. C2 shares base AB with C1 and has radius r and polar angle $\phi = \cos^{-1}\left(\frac{r}{2R}\right)$. In $\triangle gOB$, $\phi = \angle OgB$ and $\Phi = \angle gOB$. This diagram depicts the situation for $0 \leq r \leq \sqrt{2}R$. As r grows from 0, cap C1 increases in size, reaching a D-hemisphere at $r = \sqrt{2}R$.

Difficulty of finding a better value: square root of sphere function, normal search, technical stuff

Denote the volume of the n-ball $B_n(r)$ of radius r by $vol(B_n(r)) \equiv V_n^B(r)$ $(= c_n r^n)$.

The volume of an n-cap of polar angle ϕ and radius r is

$$\int_{r\cos\phi}^{r} V_{n-1}^B((r^2 - z^2)^{n/2})dz = \int_{\phi}^{0} V_{n-1}^B(r\sin\theta)(-r\sin\theta)d\theta$$
$$= rV_{n-1}^B(r)\int_0^{\phi}\sin^n\theta d\theta$$

Useful identity:

$$\int_0^{\phi}\sin^n\theta d\theta \equiv J_n(\phi) = 2B_{\sin^2\phi}(\frac{n+1}{2},\frac{1}{2})$$

where $B_z(a,b)$ is the incomplete beta function, defined as $B_z(a,b) = \int_0^z t^{a-1}(1-t)^{b-1}dt$ and $B_1(a,b) \equiv B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$ is the complete beta function.

Difficulty of finding a better value: square root of sphere function, normal search, technical stuff

The update probability is

$$\begin{aligned} P_u &= \frac{vol(\text{cap of angle } \phi) + vol(\text{cap of angle } \Phi)}{vol(\text{ball of radius } r)} \\ &= \frac{rV_{n-1}^B(r)J_n(\phi) + RV_{n-1}^B(R)J_n(\Phi)}{2rV_{n-1}^B(r)J_n(\pi/2)} \\ &= \frac{J_n(\phi) + \left(\frac{R}{r}\right)^n J_n(\Phi)}{2J_n(\pi/2)} \end{aligned}$$

Difficulty of finding a better value: square root of sphere function, normal search, technical stuff

Simple geometry gives the relations $\cos\Phi = 1 - \frac{r^2}{2R^2}, \cos\phi = \frac{r}{2R}, \frac{R}{r} = \frac{\sin\phi}{\sin\Phi}$ and some analysis provides

$$\lim_{a\to\infty} B_z(a,b) = \frac{z^a(1-z)^{b-1}}{a} + O(\frac{1}{a^2})$$
$$\lim_{a\to\infty} B(a,b) = a^{-b}\Gamma(b).$$

(The first can be derived with integration by parts, or, an asymptotic expansion in $1/a$ can be derived by a Laplace transform. The expansion is valid for small $z = \sin^2\phi$ - a different expansion is needed for $\phi \to \pi/2$, the small r limit. The second follow from Stirling's approximation $\Gamma(z) \sim (z/e)^z$ and $\lim_{n\to\infty}(1+x/n)^n = e^x$.)

Difficulty of finding a better value: square root of sphere function, normal search

$$\lim_{D\to\infty} P_u = \frac{\sin^D \phi(\tan\Phi + \tan\phi)}{\sqrt{2\pi D}}.$$

Since $0 < \sin\phi < 1$ then $P_u \to 0$ as $D \to \infty$.

Difficulty of finding a better value: square root of sphere function, normal search, simpler

Or, enclose the lens by an n-ball of radius $r \sin \phi$. Then $P_u < vol(B_n(r \sin \phi))/vol(B_n(r)) = \sin^n \phi$. In fact the result follows by enclosing the lens in any ball of radius less than r.

