Dynamic Optimization using Self-Adaptive Differential Evolution

IEEE Congress on Evolutionary Computation (IEEE CEC 2009), Trondheim, Norway, May 18-21, 2009

J. Brest, A. Zamuda, B. Bošković, M. S. Maučec, V. Žumer

Faculty of Electrical Engineering and Computer Science
University of Maribor

May 19, 2009

Introduction

- Differential Evolution (DE) is simple yet powerful EA algorithm for optimizing continuous functions, e.g. *static optimization environment*.
- CEC 2009 special session on evolutionary computation in *dynamic* and *uncertain environments*.
- Main goal: self-adaptive DE algorithm + multi-populations

Introduction

In this presentation:

- hybridization of our self-adaptive differential evolution algorithm *jDE* with multi-populations, aging, overlapping search
- performance comparison on the set of benchmark problems

The Differential Evolution Algorithm

- Price & Storn, 1995 (JGO:1997)
  - NP .. population size (D-dimensional vectors)
  - F .. mutation scale factor
  - CR .. crossover parameter

The Differential Evolution Algorithm

- Price & Storn, 1995 (JGO:1997)
  - NP .. population size (D-dimensional vectors)
  - F .. mutation scale factor
  - CR .. crossover parameter

- "rand/1" strategy:
  $\vec{v}_i^{(G)} = \vec{x}_{r_1}^{(G)} + F \cdot (\vec{x}_{r_2}^{(G)} - \vec{x}_{r_3}^{(G)}), \quad r_1 \neq r_2 \neq r_3 \neq i$

The Differential Evolution Algorithm

- Price & Storn, 1995 (JGO:1997)
  - NP .. population size (D-dimensional vectors)
  - F .. mutation scale factor
  - CR .. crossover parameter

- "rand/1" strategy:
  $\vec{v}_i^{(G)} = \vec{x}_{r_1}^{(G)} + F \cdot (\vec{x}_{r_2}^{(G)} - \vec{x}_{r_3}^{(G)}), \quad r_1 \neq r_2 \neq r_3 \neq i$

- $u_{i,j}^{(G)} = \begin{cases} v_{i,j}^{(G)} & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand}, \\ x_{i,j}^{(G)} & \text{otherwise,} \end{cases}$
  where $i = 1, 2, ..., NP$ and $j = 1, 2, ..., D$.

The Differential Evolution Algorithm

- Price & Storn, 1995 (JGO:1997)
  - NP .. population size (D-dimensional vectors)
  - F .. mutation scale factor
  - CR .. crossover parameter

- "rand/1" strategy:
  $$\vec{v}_i^{(G)} = \vec{x}_{r_1}^{(G)} + F \cdot (\vec{x}_{r_2}^{(G)} - \vec{x}_{r_3}^{(G)}), \quad r_1 \neq r_2 \neq r_3 \neq i$$

- $$u_{i,j}^{(G)} = \begin{cases} v_{i,j}^{(G)} & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand}, \\ x_{i,j}^{(G)} & \text{otherwise,} \end{cases}$$
  where $i = 1, 2, ..., NP$ and $j = 1, 2, ..., D$.

- $\vec{x}$, $\vec{u}$ better survives

The Self-adaptive DE Algorithm: *jDE* algorithm

$$F_i^{(G+1)} = \begin{cases} F_l + rand_1 \cdot F_u & \text{if } rand_2 < \tau_1, \\ F_i^{(G)} & \text{otherwise,} \end{cases}$$

$$CR_i^{(G+1)} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2, \\ CR_i^{(G)} & \text{otherwise.} \end{cases}$$

$\tau_1 = 0.1, \tau_2 = 0.1, F_l = 0.1, F_u = 0.9$ (fixed values)
$F \in [0.1, 1.0], CR \in [0, 1]$

[4] J. Brest et al. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE TEVC*, 10(6):646–657, 2006.

Hybridized algorithm for solving Dynamic Optimization Problems (DOPs)

- **multi-populations** (random indexes r_1, r_2, and r_3 indicate vectors (individuals) that belong to same subpopulation as the trial vector $\vec{x}_i$)
- **self-adaptive control mechanism, F belongs to interval** $[0.36, 1]$ ($F_l = 0.36$ – suggested by D. Zaharie [22])
- **aging at individual level** (an individual that stagnates in local optimum should be reinitialized)
- **overlapping search between two subpopulations** (distance of the best individuals of the subpopulations)
- **reinitialization** (when individual is close to local best)
- **archive** (currently best individual is added to archive after each change is detected)

Our algorithm – Multi-populations

- random indexes r_1, r_2, and r_3 indicate vectors (individuals) that belong to same subpopulation as the trial vector $\vec{x}_i$)
- more populations without any information sharing (except overlapping search between two best individuals of two subpopulations)
- subpopulations search different regions – diversity is important feature in DOPs

