Enhancing the scalability of metaheuristics by cooperative coevolution

Ciprian Crăciun   Monica Nicoară   Daniela Zaharie

West University of Timişoara
Faculty of Mathematics and Informatics

5th June 2009, Sozopol

Agenda

Context

- optimizing **high dimensional** functions
- by using **population based metaheuristics**
  - **EA** – *Evolutionary Algorithms*
  - **PSO** – *Particle Swarm Optimization*
  - **DE** – *Differential Evolution*
  - **HS** – *Harmony Search*
  - etc.
- but considering a **cooperative approach**

Problem

- for the function $f : D \mapsto \mathbb{R}$
- where
  - $D = D_1 \times D_2 \times \cdots D_n$
  - $D_i = [a_i, b_i] \subseteq \mathbb{R}$
  - n is the size of the problem
  - usually $D_1 = D_2 = \cdots = D_n$
  - x_ϕ evaluates to the minimum value $f_\phi = f(x_\phi)$
- find ...

Problem (I)

- find $x_\star$ for which $f_\star = f(x_\star)$ is the closest to f_ϕ
- but not exceeding $nfe_\star$ evaluations of function f
- in our experiments $nfe_\star = 5000 \cdot n$
- summarized: **given a fixed amount of resources find the best solution!**

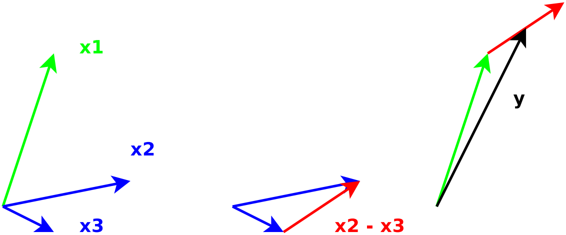
Problem (II)

- find the smallest number of evaluations nfe_ϵ
- for which $f(x_\star) - f(x_\phi) < \epsilon$
- in our experiments $\epsilon = 10^{-10}$
- summarized: **how many resources do we need to find a "good enough" solution?**

DE – *Differential Evolution [Storn-1995]*

- let $X = \{x^1, x^2, \ldots, x^m\} \subset D$ be a population of possible solutions
- we construct a new population $Y = \{y^1, y^2, \ldots, y^m\} \subset D$
- by combining elements from X
  - $y^i_j = \begin{cases} x^{r_1}_j + F^i \cdot (x^{r_2}_j - x^{r_3}_j) & \text{if } U < CR^i \\ x^i_j & \text{otherwise} \end{cases}$
- where
  - r_1, r_2 and r_3 are distinct random values in $\{1, 2, \ldots, m\}$
  - U is a random value in the range $[0, 1]$
  - CR is a control parameter: crossover rate
  - F is another control parameter: scale factor
  - m is the size of the population
- we construct the next X' by choosing $x'^i = best(x^i, y^i)$
- we use the **self-adaptive** **jDE** *[Brest-2007]*
- there are both synchronous / asynchronous variants

DE example



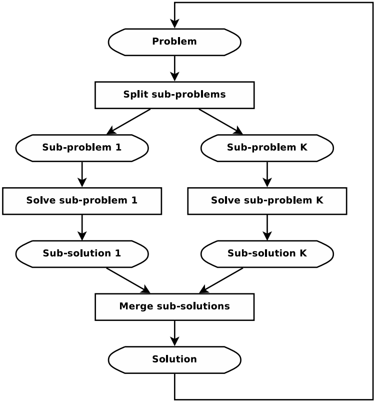
HS – *Harmony Search [Geem-2001]*

- **HS** resembles **DE**
- except the generation function
  - $y_j^i = \begin{cases} x_r^i + bw \cdot U_3 & \text{if } U_1 < HMCR \text{ and } U_2 < PAR \\ x_r^j & \text{if } U_1 < HMCR \text{ and } U_2 \geq PAR \\ random(D_j) & \text{otherwise} \end{cases}$
- where
  - r is a random value in $\{1, 2, \ldots, m\}$
  - U_1 and U_2 are random vales in the range $[0, 1]$
  - U_3 is a random value in the range $[-1, 1]$
  - $\mathbf{bw} = \sqrt{\mathbf{var}(\mathbf{X_j})}$: perturbation magnitude
    - described in *[Mukhopadhyay-2008]*
  - $HMCR = 0.99$ is a control parameter: crossover rate
  - $PAR = 0.75$ is another control parameter: perturbation rate