Difficulty of finding a better value: square root of sphere function, normal search

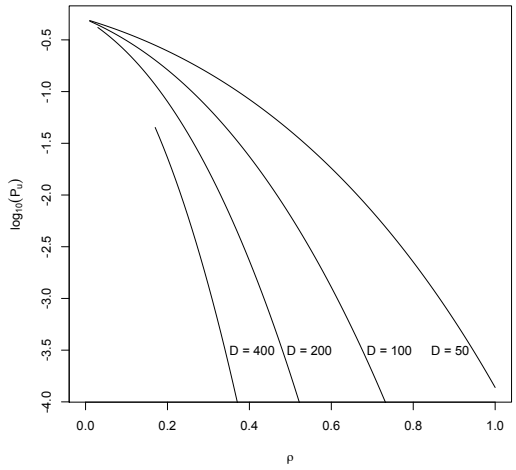


Figure 2: Plot showing variation of update probability P_u as a function of $\rho = r/R$ for various dimensions $D = 50, 100, 200, 400$

Difficulty of finding a better value: square root of sphere function, normal search

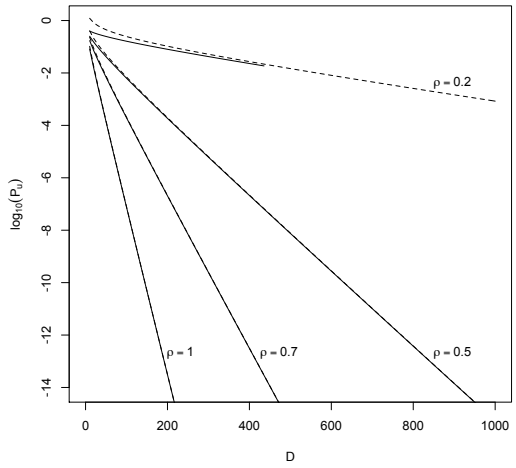


Figure 3: Update probability P_u as a function of D for $\rho = 0.2,\ 0.5,\ 0.7$. The full line is the exact value, and the dashed line shows the large-D asymptotic expression for P_u.

Strange fact concerning standing on an n-ball

Another way to look at this result is to imagine yourself standing on the surface of an n-ball. Then the volume below your feet is exponentially small compared to the volume of space around you. The surface curves so rapidly, the surface appears as a needle and you are standing on the tip.

But how do n-balls appear to microbes?

Strange fact concerning a microbe on an n-ball

Consider fixed n but decreasing r.

From the small z approximations

$$\begin{aligned} B_z(a,b) &\sim \frac{z^a}{a}\left(1+\frac{a(1-b)}{a+1}z+O(z^2)\right) \\ B_{1-z}(a,b) &= B(a,b)-B_z(b,a) \\ &\sim B(a,b)-\frac{z^b}{b}\left(1+\frac{b(1-a)}{b+1}z+O(z^2)\right) \end{aligned}$$

we find

$$\lim_{r\to 0}(R/r)^n B_{sin^2\Phi}(a,b)=\frac{2\rho}{n+1}-\frac{(n-1)\rho^3}{4(n+3)}+O(\rho^5)$$

and

$$\lim_{r\to 0} B_{sin^2\phi}(a,b)=B(a,b)-2\rho+\frac{(n-1)\rho^3}{3}+O(\rho^5)$$

where $\rho=r/R$.

Strange fact concerning a microbe on an n-ball

Combining terms we have

$$\lim_{r\to 0} P_u = \frac{B(a,b) - (\frac{2n}{n+1})\rho + \frac{(n-1)(4n+9)}{12(n+3)}\rho^3}{2B(a,b)} \to 1/2,$$

which demonstraties that $P_u \to 1/2$ as $r \to 0$ at fixed n.

The conclusion is that an update from any position is exponentially unlikely for $||x||_2$, unless the search spread is exponentially tiny, in which case progress is exponentially slow.

Hopelessness of high dimensional search

But the result generalises. Suppose we search from a point p and better regions are enclosed by n-balls. Update will be very unlikely as long as the total volume of all the n-balls is less than the search volume.

