Runtime Analysis of Binary PSO

Dirk Sudholt and Carsten Witt

Technische Universität Dortmund
Fakultät für Informatik, LS 2

Dagstuhl 2008

Particle Swarm Optimization

Particle Swarm Optimization

- Bio-inspired optimization principle developed by Kennedy and Eberhart (1995).
- Mostly applied in continuous spaces.
- Swarm of particles, each moving with its own velocity.
- Velocity is updated according to
  - own best position and
  - position of the best individual in its neighborhood.
- Here: neighborhood = the whole swarm.
- Behavior derived from social-psychology theory.

Binary PSO

Binary PSO

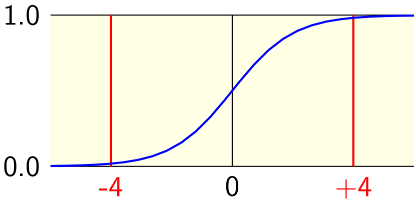
- Developed by Kennedy and Eberhart (1997).
- Goal: optimize pseudo-Boolean function $f\colon \{0,1\}^n \to \mathbb{R}$.
- Swarm contains μ particles.
- Record global best particle x^*.
- The i-th particle maintains triplet
  1. current position $x^{(i)} \in \{0,1\}^n$,
  2. own best position $x^{*(i)} \in \{0,1\}^n$, and
  3. a real-valued velocity $v^{(i)} \in \mathbb{R}$.

What is the meaning of velocity in binary spaces?

Creating New Positions

Probabilistic construction using velocity v and sigmoid function $s(v)$:

$$\mathrm{P}(x_j = 1) = s(v_j) = \frac{1}{1+e^{-v_j}}$$



Restrict velocities to $v_j \in [-v_{\max}, +v_{\max}]$.

- Common practice: $v_{\max} = 4$.
- Much better: $v_{\max} := \ln(n-1)$:

$$\frac{1}{n} \leq \mathrm{P}(x_j = 1) \leq 1 - \frac{1}{n}.$$

Updating Velocities

Update current velocity vector according to

- cognitive component → towards own best: $x^{*(i)} - x^{(i)}$ and
- social component → towards global best: $x^{*} - x^{(i)}$.

Learning rates c_1, c_2 affect weights for the two components.

Random scalars $r_1 \in U[0, c_1]$, $r_2 \in U[0, c_2]$ chosen anew in each generation:

$$v^{(i)} = v^{(i)} + r_1(x^{*(i)} - x^{(i)}) + r_2(x^{*} - x^{(i)})$$

The Whole Algorithm

Algorithm (Binary PSO)

1. *Initialize velocities with* 0^n *and all solutions with* $\perp$.
2. *Choose* $r_1 \in U[0, c_1]$ *and* $r_2 \in U[0, c_2]$.
3. *For* $j := 1$ *to* μ *and* $i := 1$ *to* n *do*
   Set $x_i^{(j)} := 1$ *with probability* $s(v_i^{(j)})$, *else* $x_i^{(j)} := 0$.
4. *For* $j := 1$ *to* μ *do*
   If $f(x^{(j)}) > f(x^{*(j)})$ *then* $x^{*(j)} := x^{(j)}$.
   If $f(x^{*(j)}) > f(x^*)$ *then* $x^* := x^{*(j)}$.
5. *For* $j := 1$ *to* μ *do*
   Set $v^{(j)} := v^{(j)} + r_1(x^{*(j)} - x^{(j)}) + r_2(x^* - x^{(j)})$.
   Restrict each component of $v^{(j)}$ *to* $[-v_{\max}, v_{\max}]$.
6. *Goto 2.*

The 1-PSO

Special case: 1-PSO with $\mu = 1$, $c_1 = 0$, and $c_2 = 2$.

Algorithm (1-PSO)

1. *Initialize* $v = 0^n$ *and* $x^* = \perp$.
2. *Choose* $r \in U[0, 2]$.
3. *For* $i := 1$ *to* n *do*
   Set $x_i := 1$ *with probability* $s(v_i)$*, else* $x_i := 0$.
4. *If* $f(x) > f(x^*)$ *then* $x^* := x$.
5. *Set* $v := v + r(x^* - x)$.
   Restrict each component of v *to* $[-v_{\max}, v_{\max}]$.
6. *Goto 2.*

Understanding Velocities

1-PSO: update increases velocity by $r(x^* - x)$.

Strange: velocity v_i is changed only if $x_i \neq x_i^*$.

