Learning the Best Combination of Solvers in a Distributed Global Optimisation Environment

Tamás Vinkó, Dario Izzo

Advanced Concepts Team
DG-PI, ESTEC
European Space Agency

AGO2007, June 16, 2007

Motivation

- There are several software platforms and physical infrastructures to establish distributed computing environment.
- Among of them, the most famous platforms are the volunteer based ones, especially the BOINC.
- The SETI@Home is one of the best known project using the BOINC framework.
- Typically in these projects the distributed computation is such that each client computation is quite independent from the others (data processing).

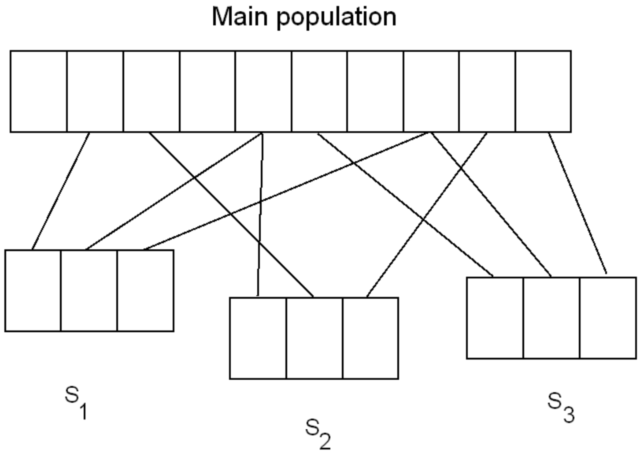
ACT–DC

Advanced Concepts Team's Distributed Computing Environment:

- highly modular development
- readily available data storage classes
- can deal with a wide range of computationally hard problems, including single-, and multi-objective global optimization, constraint satisfaction, numerical integration, satellite data processing, computational fluid dynamics, etc.

The development was done by Mihály Csaba Markót (was RF at ESA/ACT in 2004–2006).

Distribution strategy



This implementation makes efficient our solvers due to the diversity of the population elements in terms of their location.

Implemented solvers

- Differential Evolution
- Particle swarm optimization
- Genetic algorithm
- Simulated annealing

Common feature: stochastic solvers, all population based but SA.

DiGMO

No free lunch theorem: if an algorithm performs well on one class of problems then it must do poorly on other problems. Moreover, an algorithm that performs well on one class of problems must perform worse than random search on all remaining problems.

What about applying all of them?

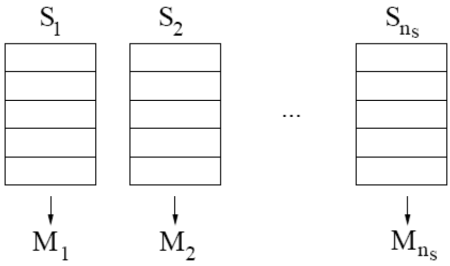
Based on some selection rule the server can choose between the solvers to be applied in the next round.

We call this implementation DiGMO (Distributed Global Multiobjective Optimizer).

Server algorithm

```
 1: Initialize;
 2: while (1)
 3:     Listening;
 4:     Receive data and extract the message;
 5:     Update server internal data;
 6:     Set response to be sent;
 7:     Send message to client and update server data;
 8:     Close client socket;
 9:     if Finish() then break;
10: end while
11: Close;
```

Update server internal data



In case of minimization, let

$$s := \sum_{i=1}^{n_s} \frac{1}{M_i}, \quad P_i := \alpha \frac{1}{sM_i} + (1-\alpha)\frac{1}{n_s},$$

where $0 < \alpha < 1$.

The α value gives some chances to the solvers with low probability to be selected.

Selection rules I.

Rule A. Put the obtained objective function value by the solver to the stack.
We refer to this rule as CO-f-valueofα (e.g. CO-f-0.1).

Rule B. The difference between the reached objective value of the current run and the best known function value is put to the history stack.
We refer to this rule as CO-d-valueofα (e.g. CO-d-0.1).

Selection rules II.

Rule C. Let μ_I the mean fitness value of the selected subpopulation before the run and μ_O after the run of the selected solver. Put the difference $\mu_O - \mu_I$ to the stack.
We refer to this rule as CO-μ-valueofα (e.g. CO-μ-0.1).