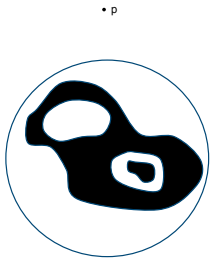


Figure 4: The search volume is focused on a point $\boldsymbol{p}$. Black regions depict volumes with lower function value then $f(\boldsymbol{g})$. The update volume is enclosed by a bounding sphere. The update probability is therefore bounded by the probability that a trial position lies inside the bounding sphere. The analysis for update within a spherical fitness volume applies and again we deduce that $P_u \to 0$ as $n \to \infty$.

In the light of all this gloominess, is high dimensional optimisation hopeless?

Yes.

But we can optimise in low dimensions (1, 2, 3, and 4, perhaps 10) and high dimensional spaces have low dimensional subspaces, so perhaps we can proceed by low dimensional optimisation.

5 Subspace optimisation

The idea is to break H into K subspaces $\oplus_1^K H_k$ of dimension n_k each ($\sum n_k = n$). Then try and optimise $x_{1:n}$ in each subspace separately whilst holding components in in the other subspaces fixed.

For example, spilt x into $x_{1:k}x_{k+1:n}$ and hold $x_{k+1:n}$ fixed whilst seeking trials $y_{1:k}$ such that $y_{1:n_k}x_{k+1,n}$ provides a better solution than x.

The venerable line search, a traditional algorithm well described in Numerical Recipes follows this procedure with $K = n,\ n_k = 1$.

However we are interested in optimisation by a population. What subspace dimensionality is preferred?

Subspace optimisation: $||x||_\infty$

x contains a single component, x_m that has to be reduced, and this occurs in just one subspace.

So all subspace trials that are not in x_m's subspace produce no improvement.

Suppose that a trial in the subspace containing m manages an update with probability p^{n_k} i.e. each component has a probability of p of producing a component $y_i : |y_i| < |x_m|$.

Then the probability of update can be estimated

$$P_u \sim \frac{n_k}{n} p^{n_k}.$$

Optimising P_u in n_k gives $n_k = log(\frac{1}{p})$. One dimensional subspaces are preferred for $p > 0.22$. For smaller probabilities, larger subspaces are optimal.

Subspace optimisation: $||x||_2$

Now look at $||x||_2$ and search using normal sampling, $y_i \sim N(x_i, \sigma_i)$.

Update will happen if (choosing subspace $k = 1$ for simplicity), $y_1^2 + y_2^2 + \ldots + y_{n_1}^2 < x_1^2 + x_2^2 + \ldots + x_{n_1}^2$.

The statistic $Y = \sum_1^{n_1} y_i/\sigma_i$ follows the non-central chi squared distribution $\chi^2_{n_1,\lambda}$ where $\lambda = \sum(x_i/\sigma_i)^2$.

The CDF $P(y; n_1, \lambda)$ is an infinite sum over complete and incomplete gamma functions.

Subspace optimisation: $||x||_2$

In any case, we find

$$P_u = P(\lambda; n_1, \lambda)$$

which is a decreasing function of n_1. The optimal subspace dimension is therefore $n_k = 1$, which fits with the picture of the spherical symmetry of the normal distribution at constant σ_i in each dimension.

The volume enclosed by P_u will be an $n-$ball and the ratio of the volumes of the n-ball to the n-cube falls exponentially with n. We expect that n_k might be optimal.

Subspace optimisation: not yet the whole story

P_u is not the whole story.

We should also consider the rate of progress $\mathbb{E}P_u\delta x$.

A high P_u but small subspace dimensionality has to be balanced with the need (in population algorithms) to give each individual an update. One iteration through the population requires NK evaluations, which rises to Nn for one dimensional subspaces.

