Accelerated Numerical Optimization with Explicit Consideration of Model Constraints

Lucia Damiani[1], Ariel I. Diaz[2], Javier Iparraguirre[2] and Aníbal M. Blanco[1]

[1]PLAPIQUI (CONICET-UNS)
[2]UTN, Bahía Blanca

ldamiani@plapiqui.edu.ar
arielivandiaz@gmail.com
j.iparraguirre@computer.org
ablanco@plapiqui.edu.ar

utn bhi

PLAPIQUI

About numerical optimization

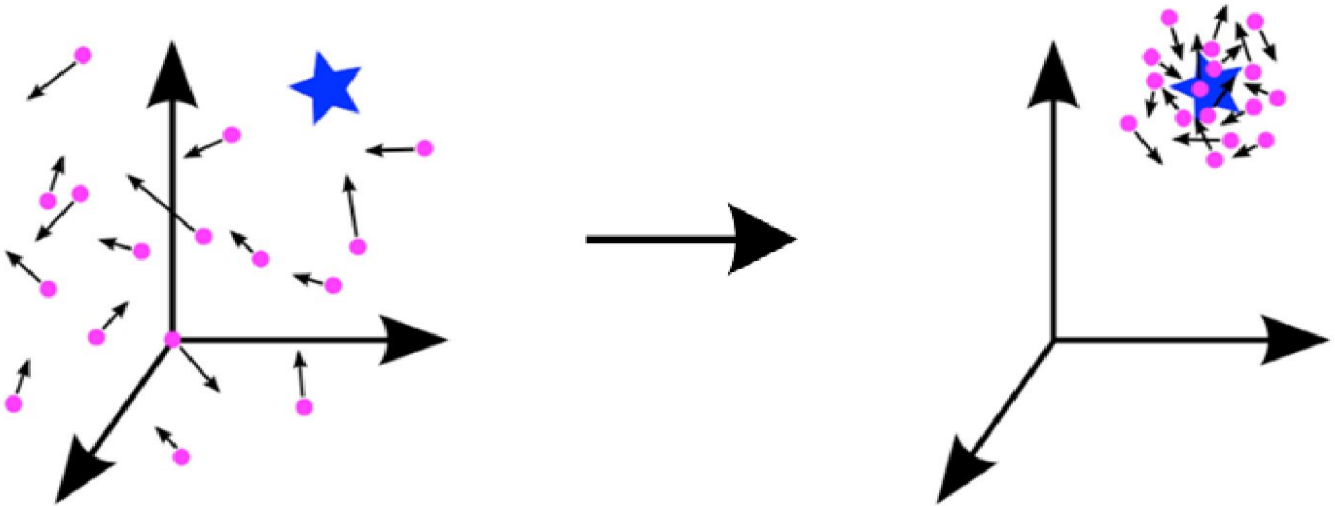
- Numerical optimization has an outstanding role in science and engineering.
- The goal consist on finding a set of variables that optimizes an objective.
- It is a challenging task if variables are combined in non-linear functions and there are additional constraints.
- There two main approaches to optimization: deterministic or metaheuristic optimization.
- Particle Swarm Optimization (PSO) is a popular metaheuristic.

Our proposal

- We present a GPU based implementation of the PSO algorithm.
- The major contribution is the explicit consideration of a constraint handling methodology.
- The implementation is based on the PyCUDA platform.
- Performance is tested on a set of typical benchmark problems.

Particle Swarm Optimization (PSO)