Let $x_i^* = 1$, then probability to increase v_i is

$$1 - s(v_i) \;=\; s(-v_i) \;=\; \frac{1}{1 + e^{v_i}}.$$

$\Rightarrow$ at least $1/2$ for $v_i < 0$, but decreases rapidly with growing v_i.

Lower Bounds

Theorem

If f has a unique global optimum, $\mu = \mathrm{poly}(n)$, and $c_1 + c_2 = O(1)$, the expected number of generations of the Binary PSO on f is $\Omega(n/\log n)$.

Proof ideas:

Lower Bounds

Theorem

If f has a unique global optimum, $\mu = \mathrm{poly}(n)$, and $c_1 + c_2 = O(1)$, the expected number of generations of the Binary PSO on f is $\Omega(n/\log n)$.

Proof ideas:

- W. l. o. g. optimum is 1^n.

Lower Bounds

Theorem

If f has a unique global optimum, $\mu = \mathrm{poly}(n)$, and $c_1 + c_2 = O(1)$, the expected number of generations of the Binary PSO on f is $\Omega(n/\log n)$.

Proof ideas:

- W. l. o. g. optimum is 1^n.
- Consider period of $cn/\log n$ generations, c small.

Lower Bounds

Theorem

If f has a unique global optimum, $\mu = \mathrm{poly}(n)$, and $c_1 + c_2 = O(1)$, the expected number of generations of the Binary PSO on f is $\Omega(n/\log n)$.

Proof ideas:

- W. l. o. g. optimum is 1^n.
- Consider period of $cn/\log n$ generations, c small.
- Call a bit x_i weak if $\mathrm{P}(x_i = 1) \leq p := 1 - \frac{e\ln(\mu n)}{n}$.

Lower Bounds

Theorem

If f has a unique global optimum, $\mu = \mathrm{poly}(n)$, and $c_1 + c_2 = O(1)$, the expected number of generations of the Binary PSO on f is $\Omega(n/\log n)$.

Proof ideas:

- W. l. o. g. optimum is 1^n.
- Consider period of $cn/\log n$ generations, c small.
- Call a bit x_i weak if $\mathrm{P}(x_i = 1) \le p := 1 - \frac{e \ln(\mu n)}{n}$.
  Equivalent: $v_i \le s^{-1}(p)$.

Lower Bounds

Theorem

If f has a unique global optimum, $\mu = \mathrm{poly}(n)$, and $c_1 + c_2 = O(1)$, the expected number of generations of the Binary PSO on f is $\Omega(n/\log n)$.

Proof ideas:

- W. l. o. g. optimum is 1^n.
- Consider period of $cn/\log n$ generations, c small.
- Call a bit x_i weak if $\mathrm{P}(x_i = 1) \leq p := 1 - \frac{e\ln(\mu n)}{n}$.
  Equivalent: $v_i \leq s^{-1}(p)$.
- v_i-increase at $v_i \geq s^{-1}(p) - (c_1 + c_2)$ has prob. $O\left(\frac{\ln(\mu n)}{n}\right)$.

Lower Bounds

Theorem

If f has a unique global optimum, $\mu = \text{poly}(n)$, and $c_1 + c_2 = O(1)$, the expected number of generations of the Binary PSO on f is $\Omega(n/\log n)$.

Proof ideas:

- W. l. o. g. optimum is 1^n.
- Consider period of $cn/\log n$ generations, c small.
- Call a bit x_i weak if $\mathrm{P}(x_i = 1) \leq p := 1 - \frac{e \ln(\mu n)}{n}$.
  Equivalent: $v_i \leq s^{-1}(p)$.
- v_i-increase at $v_i \geq s^{-1}(p) - (c_1 + c_2)$ has prob. $O\left(\frac{\ln(\mu n)}{n}\right)$.
- Always n/e weak bits in all particles w. h. p.

Lower Bounds

Theorem

If f has a unique global optimum, $\mu = \mathrm{poly}(n)$, and $c_1 + c_2 = O(1)$, the expected number of generations of the Binary PSO on f is $\Omega(n/\log n)$.

Proof ideas:

- W. l. o. g. optimum is 1^n.
- Consider period of $cn/\log n$ generations, c small.
- Call a bit x_i weak if $\mathrm{P}(x_i = 1) \leq p := 1 - \frac{e \ln(\mu n)}{n}$.
  Equivalent: $v_i \leq s^{-1}(p)$.
- v_i-increase at $v_i \geq s^{-1}(p) - (c_1 + c_2)$ has prob. $O\left(\frac{\ln(\mu n)}{n}\right)$.
- Always n/e weak bits in all particles w. h. p.
- Probability to find optimum at most $p^{n/e} \leq 1/(\mu n)$ for single trial and $O(n/\log n \cdot \mu) \cdot 1/(\mu n) = o(1)$ for whole period.

