MASSIVE PARALLELISM OF HARMONY MEMORY IN N-DIMENSIONAL SPACE

DR. SHAFAATUNNUR HASAN
GPU PRINCIPAL RESEARCHER, UTM BIG DATA CENTRE,
IBNU SINA INSTITUTE FOR SCIENTIFIC & INDUSTRIAL RESEARCH
UNIVERSITI TEKNOLOGI MALAYSIA, 81310 SKUDAI JOHOR

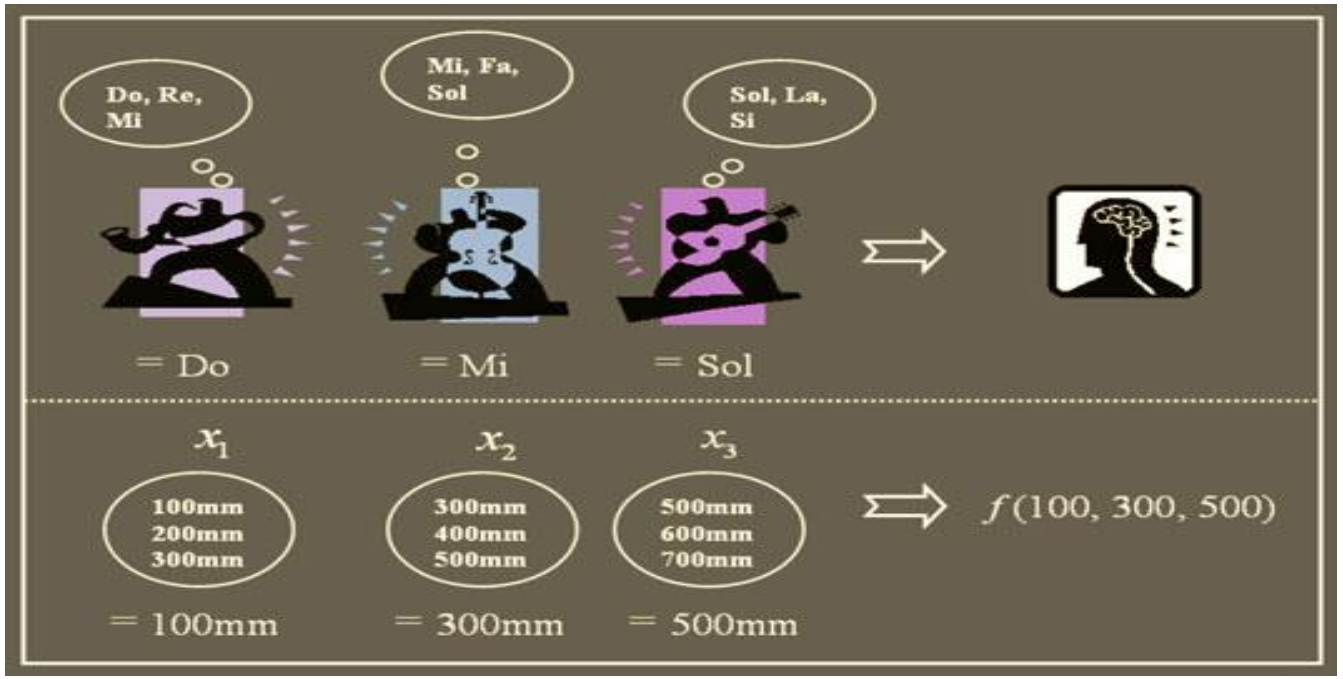
PROFESSOR DR. SITI MARIYAM SHAMSUDDIN
DIRECTOR, UTM BIG DATA CENTRE,
IBNU SINA INSTITUTE FOR SCIENTIFIC & INDUSTRIAL RESEARCH
UNIVERSITI TEKNOLOGI MALAYSIA, 81310 SKUDAI JOHOR

What we are doing??

Developing the GPU-based
Aesthetics Searching Algorithm for Big N-Dimensional Problems

Harmony Search Algorithm (HSA)??

Population-based metaheuristic optimization algorithm, that imitates the music improvisation process where the musicians improvise their instruments' pitch by searching for a perfect state of harmony.