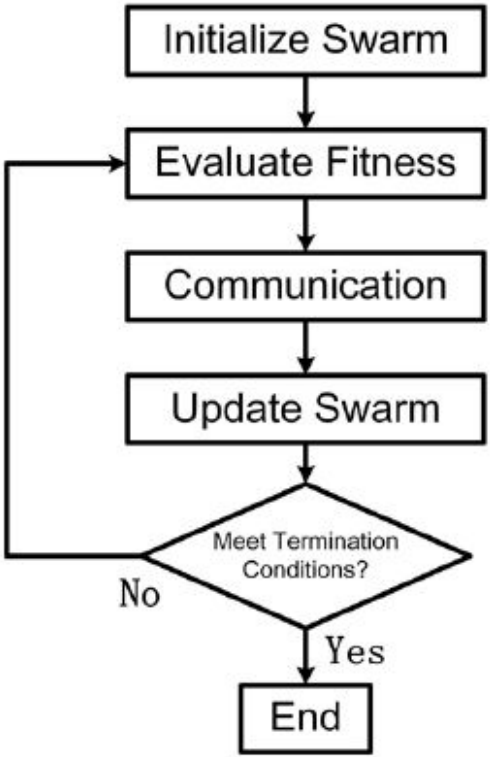
PSO



Initial state

Solution

Basic algorithm

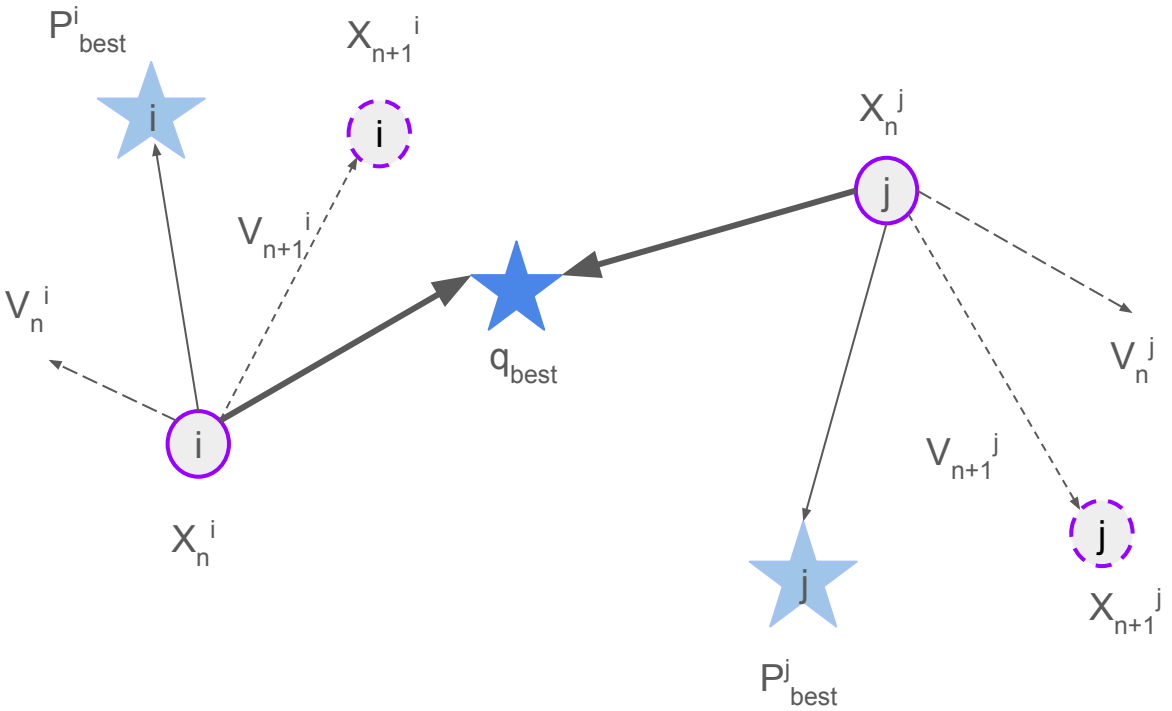


General formulation

$$\text{Min}_{\mathbf{x}}\ f(\mathbf{x}),\ \text{st.}\mathbf{h}(\mathbf{x})=\mathbf{0},\ \mathbf{g}(\mathbf{x})\leq\mathbf{0},\ \mathbf{x}\epsilon\mathbf{X}$$

f(.): objective function, h(.): equality constraints, g(.): inequality constraints

Position update



Original velocity

Velocity towards q_{best}

Velocity towards p_{best}

Resultant velocity

utn bhi

PLAPIQUI

Iteration step update

- Speed update

$$\mathbf{v}_i^{k+1} = w^k \mathbf{v}_i^k + c_1 \mathbf{r_1}^k (\mathbf{p}_i^k - \mathbf{x}_i^k) + c_2 \mathbf{r_2}^k (\mathbf{q}^k - \mathbf{x}_i^k)$$