Rule D. In this rule there is no learning and all the probabilities are equal. This can be obtained by letting $\alpha = 0$ in the previous rules.
We refer to this rule as CO-r.

Server algorithm

```
1:  Initialize;
2:  while (1)
3:      Listening;
4:      Receive data and extract the message;
5:      Update server internal data;
6:      Set response to be sent;
7:      Send message to client and update server data;
8:      Close client socket;
9:      if Finish() then break;
10: end while
11: Close;
```

Set response to be sent

The algorithm selects a solver to be used by the client.

At this step we have the probability level R_j of the solver j. Here $R_1 := P_1$ and $R_j := R_{j-1} - P_j$ $(j = 2, \ldots, n_s)$.

In order to make this selection, a random number, $0 < \rho < 1$ is generated. The solver j is selected for which $\rho < R_j$ holds and j is the smallest index.

Experimental results

Results obtained distributing in the ACT-DC single stand-alone solvers as well as the cooperative solver are reported.

Our aim is not to find the best solver or to find better solution to the problems taken into account, but to use a fixed setup of the solvers, to allow a fixed number of objective function evaluation and to see if the proposed cooperative strategies are able to improve the performances.

Experimental results

The total number of function evaluation was set up to 800,000.

Each solver was allowed to take 20,000 function evaluations.

The size of the main population was 60, while the subpopulation size was fixed to 10.

We tested all the stand alone solvers and the cooperative version with the four different rules.

Each experiment was repeated 20 times and average and standard deviations of results were recorded.

Cassini-like mission (EVVEJS)

Table: Performance of the different stand-alone solvers and the cooperative rules on EVVEJS. The best function value, the average function value, standard deviation of the reached function values are indicated.

solver	best	aver.	std.dev.
DE	4.93071	6.26080	2.11509
PSO	4.94546	9.70161	3.16255
SA	6.43017	9.31858	2.45137
GA	5.63979	6.63026	2.42053
CO-f-0.1	4.93071	5.95117	1.84035
CO-d-0.1	5.30320	5.82047	1.56067
CO-μ-0.1	4.93071	5.55113	1.25423
CO-r	4.93071	6.2118	1.92312

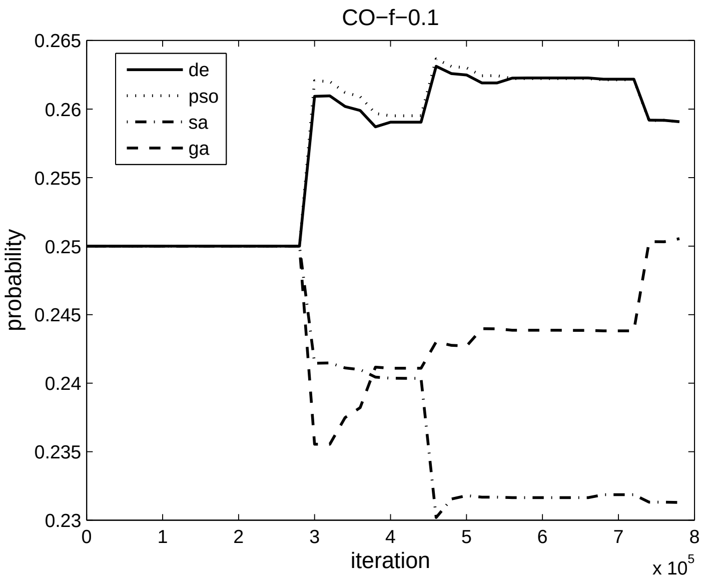


Figure: The probabilities of the best run using Rule A on EVVEJS.