Method

- **splitting the problem** into smaller sub-problems
- evolving each sub-problem in isolation
- at certain points **merging the sub-solutions** into a solution
- repeating the process until an acceptable solution is found
- described in *[Potter-1994]*
- recent work in *[Yang-2008]*, *[Shi-2005]*, *[Berg-2004]*

Method diagram



Main issues

- how to split the problem into sub-problems?
- how to evolve the sub-problems?
  - how to evaluate?
  - how to merge solutions?
- what are the **best practices** for setting control parameters?

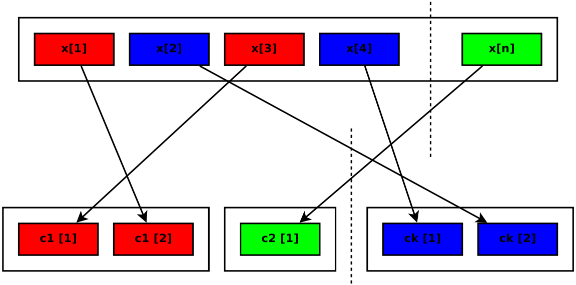
Components

- each solution $x = (x_1, x_2, \ldots, x_n)$
- should be decomposed into components $x = \langle C_1(x), C_2(x), \ldots, C_K(x) \rangle$
- where
  - K is the number of components
  - $C(x)$ is a component **of non consecutive variables**
  - $\langle \cdot \rangle$ is the merging function
- by notation $C_k(X)$ is a population composed from the k-th component of all elements in X

Choosing the components

- (ideally) based on the corellation between the variables
- choosing the number of components
  - $K = n$
  - $K \in \{K_{min}, \ldots, K_{max}\}$, K_{min} and K_{max} are control parameters
  - K is **fixed** and is a control parameter
- choosing the number of variables in each component n_C
  - $n_C = n/K$ – all components are **equal in size**
  - $avg(n_C) = n/K$
- assigning variables to each component
  - randomly distributed
  - only at the beginning of the evolution
  - at the beginning of **each generation**

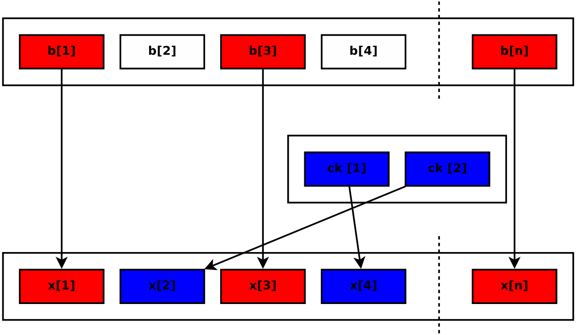
Choosing the components (diagram)



Evaluating the components

- after creating $C_k(y) = generate(C_k(X))$ for the solution x
- how to evaluate it $f(C_k(y)) = ?$
- by creating a virtual solution
  - $y = \langle C_1(w), \ldots, C_{k-1}(w), C_(y), C_{k+1}(w), \ldots, C_K(w) \rangle$
- where w is either
  - the **current solution** x
  - the best solution $x_\star$
  - the worst solution
  - a random solution
  - or evaluate some of the above and take the maximum

Evaluating the components (diagram)



Combinations

- most algorithms are both synchronous and asynchronous
- the same can be said about the **CC** framework
- the overall algorithm could be
  - **fully-synchronous**
  - **fully-asynchronous**
  - synchronous for components / asynchronous for a component
  - (there is no sense in the reverse of the above)

sCC – *Synchronous CC*

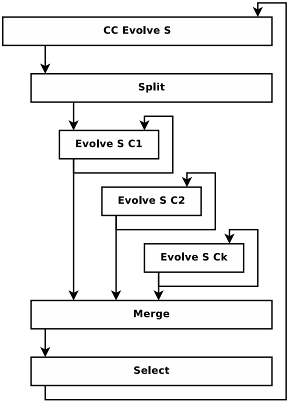
CCEvolveS(X)

```
initialize and evaluate X
while goal for X not reached do
    choose K
    construct C_1(X) ... C_K(X)
    for k ∈ overline{1,K} do
        C_k(V) ← evolveS(C_k(X), X)
    end for
    evaluate V
    X ← select(X, V)
end while
return X
```