F belongs to interval $[0.36, 1]$

- D. Zaharie [22] "critical values for the control parameters of DE"
- $2F^2 - 2/NP + CR/NP = 0$ ... "can be considers to be critical" (this formula has an error in the paper)
- assume $CR = 0$ and $NP = 10$ then critical value for F is 0.308 (0.424 when $NP = 5$)
- we set $F_l = 0.36$ in all experiments

Our algorithm – aging at individual level

- an individual that stagnates in local optimum should be reinitialized
- each individual has its own *age*-variable (incremented once per generation)
- three rules for aging (see Alg. 1):
  - *global best* is not reinitialized
  - when *local best* needs to be reinitialized, the whole subpopulation with some probability is reinitialized
  - another individual is reinitialized with some probability

Our algorithm – Individual's improvement and aging

- when an improvement of individual occurs the *age* is set to some small value – the new promising individual should stay in population for more generations
- the distance measure and fitness are used to make decision when individual's improvement is *small* or *big* (see Alg. 4)

Our algorithm – Archive

- algorithm starts with empty archive
- currently best individual is added to the archive, after each change is detected
- an individual is selected from archive only for the first subpopulation

Parameter Settings

- F self-adaptive,
- CR self-adaptive,
- $NP = 50$,
- number of sub-populations: 5 (the size of each sub-populations was 10).

Results

Table: CEC'09 Dynamic Optimization benchmark functions

F_1	Rotation peak function
F_2	Composition of Sphere's function
F_3	Composition of Rastrigin's function
F_4	Composition of Griewank's function
F_5	Composition of Ackley's function
F_6	Hybrid Composition function

Results

Table: Error Values Achieved for Problems F_1

Dimension(n)	Peaks(m)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	10	Avg_best	0	0	0	0	0	0
		Avg_worst	0.910466	32.1705	31.7827	0.919964	18.392	32.7662
		Avg_mean	0.028813	3.5874	2.99962	0.015333	2.17757	1.1457
		STD	0.442537	7.83849	7.12954	0.288388	4.38812	5.72962
	50	Avg_best	0	0	0	0	0	0
		Avg_worst	3.92056	30.1958	27.6823	1.21212	9.08941	33.1204
		Avg_mean	0.172355	4.08618	4.29209	0.0877388	0.948359	1.76542
		STD	0.763932	6.4546	6.74538	0.24613	1.76552	5.82652
T_7(5-15)	10	Avg_best	—	—	0	—	—	—
		Avg_worst	—	—	34.8377	—	—	—
		Avg_mean	—	—	3.5017	—	—	—
		STD	—	—	7.89858	—	—	—
	50	Avg_best	—	—	0	—	—	—
		Avg_worst	—	—	29.768	—	—	—
		Avg_mean	—	—	4.36913	—	—	—
		STD	—	—	6.9321	—	—	—

Results

Table: Error Values Achieved for Problems F_2

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best	0	0	0	0	0	0
	Avg_worst	15.4426	435.019	468.43	10.6608	459.147	49.5327
	Avg_mean	0.963039	43.0004	50.1906	0.793141	67.0523	3.36653
	STD	3.08329	114.944	124.015	2.53425	130.146	12.9738
T_7(5-15)	Avg_best	—	—	0	—	—	—
	Avg_worst	—	—	226.332	—	—	—
	Avg_mean	—	—	13.2524	—	—	—
	STD	—	—	45.7797	—	—	—

Results

Table: Error Values Achieved for Problems F_3

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best	0	9.70434e-08	3.13019e-10	0	5.35102e-10	8.17124e-14
	Avg_worst	238.417	938.858	944.695	922.236	874.852	1226.38
	Avg_mean	11.3927	558.497	572.105	65.7409	475.768	243.27
	STD	58.1106	384.621	386.09	208.925	379.89	384.98
T_7(5-15)	Avg_best	—	—	0	—	—	—
	Avg_worst	—	—	853.061	—	—	—
	Avg_mean	—	—	153.673	—	—	—
	STD	—	—	286.379	—	—	—

Results

Table: Error Values Achieved for Problems F_4

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	**Avg_best**	**0**	**0**	**0**	**0**	**0**	**0**
	Avg_worst	**19.623**	**475.7**	**544.92**	**16.6057**	**510.193**	**28.4483**
	Avg_mean	**1.48568**	**49.5044**	**51.9448**	**1.50584**	**69.4395**	**2.35478**
	STD	**4.47652**	**135.248**	**141.78**	**4.10062**	**144.041**	**5.78252**
T_7**(5-15)**	**Avg_best**	—	—	**0**	—	—	—
	Avg_worst	—	—	**163.727**	—	—	—
	Avg_mean	—	—	**11.7425**	—	—	—
	STD	—	—	**39.4469**	—	—	—

Results

Table: Error Values Achieved for Problems F_5

Dim.(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best	4.10338e-14	4.16556e-14	4.15668e-14	4.08562e-14	4.24549e-14	4.08562e-14
	Avg_worst	4.89413	9.6899	10.1371	4.75098	9.28981	4.78684
	Avg_mean	0.159877	0.333918	0.357925	0.108105	0.409275	0.229676
	STD	1.02554	1.64364	1.83299	0.826746	1.90991	0.935494
T_7(5-15)	Avg_best	—	—	4.12115e-14	—	—	—
	Avg_worst	—	—	11.8188	—	—	—
	Avg_mean	—	—	0.434294	—	—	—
	STD	—	—	2.22792	—	—	—