Analogy between improvisation and optimization

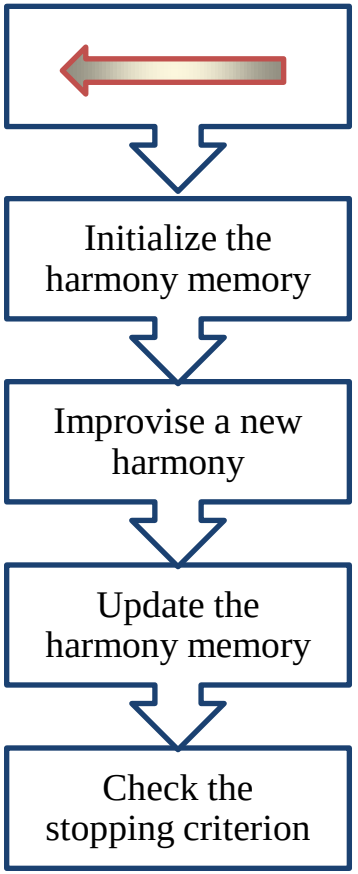
HSA Notations

Initialize the problem and algorithm parameters

In Step 1, the optimization problem is specified as follows:

$$Minimum \quad F(x) \quad \text{subject to} \quad x_i \in X_i = 1,2,\ldots,N$$

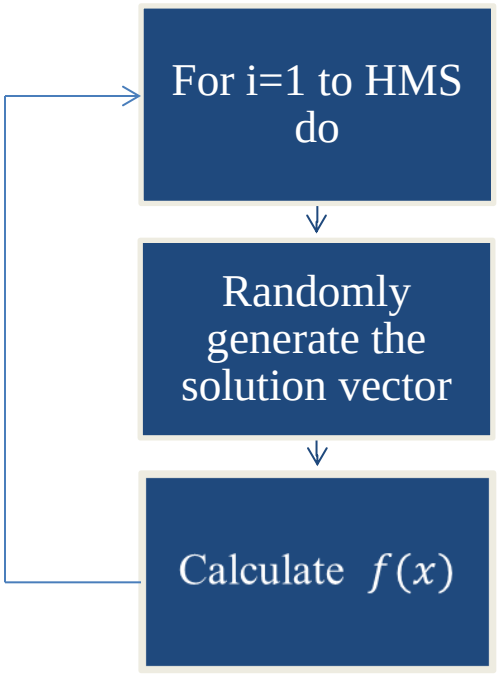
$F(x)$	• *Objective function*
X_i	• *Decision Variable*
N	• *Number of Decision Variables*
HMS	• *Number of solution vectors in HM*
$HMCR$	• *HM Considering Rate*
PAR	• *Pitch Adjusting Rate*
bw	• *Distance bound wide*
NI	• *Number of solution vector generations*



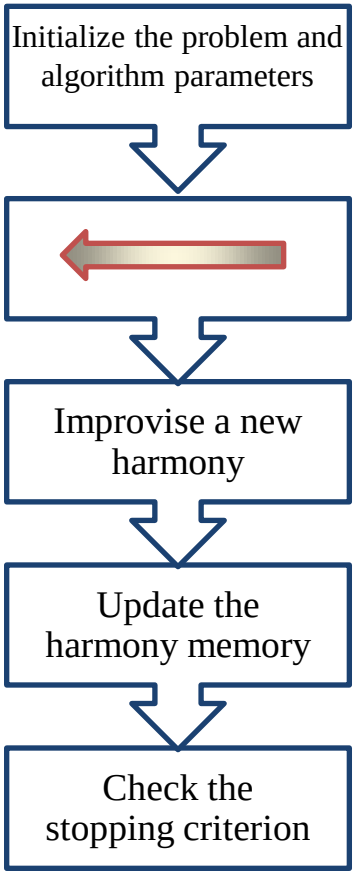
HSA Implementation

Initialize the harmony memory (HM)

In Step 2, the HM matrix is filled with as many randomly generated solution vectors as the HMS.



$$HM = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_{N-1}^1 & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_{N-1}^2 & x_N^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{HMS-1} & x_2^{HMS-1} & \cdots & x_{N-1}^{HMS-1} & x_N^{HMS-1} \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_{N-1}^{HMS} & x_N^{HMS} \end{bmatrix}$$