It might be better to tolerate a smaller success rate per individual in order to conserve evaluations and allow information to flow through the population i.e. increase the number of iterations. (We are assumed that the constrained resource is total number of function evaluations.)

6 Subspace optimisation with a population

Line search - $n_k = 1$ and $N = 1$.

The extension to $N, n_k > 1$ yields many possibilities, and also dangers.

In cooperative schemes, the subpopulations optimise in their respective subspaces, but the fitness is determined by an evaluation across the subpopulations.

We might expect subspace search to be a successful strategy for separable functions if the optimisation subspaces align with the separation subspaces.

Dangers of subspace optimisation: trapping optima

Subspace optimisation suffers from trapping optima. These are optima that the algorithm can never leave, even if all points in each subspace are evaluated! This can happen at a saddle point, but also in a basin of attraction of a secondary optimum.

Dangers of subspace optimisation: trapping optima

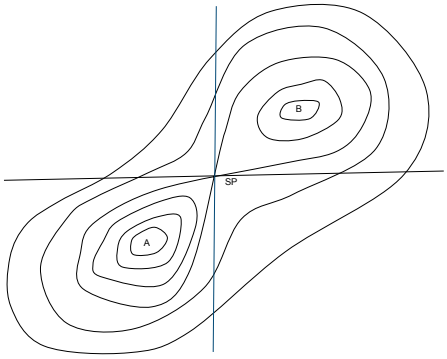


Figure 5: Contour plot of a two dimensional function with global minimum at A, a local minimum at B and a saddle point at SP.

Dangers of subspace optimisation: trapping optima

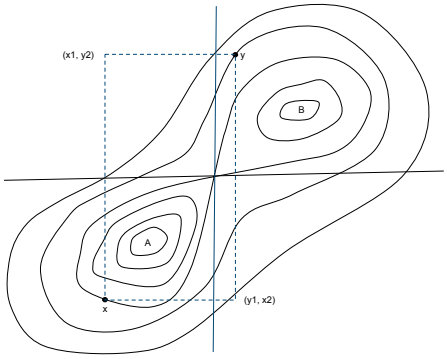


Figure 6: Trial position $\boldsymbol{x} = (x_1, x_2)$ is rejected when evaluated in subspaces 1 and 2. The contect vector is $\boldsymbol{y} = (y_1, y_2)$ and the function value contours decrease towards the two minima at A and B. The points $c_1(\boldsymbol{y}, \boldsymbol{x}) = (x_1, y_2)$ and $c_2(\boldsymbol{y}, \boldsymbol{x}) = (y_1, x_2)$ have higher function values than the current best, $f(\boldsymbol{y})$.

Dangers of subspace optimisation: outdated memory

Cooperative schemes split a swarm into K subswarms. Each subspace has s components and $n = Ks$. A context vector $\hat{g}$ is constructed by concatenating the global bests g_k of each subswarm, $\hat{g} = g_1 g_2 \dots g_K$.

The problem is that the personal best of particle i in subspace k is evaluated in context $\hat{g}(t')$ and the global best of subswarm k is with respect to a context at possibly a different time, $\hat{g}(t'')$.

Situations can occur when a trial is rejected even if it is the best ever tested.

Dangers of subspace optimisation: outdated memory

This is the problem of outdated memory. Ideally, whenever the context changes, all values of personal bests should be re-evaluated. Exactly the same problem occurs in dynamic optimisation when a shift in the global optimum renders all stored values inaccurate at best, and at worst, misleading.

However scrupulous memory updating in new contexts is very expensive in terms of the allocated total function evaluations available.

The solution in CBBJ is to decouple the context vector from the global bests in each subswarm. Every trial is compared to the context vector, and the context vector is updated irrespective of what happens to the personal and global bests.

Possible dangers of subspace optimisation: random groupings

Various authors have proposed random subspace groupings. The idea is to counteract component interaction.