evolveS$(C_k(X), X)$

```
C_k(W) ← C_k(X)
while goal for C_k(X) not reached do
    C_k(Y) ← generate(C_k(W))
    Y ← merge(C_k(Y), X)
    evaluate Y
    C_k(Y) ← select(C_k(Y), C_k(W))
    C_k(W) ← C_k(Y)
end while
return C_k(Y)
```

sCC diagram


CC Evolve S
Split
Evolve S C1
Evolve S C2
Evolve S Ck
Merge
Select

aCC – *Asynchronous CC*

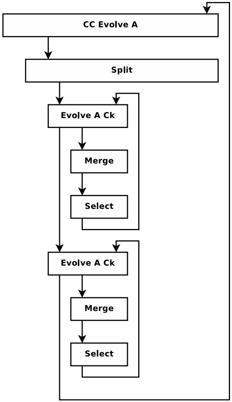
CCEvolveA(X)

```
initialize and evaluate X
while goal for X not reached do
    choose K
    construct C_1(X), ..., C_K(X)
    for k ∈ 1,K (overline) do
        while goal for C_k(X) not reached do
            for C_k(x) ∈ C_k(X) do
                C_k(x) ← evolveA(C_k(x), C_k(X), X)
            end for
        end while
    end for
end while
return X
```

evolveA$(C_k(x), C_k(X), X)$

```
C_k(y) ← generate(C_k(X))
y ← merge(C_k(y), X)
evaluate y
C_k(y) ← select(C_k(x), C_k(y))
return C_k(y)
```

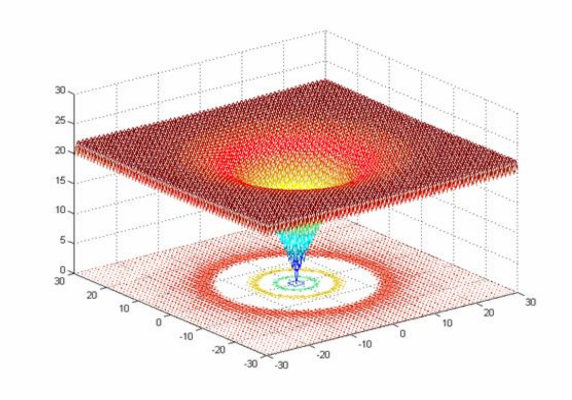
aCC diagram


CC Evolve A
Split
Evolve A Ck
Merge
Select
Evolve A Ck
Merge
Select

Overview

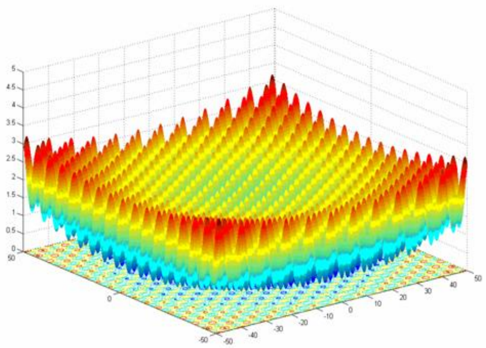
- we have conducted 992 preliminary tests
- the presented results are gathered from **384 tests**
- each test has **30 independent runs**
- benchmark functions
  - from **CEC2008** benchmark for **large scale optimization**
    - Ackley – non-separable (medium difficulty)
    - Griewank – non-separable (medium/high difficulty)
    - Rastrigin – separable (medium/high difficulty)
    - Sphere – separable (easy function)
  - all test functions are **shifted**
  - described in *[Tang-2007]*