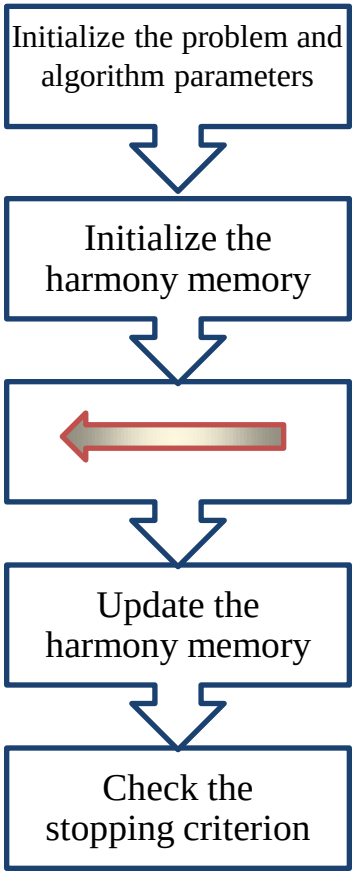


HSA Implementation

Improvise a new harmony

A new harmony vector is generated based on three rules

- memory consideration
- pitch adjustment
- random selection



HSA Implementation

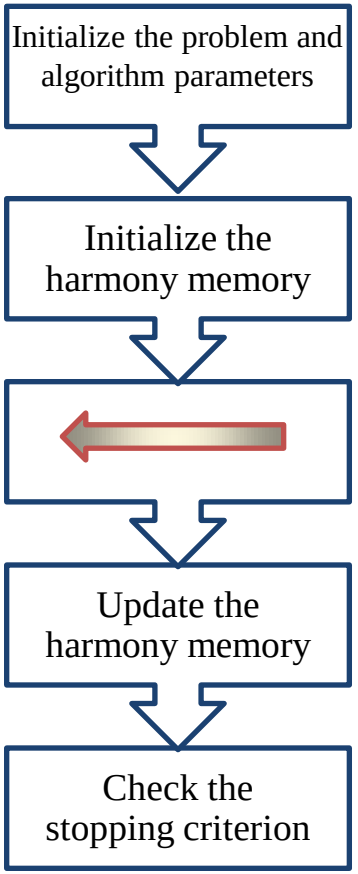
Improvise a new harmony

In Step 3, HM consideration, pitch adjustment or random selection is applied to each variable of the new harmony vector in turn.

$$x_i' \leftarrow \begin{cases} x_i' \in \{x_i^1, x_i^2, \ldots, x_i^{HMS}\} & \text{with probability HMCR,} \\ x_i' \in X_i & \text{with probability } (1 - \text{HMCR}) \end{cases}$$

$$\text{Pitch adjusting decision for } x_i' \leftarrow \begin{cases} \text{Yes} & \text{with probability PAR,} \\ \text{No} & \text{with probability } (1 - \text{PAR}). \end{cases}$$

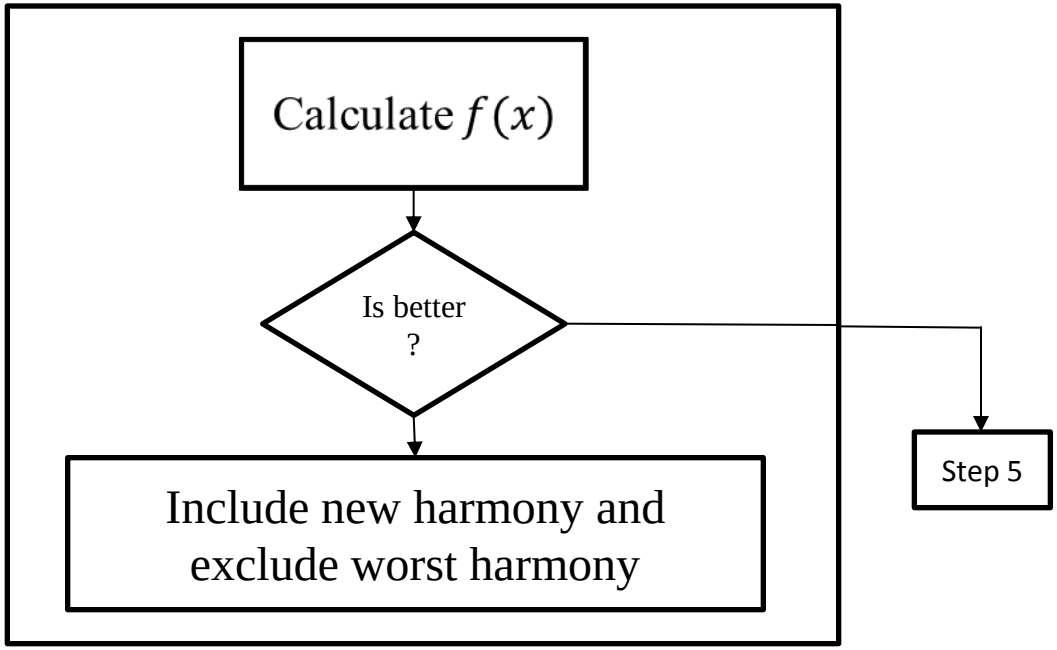
$$x_i' \leftarrow x_i' \pm \text{rand}() * \text{bw}$$