A firm definition of component interaction is not to be found in the literature but presumably is one of two types:

1. f is separable into two or more subspaces, but this separation is not known to the algorithm.

2. During the course of optimisation, a position x might be reached where movement in a particular subspace H_k is beneficial e.g. moves x away from a local optimum or causes a large reduction of function value (for example if $x_{i,j,k...}$ have particularly high values and $x_{i,j,k...}$ occur in f with a positive power.) H_k is not a separable subspace of f and might not consist of contiguous components or be a subspace in the original grouping.

Possible dangers of subspace optimisation: random groupings

Subspace shuffling and subspace size might occur according to a fixed scheme, or the subspaces might be chosen at randomly whenever shuffling is implemented.

In CCPSO2 (Li and Yao TEC (2012) 16:210-224), if no improvement occurs after a complete iteration then a new subspace size s is chosen from a number of preset alternatives.

This yields a high rate of re-grouping, higher than in prior cooperative PSO's. The idea is that the system has a higher chance of capturing interacting variables.

Possible dangers of subspace optimisation: random groupings

It might be argued, however, that each regrouping disrupts the swarm. A PSO typically needs several iterations for effective information transfer. The swarm functions as a complex system and its behaviour is not manifest on short time scales. Regrouping places particles in completely different subswarms.

Cooperative BBJ

Rather than think of subswarms, the cooperative algorithm is perhaps easier to conceive as a single swarm with each particle searching in each of the subspaces.

In the following CBBJ algorithm, the swarm update loop, the one that moves the particles, is identical to the BBJ update rule.

Cooperation enters at the evaluation stage where a particle's personal best p_i has K values f_{ki}, each associated with a different subspace.

The swarm best g also has K values.

There is a single context vector $\hat{g}$ with a unique value, representing the best value ever fund by the swarm.

$i = 1 \ldots N,\ k = 1 \ldots K,\ d = 1 \ldots n$

$p_i = \oplus p_{ik},\ g = \oplus g_k = \oplus c_k$

$c_k(x, y)$ injects subspace component of y into x

$\forall i$

$x_{id} \sim N(g_d, \alpha|p_{(i-1)d} - p_{(i+1)d}|)$

$\theta = (u \sim U(0,1) < pJ)\ ?\ 1 : 0$

$x_i = (1 - \theta)(g + \alpha|p_{(i-1)d} - p_{(i+1)d}|N(0,1)) + \theta U(-X, X)$

$\forall k\ \forall i$

$f_x = f(c_k(\hat{g}, x_i))$

`if` $(f_x < f_{ki})$ `// update personal best`

$p_i \leftarrow c_k(p_i, x_i)$

$f_{ki} \leftarrow f_x$

`if` $f_x < f_k$ `// update swarm best`

$g = c_k(g, p_i)$

$f_k = f_x$

`if` $(f_x < f_{\hat{g}})$ `// update context`

$\hat{g} \leftarrow c_k(\hat{g}, x_i)$

$f_{\hat{g}} = f_x$

7 Empirical investigation of subspace dimensionality

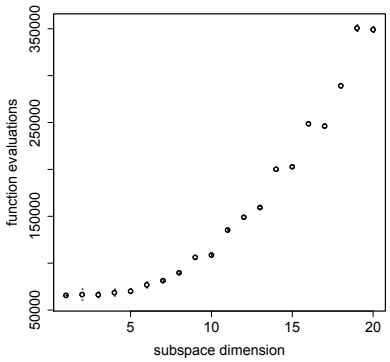


Figure 7: Mean evaluations need to optimise 2008.1, shfted sphere, in 100D, by sgl-BBJ. All parameters, error bars, runs etc as for Figure 12. The steps that are superimposed on the curve are possibly due to the uneven subdivision into subspaces when $D \ \% \ D_k \neq 0$ e.g. at $D_k = 19$, the algorithm will chose $5 \times 19 + 1 \times 4$ and at $D_k = 5$ the algorithm will chose $5 \ \times \ 20$.