Freezing Times

Unless x^* is exchanged, v_i freezes to $-v_{\max}$ (if $x_i^* = 0$) or $v_{\max}$ (if $x_i^* = 1$) sooner or later.

Lemma

Expected freezing time is $O(n)$ for single bits and $O(n \log n)$ for n or μn bits if $\mu = \mathrm{poly}(n)$.

Freezing Times

Unless x^* is exchanged, v_i freezes to $-v_{\max}$ (if $x_i^* = 0$) or $v_{\max}$ (if $x_i^* = 1$) sooner or later.

Lemma

Expected freezing time is $O(n)$ for single bits and $O(n \log n)$ for n or μn bits if $\mu = \text{poly}(n)$.

After freezing, 1-PSO temporarily behaves like the (1+1) EA*.

Algorithm ((1+1) EA*)

1. *Choose x^* uniformly at random.*
2. *Create x by flipping each bit in x^* with prob. $1/n$.*
3. *If $f(x) > f(x^*)$ then $x^* := x$.*
4. *Goto 2.*

Fitness Level Arguments

Let $f := \{0,1\}^n \to \{0,\ldots,m\}$ and s_i be the minimum probability of the (1+1) EA* to increase the fitness from i.

Upper bound for the (1+1) EA*

$$\sum_{i=0}^{m-1} \frac{1}{s_i}$$

Fitness Level Arguments

Let $f := \{0,1\}^n \to \{0,\dots,m\}$ and s_i be the minimum probability of the (1+1) EA* to increase the fitness from i.

Upper bound for the (1+1) EA*

$$\sum_{i=0}^{m-1} \frac{1}{s_i}$$

Upper bound for the 1-PSO

$$O(m \cdot n \log n) + \sum_{i=0}^{m-1} \frac{1}{s_i}$$

Fitness Level Arguments

Let $f := \{0,1\}^n \to \{0,\dots,m\}$ and s_i be the minimum probability of the (1+1) EA* to increase the fitness from i.

Upper bound for the (1+1) EA*

$$\sum_{i=0}^{m-1} \frac{1}{s_i}$$

Upper bound for the 1-PSO

$$O(m \cdot n \log n) + \sum_{i=0}^{m-1} \frac{1}{s_i}$$

Upper bound for generations of Binary PSO with $c_1 := 0, c_2 := 2$

$$O\left(m \cdot n \log n + \frac{1}{\mu} \sum_{i=0}^{m-1} \frac{1}{s_i}\right)$$

The 1-PSO on OneMax

Fitness level arguments only yield $O(n^2 \log n)$ for the 1-PSO on OneMax.

The 1-PSO on OneMax

Fitness level arguments only yield $O(n^2 \log n)$ for the 1-PSO on OneMax.

Theorem

The expected optimization time of the 1-PSO on OneMax *is* $O(n \log n)$.

The 1-PSO on OneMax

Fitness level arguments only yield $O(n^2 \log n)$ for the 1-PSO on OneMax.

Theorem

The expected optimization time of the 1-PSO on OneMax is $O(n \log n)$.

Let $P_t := \mathrm{P}(x_i = 1)$ after t generations with $x_i^* = 1$ and initial velocity $v_i = -v_{\max}$.

Definition

A bit is t-strong when the probability of setting it to 1 stochastically dominates P_t.

Distribution of Velocities

Lemma

Let $t \geq 16 \ln n$, $1 \leq i \leq t/96$ *and* n *be large enough.*

$$\mathrm{Prob}\left(P_t \geq 1 - \frac{96i}{t}\right) \;\geq\; 1 - e^{-i}.$$

1-PSO on OneMax

Temporarily assume a 1-bit is never reset to 0 in x^*.

Phase i ends when

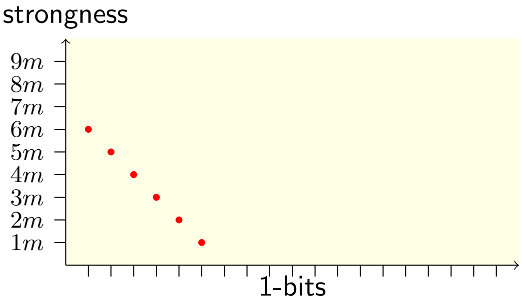
1. The best-so-far OneMax-value is at least i.
2. At least i 1-bits in x^* are well-layered:
   $1 \leq j \leq i$: bit j is jm-strong, $m := 384 \ln n$.