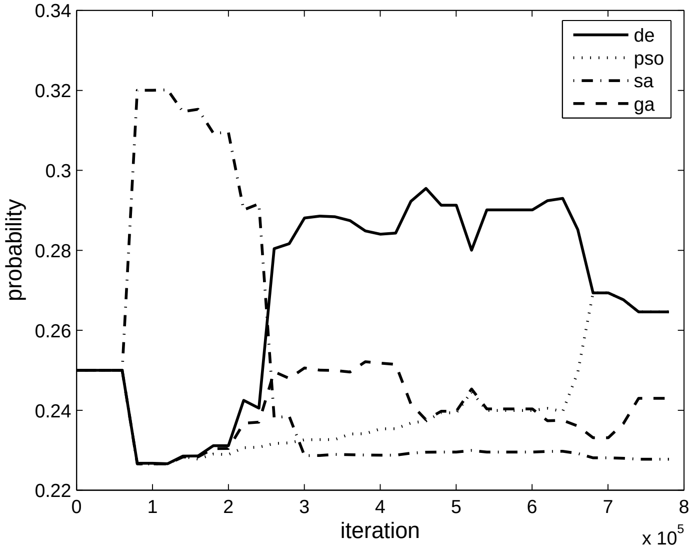


Figure: The probabilities of the best run using Rule B on EVVEJS.

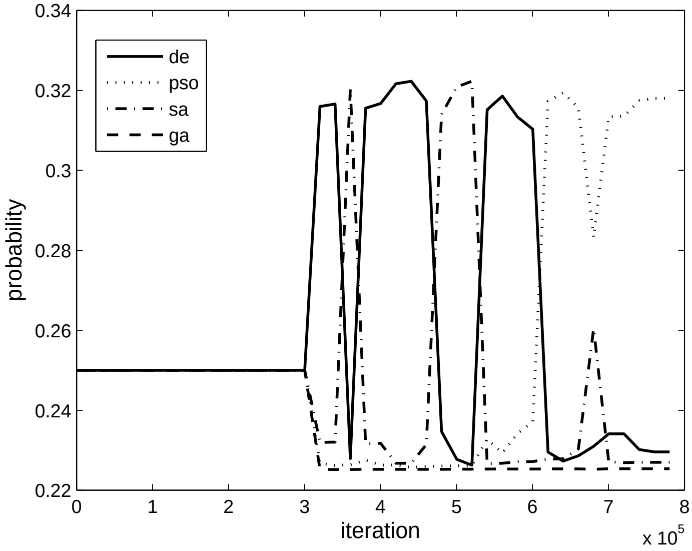


Figure: The probabilities of the best run using Rule C on EVVEJS.

Mission to asteroid TW229 (EVEVEJSA)

Table: Performance of the different stand-alone solvers and the cooperative rules on EVEVEJSA. The best function value, the average function value, standard deviation of the reached function values are indicated. Units are 1e+6.

solver	best	aver.	std.dev.
DE	1.33001	1.11737	0.098539
PSO	1.21496	0.77964	0.379116
SA	1.13438	0.66648	0.296399
GA	1.59310	1.12833	0.212759
CO-f-0.1	1.62913	1.225158	0.189813
CO-d-0.1	1.74324	1.34113	0.180753
CO-μ-0.1	1.76075	1.29224	0.230117
CO-r	1.69048	1.30178	0.204804

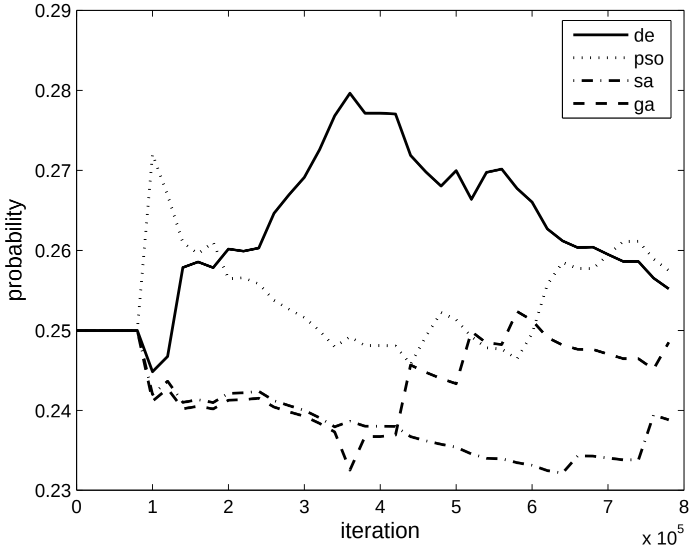


Figure: The probabilities using Rule A on EVEVEJSA.