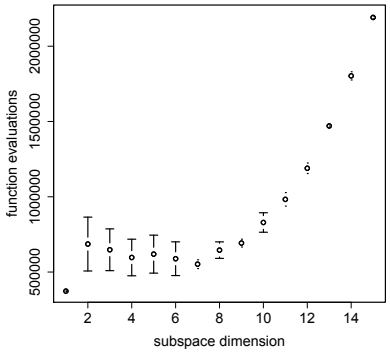


Figure 8: Mean evaluations need to optimise 2008.1, shfted sphere, in 500D, by sgl-BBJ. All parameters, error bars, runs etc as for Figure 12. The function was not optimised within 5000D evaluations for any $D_k > 15$.

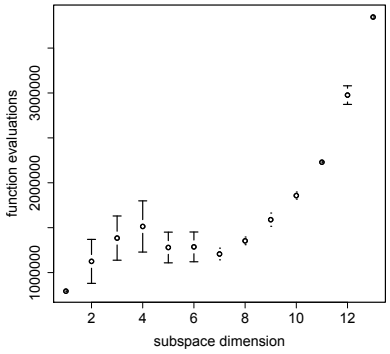


Figure 9: Mean evaluations need to optimise 2008.1, shfted sphere, in 1000D, by sgl-BBJ. All parameters, error bars, runs etc as for Figure 12. The function was not optimised within 5000D evaluations for any $D_k > 13$.

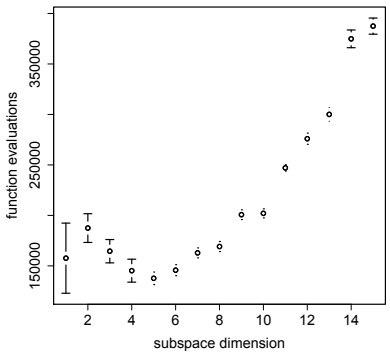


Figure 10: Mean evaluations need to optimise 2008.4, shfted Rastrigin, in 100D, by sgl-BBJ. All parameters, error bars, runs etc as for Figure 12. The function was not optimised within 5000D evaluations on some or all runs for any $D_k > 15$.

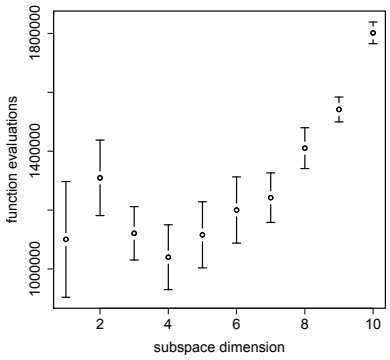


Figure 11: Mean evaluations need to optimise 2008.4, shfted Rastrigin, in 500D, by sgl-BBJ. All parameters, error bars, runs etc as for Figure 12. The function was not optimised within 5000D evaluations on some or all runs for any $D_k > 10$.

$s = 1$ can indeed offer the best answer, and especially for spherically symmetric functions, but it has a large margin of error for a multimodal function. $s = 4$ is a good choice for the latter.

8 Scalability of CBBK

Evidence that cooperation breaks the curse of dimensionality and offers a scalable algorithm.

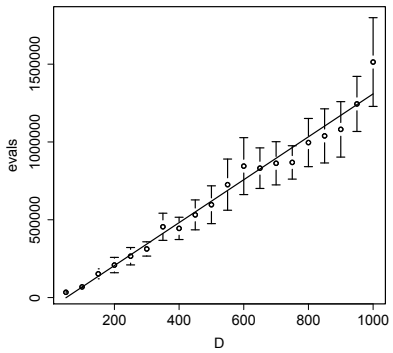


Figure 12: Plot of mean evaluations needed to achieve an error of $1e-8$ for sgl-BBJ optimisation of CEC 2008 shifted sphere as a function of dimensionality D, for $D \in [50, 1000]$. The error bars show 95% confidence limits. Each point represents 50 runs. $\{N, D_k, \alpha, p_J\} = \{40, 4, 0.65, 0.1\}$, $\{I, S, T\} = \{uniform, nearest, 1e-8 \vee E = 5000D\}$, $\{f, D, V\} = \{2008.1, D \in [50, 1000], [-100, 100]^D\}$. The line shows the linear fit $E = mD + c, m = 1379.24 \pm 52.09, c = -70222.34 \pm 31200.29$ ($\pm$standard errors), $R^2 = 0.975$.