1-PSO on OneMax

Temporarily assume a 1-bit is never reset to 0 in x^*.

Phase i ends when

1. The best-so-far OneMax-value is at least i.
2. At least i 1-bits in x^* are well-layered:
   $1 \le j \le i$: bit j is jm-strong, $m := 384 \ln n$.



1-PSO on OneMax

Temporarily assume a 1-bit is never reset to 0 in x^*.

Phase i ends when

1. The best-so-far OneMax-value is at least i.
2. At least i 1-bits in x^* are well-layered:
   $1 \le j \le i$: bit j is jm-strong, $m := 384 \ln n$.

Well-layeredness after Phase $i - 1$
$\Rightarrow$ rediscover $i - 1$ 1-bits in expected time $O(1)$

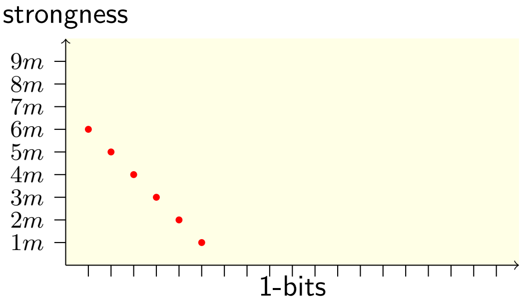
Probability to discover an i-th 1-bit is at least $\Omega((n - i)/n)$.
$\Rightarrow$ expected time $O(n/(n - i))$ for first goal of Phase i.

1-PSO on OneMax

Temporarily assume a 1-bit is never reset to 0 in x^*.

Phase i ends when

1. The best-so-far OneMax-value is at least i.
2. At least i 1-bits in x^* are well-layered:
   $1 \leq j \leq i$: bit j is jm-strong, $m := 384 \ln n$.

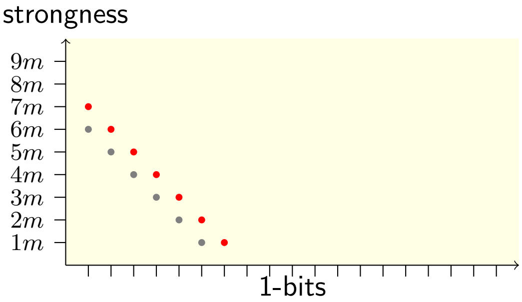


1-PSO on OneMax

Temporarily assume a 1-bit is never reset to 0 in x^*.

Phase i ends when

1. The best-so-far OneMax-value is at least i.
2. At least i 1-bits in x^* are well-layered:
   $1 \le j \le i$: bit j is jm-strong, $m := 384 \ln n$.



1-PSO on OneMax

Temporarily assume a 1-bit is never reset to 0 in x^*.

Phase i ends when

1. The best-so-far OneMax-value is at least i.
2. At least i 1-bits in x^* are well-layered:
   $1 \leq j \leq i$: bit j is jm-strong, $m := 384 \ln n$.

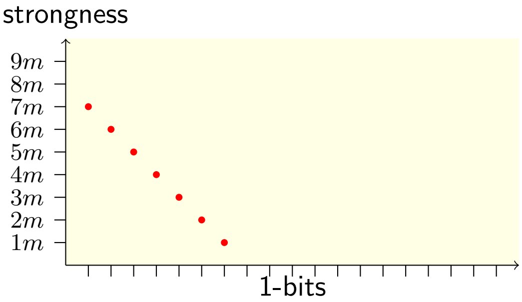
Waiting time until i 1-bits are well-layered is $O(\log n)$.

Expected runtime is bounded by

$$\sum_{i=1}^{n} O\left(\frac{n}{n-i} + \log n\right) = O(n \log n).$$

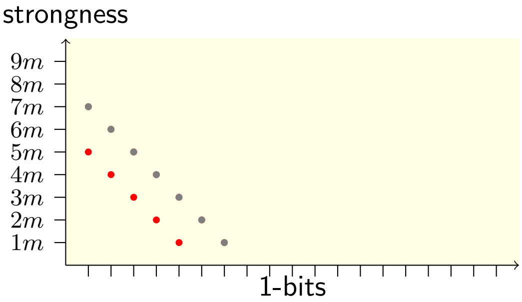
What About Resets?