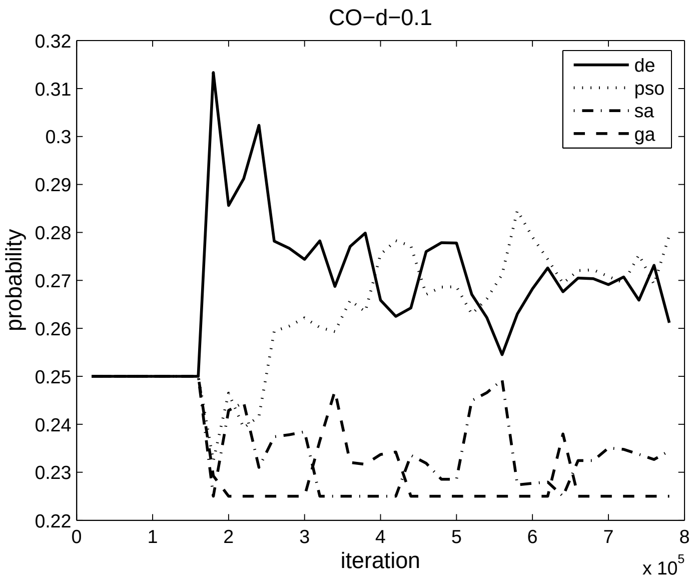


Figure: The probabilities using Rule B on EVEVEJSA.

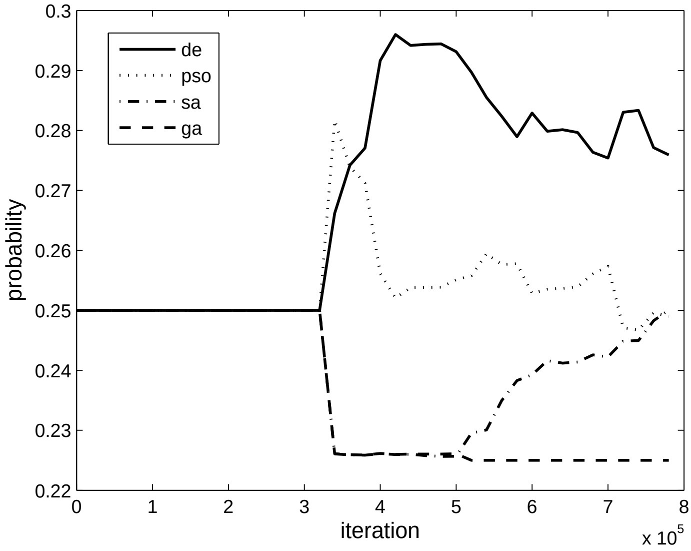


Figure: The probabilities using Rule C on EVEVEJSA.

Lennard-Jones cluster with 17 atoms ($n = 45$)

Table: Performance of the different stand-alone solvers and the cooperative rules on Lennard-Jones. The best function value, the average function value, standard deviation of the reached function values are indicated. The best known value is -61.317995.

solver	best	aver.	std.dev.
DE	-54.1713	-52.0979	1.18448
PSO	-48.9036	-44.7066	2.91419
SA	-21.5146	-17.6238	1.75364
GA	-18.3787	-15.2963	1.26401
CO-f-0.1	-56.0306	-52.4345	2.00370
CO-d-0.1	-55.4813	-52.6745	1.53009
CO-r	-56.4304	-52.5847	2.34895

Conclusions

- We introduced a new distributed solver that uses cooperatively standard versions of some stochastic solvers.
- We find that this cooperative solver outperforms the stand alone ones already if a random probability is given to assign to a client a particular solver.
- We also find that a further improvement is possible by letting the server learn which one is the most effective single solver, and modify accordingly its probability to be chosen.

Some extra announcement

- The **Ariadna** scheme of the ESA Advanced Concepts Team makes possible to collaborate with European universities.
- Global Trajectory Optimization Competition (initiated by D. Izzo, ESA/ACT).
- `www.esa.int/act`