9 Experiments

The results in Table 1 show the importance of subspace optimisation for BBJ, and ability to solve separable functions in low dimensionalities.

Table 1: Optimisation of L_p in $D = 30$. All functions have shifted optimum locations. The table reports mean number of evaluations needed to optimise each L^p function. () indicates standard err. Experiment methodology as for CEC 2008 except a termination condition of error less than $1e^{-5}$ was used. The case $p = 2/3$ has been included even though the function $||\cdot||_{2/3}$ is not a norm.

Fn	CBBJ	CKPSO	BBJ
$p = 2/3$ astroid	20563(76)	76108(600)	NEVER
$p = 1$ diamond	17080(70)	65243(300)	NEVER
$p = 2$ sphere	14491(76)	56621(234)	NEVER
$p = 10$ rounded box	14261(121)	55521(333)	NEVER

Table 2 compares CBBJ with state-of-the art PSO's in 30D.

CBBJ doesn't appear to suffer in non-separable problems despite the lack of subspace shuffling.

Table 2: CEC 2005. $D = 30$. *,† indicate a rotated/non-separable function. All functions have shifted optimum locations and non-zero biases. Terminated at 300000 evals (min error = 0). The function groupings are unimodal, multimodal and 'expanded' ($f(x) = \sum f(x_i, x_{i+1})$). CLPSO is the comprehensive learning PSO, DMPSO is the dynamic multiswarm agorithm, UPSO is a 'unified' PSO and FIPS is the fully informed PSO.

Fn		CBBJ	CLPSO	DMSPSO	UPSO	FIPS
f_1	sph	1.41(0.37)E-13	0.00(0)E+00	3.14(4.15)E+02	1.31(0.73)E+03	5.25(5.57)E+02
f_2	sch12†	7.96(11.54)E+00	3.83(1.06)E+02	7.80(0.21)E+02	7.60(5.29)E+03	1.47(0.23)E+04
f_3	ell†	1.06(0.41)E+06	1.19(0.31)E+07	5.62(6.23)E+06	5.30(3.86)E+07	1.95(1.11)E+07
f_4	sch12n†	8.91(5.65)E+03	5.40(1.25)E+03	8.56(12.9)E+02	1.88(0.61)E+04	2.07(0.31)E+04
f_5	sch26†	1.27(0.32)E+04	4.00(0.43)E+03	4.26(1.87)E+03	1.28(0.23)E+04	1.17(0.14)E+04
f_6	ros†	7.91(6.33)E+01	1.78(2.29)E+01	2.72(7.29)E+07	1.19(1.36)E+07	2.46(3.49)E+07
f_7	gri†*	4.70(6.95E-13)E+03	4.70(0)E+03	4.34(0.22)E+03	7.52(0.34)E+03	7.48(0.22)E+03
f_8	ack†*	2.09(0.07)E+01	2.07(0)E+01	2.09(0)E+01	2.10(0)E+01	2.09(0)E+01
f_9	ras	9.55(3.92)E-14	0.00(0)E+00	4.85(1.51)E+01	7.84(1.69)E+01	5.40(1.10)E+01
f_{10}	ras†*	2.25(0.43)E+02	8.02(1.50)E+01	8.00(2.00)E+01	1.59(0.55)E+02	1.53(0.25)E+02
f_{11}	wei†*	2.95(0.41)E+01	2.53(0.19)E+01	2.90(0.23)E+01	3.14(0.47)E+01	2.69(0.26)E+01
f_{12}	sch213†	2.90(3.01)E+03	1.32(0.42)E+04	7.84(6.84)E+04	8.98(5.43)E+04	5.19(3.21)E+04
f_{13}	gri+ros†	9.18(2.48)E-01	1.89(0.40)E+00	1.13(0.56)E+01	9.23(4.56)E+00	9.64(1.73)E+00
f_{14}	sch6†*	1.287320(0.07)E+01	1.25(0.03)E+01	1.21(0.07)E+01	1.28(0.04)E+01	1.23(0.03)E+01