Resets of 1-bits may exchange strong 1-bits for weak ones!



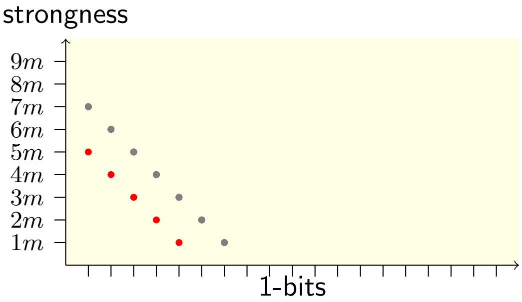
What About Resets?

Resets of 1-bits may exchange strong 1-bits for weak ones!



What About Resets?

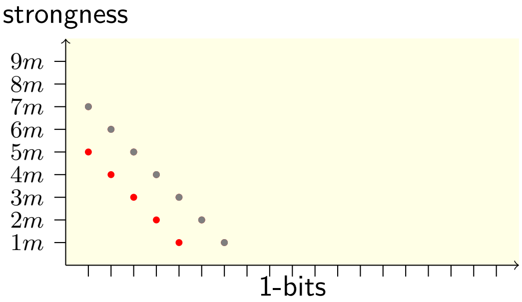
Resets of 1-bits may exchange strong 1-bits for weak ones!



- Improvement resets $O(1)$ bits of the layer in expectancy.

What About Resets?

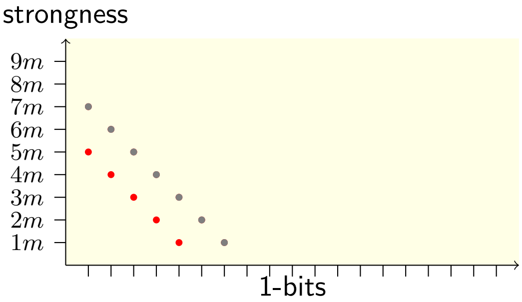
Resets of 1-bits may exchange strong 1-bits for weak ones!



- Improvement resets $O(1)$ bits of the layer in expectancy.
- Expected waiting time to recover to previous layer is $O(\ln n)$.

What About Resets?

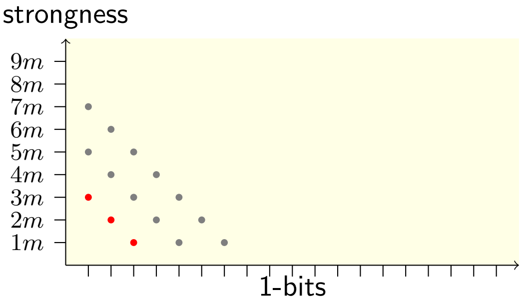
Resets of 1-bits may exchange strong 1-bits for weak ones!



- Improvement resets $O(1)$ bits of the layer in expectancy.
- Expected waiting time to recover to previous layer is $O(\ln n)$.
- Recursively repeat this argument.

What About Resets?

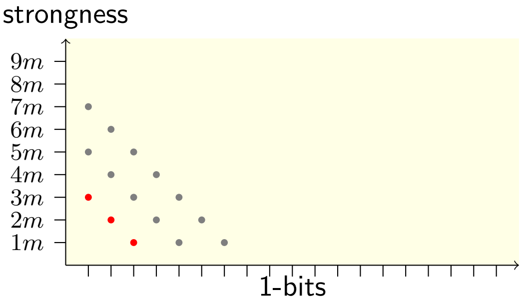
Resets of 1-bits may exchange strong 1-bits for weak ones!



- Improvement resets $O(1)$ bits of the layer in expectancy.
- Expected waiting time to recover to previous layer is $O(\ln n)$.
- Recursively repeat this argument.

What About Resets?

Resets of 1-bits may exchange strong 1-bits for weak ones!



- Improvement resets $O(1)$ bits of the layer in expectancy.
- Expected waiting time to recover to previous layer is $O(\ln n)$.
- Recursively repeat this argument.
- Expected waiting times in all improvements is $O(n \log n)$.

Conclusions

Results

- General lower bound $\Omega(n/\log n)$ for Binary PSO
- General upper bounds by fitness-level arguments if $c_1 = 0$
- Bound $O(n \log n)$ for the 1-PSO on OneMax
- Insight into the probabilistic model underlying Binary PSO.

Future Work

- Understand the interaction of particles.
- Analyses for more complicated problems.

Thanks to

Tobias Friedrich, Frank Neumann, Pietro Oliveto