Results

Table: Error Values Achieved for Problems F_6

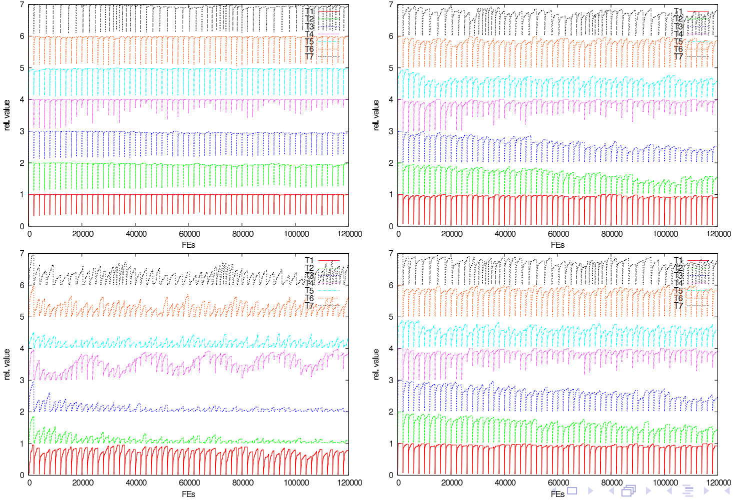
Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	**Avg_best**	**0**	**0**	**0**	**0**	**0**	**0**
	Avg_worst	**32.7204**	**51.8665**	**84.519**	**38.7914**	**191.895**	**45.0354**
	Avg_mean	**6.22948**	**10.3083**	**10.954**	**6.78734**	**14.9455**	**7.8028**
	STD	**10.4373**	**13.2307**	**23.2974**	**10.1702**	**45.208**	**10.9555**
T_7**(5-15)**	**Avg_best**	—	—	**0**	—	—	—
	Avg_worst	—	—	**58.9448**	—	—	—
	Avg_mean	—	—	**10.736**	—	—	—
	STD	—	—	**14.7267**	—	—	—

Results

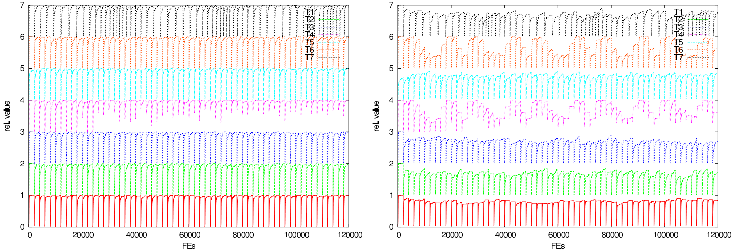
Table: Algorithm Overall Performance

	F_1(10)	F_1(50)	F_2	F_3	F_4	F_5	F_6
T_1	**0.014768**	**0.0146876**	**0.0211049**	**0.0157107**	**0.0206615**	**0.021766**	**0.0170472**
T_2	**0.0136901**	**0.0135926**	**0.0135271**	**0.00298238**	**0.013148**	**0.0208661**	**0.0139488**
T_3	**0.0138256**	**0.0135304**	**0.0130808**	**0.00281439**	**0.013545**	**0.0209286**	**0.0141912**
T_4	**0.0147164**	**0.0146941**	**0.0210035**	**0.0127621**	**0.0199268**	**0.0221962**	**0.0153046**
T_5	**0.0139415**	**0.0143644**	**0.0123976**	**0.0044056**	**0.012376**	**0.0213094**	**0.0155184**
T_6	**0.0141265**	**0.013874**	**0.017776**	**0.00734523**	**0.0179501**	**0.0207361**	**0.0139512**
T_7	**0.00911221**	**0.00898569**	**0.0101876**	**0.00549392**	**0.0101813**	**0.0137894**	**0.00942562**
Mark	**0.0941803**	**0.0937288**	**0.109078**	**0.0515143**	**0.107789**	**0.141592**	**0.099387**
Performance (sumed the mark obtained for each case and multiplied by 100): 69.7269							

Convergence graphs F_1–F_4



Convergence graphs F_5, and F_6



Discussion

- our algorithm performs very well on small step (T_1) and chaotic (T_4) change types for $F_1 - F_4$
- F_5: it obtained good results over all changed types
- F_6: it obtained very well results over all changed types
- F_3 is the most difficult one among all test problems

Conclusions

jDE algorithm with multi-populations and aging mechanism was evaluated on CEC'09 test problems – special session on dynamic optimization problems.

Overall performance: 69.7

Future plans:

- to apply additional co-operation among sub-populations
- to use sub-populations of different sizes
- to improve the usage of the archive (here, a simple variant of the archive is used)

Thank You

Questions?