Table 3 shows the high dimension results.

Table 3: CEC 2008. The functions are grouped into uni/multi-modal classes. † indicates non-separability. All functions have shifted optima and non zero biases.

Fn	Dim	CBBJ (std error)	CCPSO2	sep-CMA-ES
f_1	100	5.59e-13 (2.54e-14)	7.73E-14 (3.23E-14)	9.02E-15 (5.53E-15)
sph	500	2.93e-12 (7.37e-14)	3.00E-13 (7.96E-14)	2.25E-14 (6.10E-15)
	1000	6.13e-12 (9.38e-14)	5.18E-13 (9.61E-14)	7.81E-15 (1.52E-15)
f_2	100	4.64e+00 (8.68e-02)	6.08E+00 (7.83E+00)	2.31E+01 (1.39E+01)
sch†	500	2.13e+01 (1.995e-01)	5.79E+01 (4.21E+01)	2.12E+02 (1.74E+01)
(box)	1000	4.53e+01 (2.63e-01)	7.82E+01 (4.25E+01)	3.65E+02 (9.02E+00)
f_3	100	3.84e+02 (9.25e+01)	4.23E+02 (8.65E+02)	4.31E+00 (1.26E+01)
ros†	500	7.96e+02 (4.72e+01)	7.24E+02 (1.54E+02)	2.93E+02 (3.59E+01)
	1000	1.40e+03 (2.48e+01)	1.33E+03 (2.63E+02)	9.10E+02 (4.54E+01)
f_4	100	4.55e-13 (2.41e-14)	3.98E-02 (1.99E-01)	2.78E+02 (3.43E+01)
ras	500	3.69e-04 (3.69e-04)	3.98E-02 (1.99E-01)	2.18E+03 (1.51E+02)
	1000	3.98e-04 (3.98e-04)	1.99E-01 (4.06E-01)	5.31E+03 (2.48E+02)
f_5	100	2.11e-02 (7.94e-03)	3.45E-03 (4.88E-03)	2.96E-04 (1.48E-03)
gri†	500	5.74e-02 (2.14e-02)	1.18E-03 (4.61E-03)	7.88E-04 (2.82E-03)
	1000	1.31e-02 (5.67e-03)	1.18E-03 (3.27E-03)	3.94E-04 (1.97E-03)
f_6	100	6.40e-13 (1.49e-14	1.44E-13 (3.06E-14)	2.12E+01 (4.02E-01)
ack	500	3.46e-12 (2.68e-14)	5.34E-13 (8.61E-14)	2.15E+01 (3.10E-01)
	1000	6.94e-12 (5.08e-14)	1.02E-12 (1.68E-13)	2.15E+01 (3.19E-01)

10 Conclusions

Straight application of any search technique is doomed in high dimensional spaces.

A subspace scheme - such as cooperation - appears to break the curse of dimensionality.

But it has its dangers; trapping optima, outdated memory and the conflict between subspace shuffling and swarm destabilisation.

Cooperative BBJ is a simple extension of BBJ, simpler than other cooperative PSO schemes and simpler than other refined (and expensive) algorithms that adapt a covariance matrix.

It uses a shared context vector that is distinct from the subspace bests.

Now, back to SPECT...