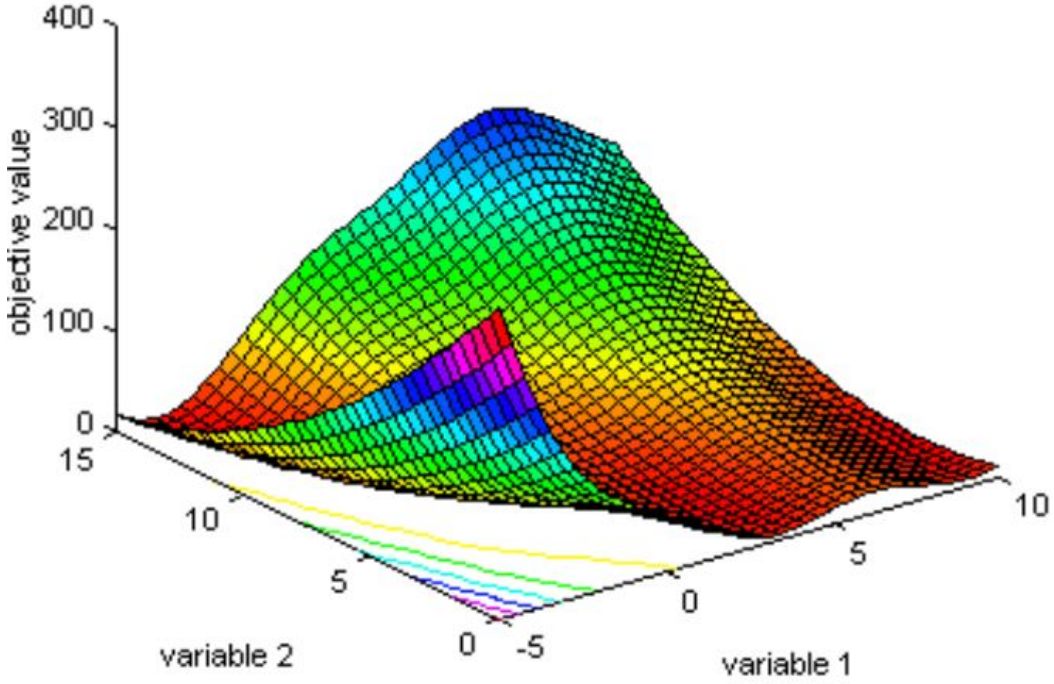
Inertia — acceleration

- Position update

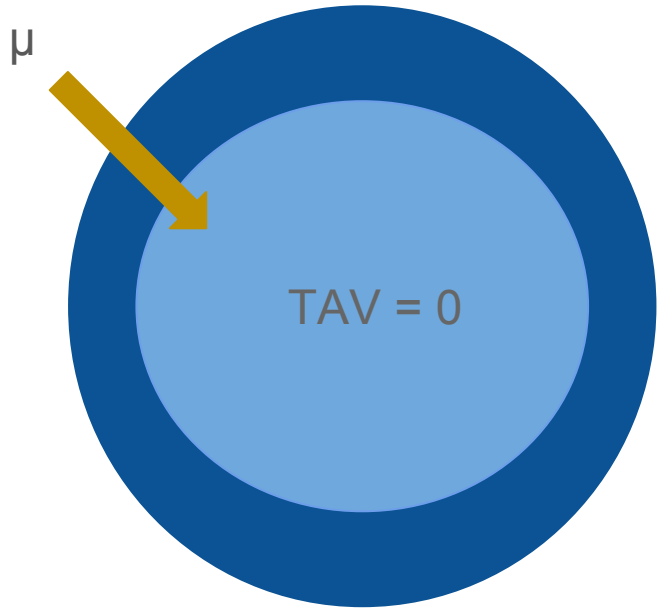
$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1}$$

x: position, v: speed, p: best position of particle, q: best position ever, r: random vector, k: actual iteration

Objective function example (without constraints)



Constraints handling



- Original region
- Relaxed region
- μ: value of relaxed constraints
- Total Absolute Violation (TAV)

Constraints handling

- Total Absolute Violation (TAV)

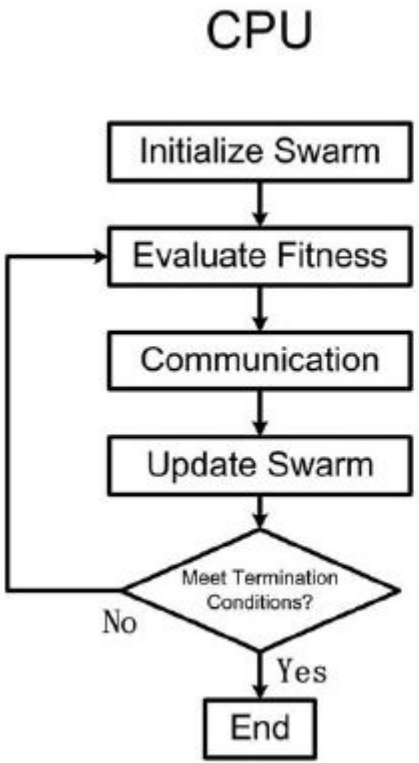
$$TAV_i^k = \sum_n \left| h_n(\mathbf{x}_i^k) \right| + \sum_m \max(0, g_m(\mathbf{x}_i^k)) \quad i=1, \ldots, N$$