Ackley function



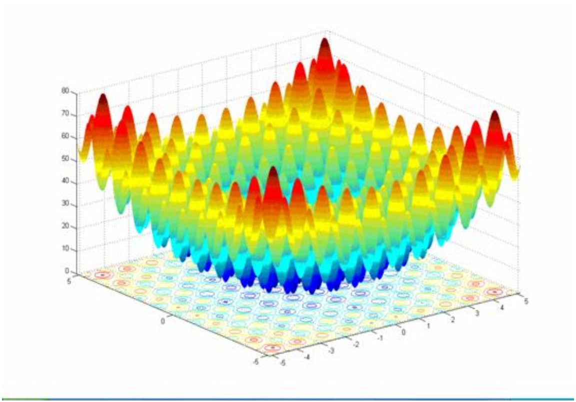
http://bit.ly/14Is0c

Griewank function



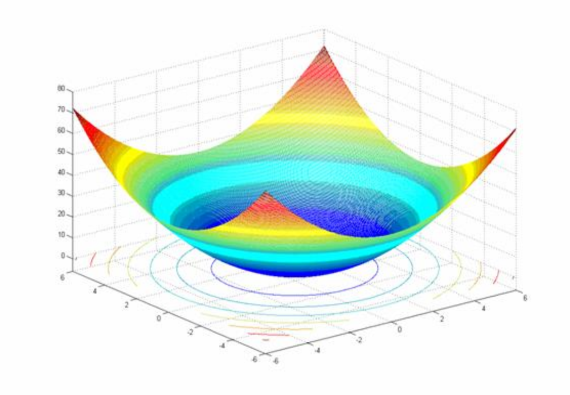
http://bit.ly/ujipZ

Rastrigin function



http://bit.ly/13IGyy

Sphere function



http://bit.ly/1frmo

Comparison between **CCDE** and **DE**, $f_\star$

$n = 200$	Ackley	Griewank	Rastrigin
sDE	$9.1e-13$	$3.4e-03$	$1.1e+00$
aDE	$1.3e-12$	$1.1e-02$	$1.4e+00$
sCCDE	**$1.6e-13$**	**$5.7e-14$**	**$0.0e+00$**
aCCDE	$1.8e-13$	**$5.7e-14$**	**$0.0e+00$**
$n = 500$	Ackley	Griewank	Rastrigin
sDE	$1.0e-11$	$2.4e-03$	$1.5e+00$
aDE	$4.3e-12$	$2.3e-02$	$1.8e+00$
sCCDE	**$4.6e-13$**	**$1.6e-13$**	**$1.3e-14$**
aCCDE	$4.8e-13$	$1.7e-13$	**$9.4e-15$**
$n = 1000$	Ackley	Griewank	Rastrigin
sDE	$4.0e-01$	$4.1e-01$	$3.4e+01$
aDE	$9.3e-01$	$3.8e-01$	$4.7e+01$
sCCDE	**$1.0e-12$**	**$3.6e-13$**	**$4.5e-14$**
aCCDE	**$1.0e-12$**	**$3.6e-13$**	**$4.5e-14$**

statistical analysis based on Student's t test for $\alpha = 0.05$

Comparison between **CCDE** and **DE**, nfe_ϵ

$n = 200$	Ackley	Griewank	Rastrigin
sDE	**100%/4.0e + 05**	93%/2.4e + 05	100%/2.6e + 05
aDE	100%/3.7e + 05	80%/2.3e + 05	**100%/2.5e + 05**
sCCDE	100%/5.0e + 05	100%/3.0e + 05	100%/4.0e + 05
aCCDE	**100%/4.7e + 05**	**100%/2.9e + 05**	**100%/3.7e + 05**
$n = 500$	Ackley	Griewank	Rastrigin
sDE	96%/1.4e + 06	96%/9.1e + 05	36%/9.4e + 05
aDE	100%/1.3e + 06	90%/8.9e + 05	20%/9.5e + 05
sCCDE	**100%/1.2e + 06**	100%/7.9e + 05	100%/1.0e + 06
aCCDE	**100%/1.2e + 06**	**100%/7.5e + 05**	**100%/9.8e + 05**
$n = 1000$	Ackley	Griewank	Rastrigin
sDE	10%/4.4e + 06	16%/2.5e + 06	0%/ − −
aDE	6%/3.9e + 06	46%/2.4e + 06	0%/ − −
sCCDE	100%/2.5e + 06	**100%/1.6e + 06**	**100%/1.9e + 06**
aCCDE	**100%/2.4e + 06**	**100%/1.5e + 06**	**100%/1.9e + 06**

statistical analysis based on Student's t test for $\alpha = 0.05$

Comparison between **CCHS** and **HS**, $f_\star$

$n = 200$	Ackley	Griewank	Rastrigin
HS	$4.2e+00$	$4.0e+00$	$4.2e+02$
CCHS	**3.6e − 13**	**1.3e − 13**	**7.5e − 15**
$n = 500$	Ackley	Griewank	Rastrigin
HS	$1.6e+01$	$1.9e+03$	$1.6e+03$
CCHS	**9.1e − 13**	**3.8e − 13**	**5.1e − 14**
$n = 1000$	Ackley	Griewank	Rastrigin
HS	$1.9e+01$	$8.6e+03$	$4.6e+03$
CCHS	**2.5e − 12**	**5.8e − 13**	**2.0e − 13**

statistical analysis based on Student's t test for $\alpha = 0.05$

Comparison between **CCHS** and **HS**, nfe_ϵ

$n = 200$	Ackley	Griewank	Rastrigin
HS	$0\%/--$	$0\%/--$	$0\%/--$
CCHS	**100%/6.2e + 05**	**100%/3.7e + 05**	**100%/4.9e + 05**
$n = 500$	Ackley	Griewank	Rastrigin
HS	$0\%/--$	$0\%/--$	$0\%/--$
CCHS	**100%/1.6e + 06**	**100%/1.0e + 06**	**100%/1.3e + 06**
$n = 1000$	Ackley	Griewank	Rastrigin
HS	$0\%/--$	$0\%/--$	$0\%/--$
CCHS	**100%/4.3e + 06**	**100%/2.8e + 06**	**100%/4.3e + 06**

statistical analysis based on Student's t test for $\alpha = 0.05$

Study for **CCDE** on Griewank 1000, $f_\star$

	K	G_K	$f_\star$
sCCDE	10	100	$3.6e-13$
aCCDE	10	100	$3.6e-13$
aCCDE	20	1	$4.9e-13$
sCCDE	20	5	$4.9e-13$
aCCDE	20	5	$4.9e-13$
aCCDE	20	10	$4.9e-13$
sCCDE	20	1	$4.9e-13$
aCCDE	20	50	$5.0e-13$
sCCDE	20	10	$5.0e-13$
sCCDE	20	50	$5.0e-13$

K – the number of components
G_K – the iterations per component

Study for **CCDE** on Griewank 1000, nfe_ϵ

	K	G_K	SR	nfe_ϵ
aCCDE	10	100	100%	$1.5e+06$
sCCDE	10	100	100%	$1.6e+06$
aCCDE	20	1	100%	$1.7e+06$
aCCDE	20	5	100%	$1.8e+06$
sCCDE	20	1	100%	$1.8e+06$
aCCDE	20	10	100%	$1.8e+06$
aCCDE	20	50	100%	$1.9e+06$
sCCDE	20	5	100%	$1.9e+06$
sCCDE	20	10	100%	$2.0e+06$
sCCDE	20	50	100%	$2.0e+06$

K – the number of components
G_K – the iterations per component

Study for **CCHS** on Griewank 1000, $f_\star$

	K	G_K	$f_\star$
CCHS	20	1	$5.8e-13$
CCHS	20	5	$6.0e-13$
CCHS	20	10	$6.4e-13$
CCHS	20	50	$1.6e-12$
CCHS	10	100	$9.6e-04$

K – the number of components
G_K – the iterations per component

Study for **CCHS** on Griewank 1000, nfe_ϵ

	K	G_K	SR	nfe_ϵ
CCHS	20	1	100%	$2.8e+06$
CCHS	20	5	100%	$3.1e+06$
CCHS	20	10	100%	$3.3e+06$
CCHS	20	50	100%	$4.0e+06$
CCHS	20	100	0%	$--$

K – the number of components
G_K – the iterations per component

Achievements

- generalized the **CC** framework
- applied successfully the methodology to **HS**
- studied the impact of the **CC** control parameters

Conclusions

- the **CC** method (with random components) **scales** quite nice
- and also **adapts easily** for new metaheuristics
- concerning the **control parameters**
  - for some, like **DE**, they don't have much impact
  - for others, like **HS**, they make a difference

Questions

- questions?
- remarks?
- comments?
- http://web.info.uvt.ro/natcomp/lssc09