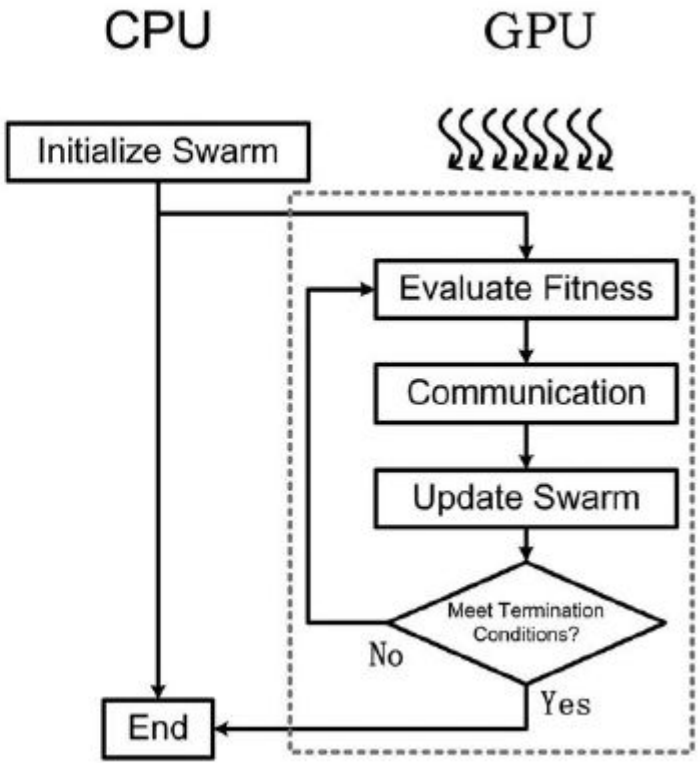
- If TAV_i^k is lower than a relaxation value μ, particle i is considered feasible in iteration k
- Parameter μ is dynamically reduced according to:

$$\mu^{k+1} = \mu^k (1 - F_F^k / N)$$

Implementation

All GPU parallel model





PSO implementation

- A serial version (Python + NumPy) and an accelerated version (PyCUDA) were implemented.
- PyCUDA provides the productivity of an interpreted language and the efficiency of hardware accelerators.
- Except for the random number initializations, all steps involved in the PSO computation are performed on the GPU.
- One particle is assigned to a CUDA block and the number of threads per block equals the dimension of the problem under study.

Pseudocode for PSO

1. Randomly initialize $\mathbf{x}_i^0$ and $\mathbf{v}_i^0$ for i=1,…,N
2. For k=1,…,kmax repeat
   a. Evaluate $\mathbf{x}_i^k$
   b. Evaluate $f(\mathbf{x}_i)$ and TAV_i^k
   c. Identify $\mathbf{p}_i^k$ (local best) according to the feasibility criteria
   d. Identify $\mathbf{q}^k$ (global best) according to the feasibility criteria
   e. Evaluate $\mathbf{v}_i^k$

Results

Results

- Problems taken from models used in the literature (Liang et al., 2006)
- In all cases w=0.9, c_1=2.0, c_2=2.0, N=50 and k_{max}= 2500
- It is reported that PSO performance becomes insensitive to swarms larger than 30-60 particles for most problems.
- T_{CPU} is the time that the serial version required to achieve a solution.
- T_{GPU} is the time required by the accelerated version.
- Each reported value is the average time of 20 independent runs.

$$\text{Speedup} = T_{CPU}/T_{GPU}$$

Systems

	System 1	System 2
Processor	Intel Skylake Core i7 6700 3.4 GHz	AMD FX-4100 Quad-Core
CPU RAM	8GB DDR4	8GB of DDR3
GPU	GTX480	GeForce GTX TITAN X GPU
OS	Linux Mint 18.1	Linux Mint 18.1

P# Ref.	D	n	m	System 1		System 2	
				TCPU(sec)	Speedup	TCPU(sec)	Speedup
1*	13	0	9	4.76	28.12	13.13	44.3
3	10	1	0	3.27	25.94	8.99	41.86
4	5	0	6	4.24	40.39	12.21	60.3
5	4	3	2	2.93	28.92	8.7	43.34
7*	10	0	8	8.17	57.63	24.37	101.64
8	2	0	2	1.69	17.91	4.88	24.78
9	7	0	4	6.07	50.5	16.1	73.65
11	2	1	0	0.94	10.02	2.68	14.4
13*	5	3	0	2.22	22.58	6.1	32.55
14*	10	3	0	8	47.13	22.08	84.12
18	9	0	13	5.44	42.06	14.52	64.2
24	2	0	2	2.02	22.78	5.5	30.22

More about results

- An average of 32x for system 1 and 51x for system 2.
- In all cases, the algorithm converged to feasible solutions.
- Success rate (runs that converged to the known global solution) was superior to 75% in most cases.
- Problems were feasible but sub-optimal solutions were achieved in less than 75% are marked with an asterisk.
- The focus of this study was on the speedup of the parallelized version.
- Standard parameterization was adopted in all cases.

Conclusions and future work

Conclusions and future work

- A accelerated PSO with explicit consideration of constraints was presented.
- TAV methodology handles equality and inequality constraints.
- Satisfactory performance was obtained, both in terms of speedups and success rate.
- Aspects to improve in terms of GPU implementation: parallel reduction and better user of shared memory
- Improvements to PSO algorithm implementation: dynamic variation of parameter w and alternative termination criteria.

Thank you!

Questions?