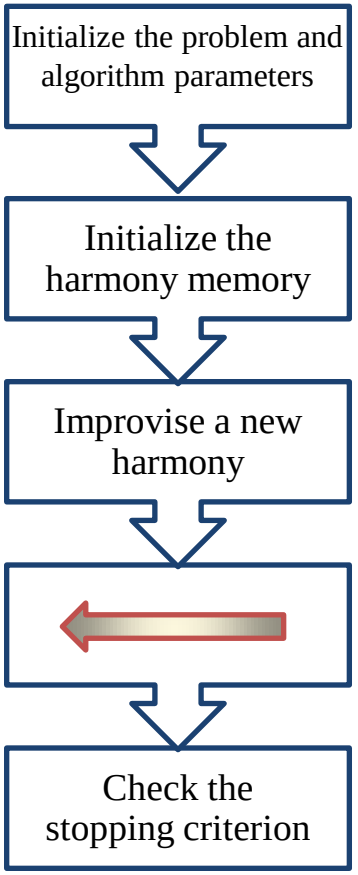


HSA Implementation

Update the harmony memory

If the new harmony vector is better than the worst harmony in the HM, the new harmony is included in the HM and the existing worst harmony is excluded from the HM.





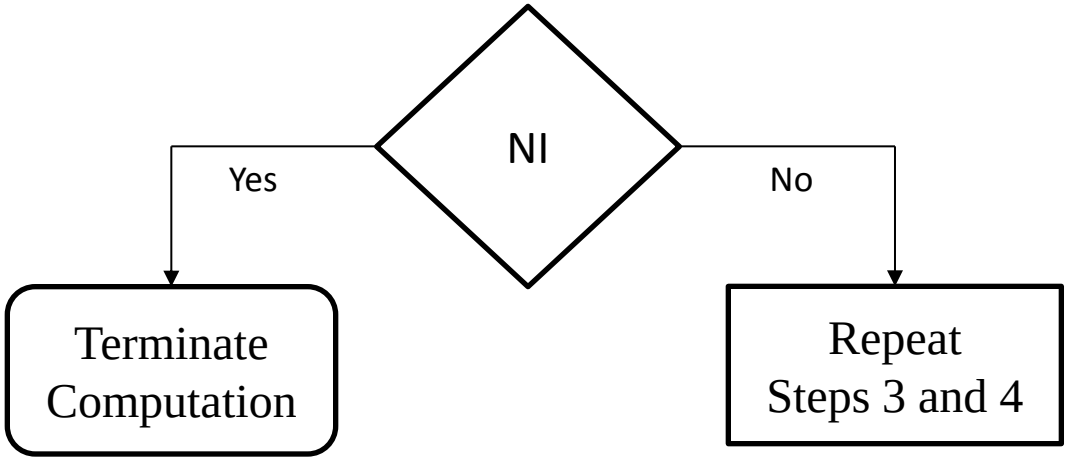
Harmony Memory (HM)

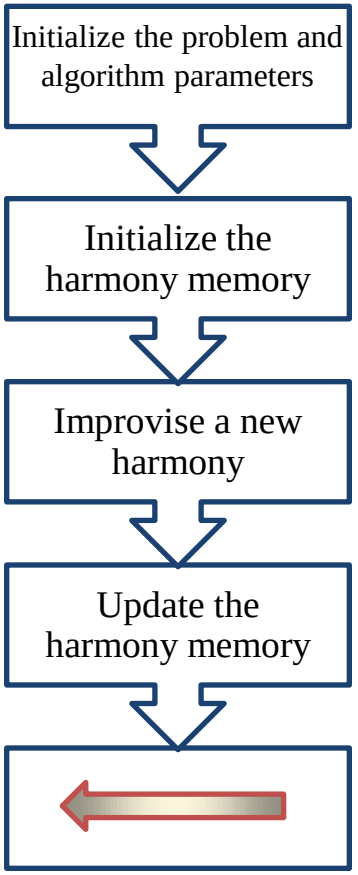
	New Harmony Memory										Cost
1	2	-1	-2	0	4	-2	-3	0	0	2	**42**
2	2	3	1	2	-2	-1	6	4	3	1	85
3	2	4	5	6	3	-3	-1	0	-1	0	101
4	1	3	-2	-5	4	-5	-3	2	1	3	103
5	-2	5	5	4	-6	-5	-1	2	1	0	137
6	4	5	1	-3	-4	0	7	0	3	4	**141**

HSA Implementation

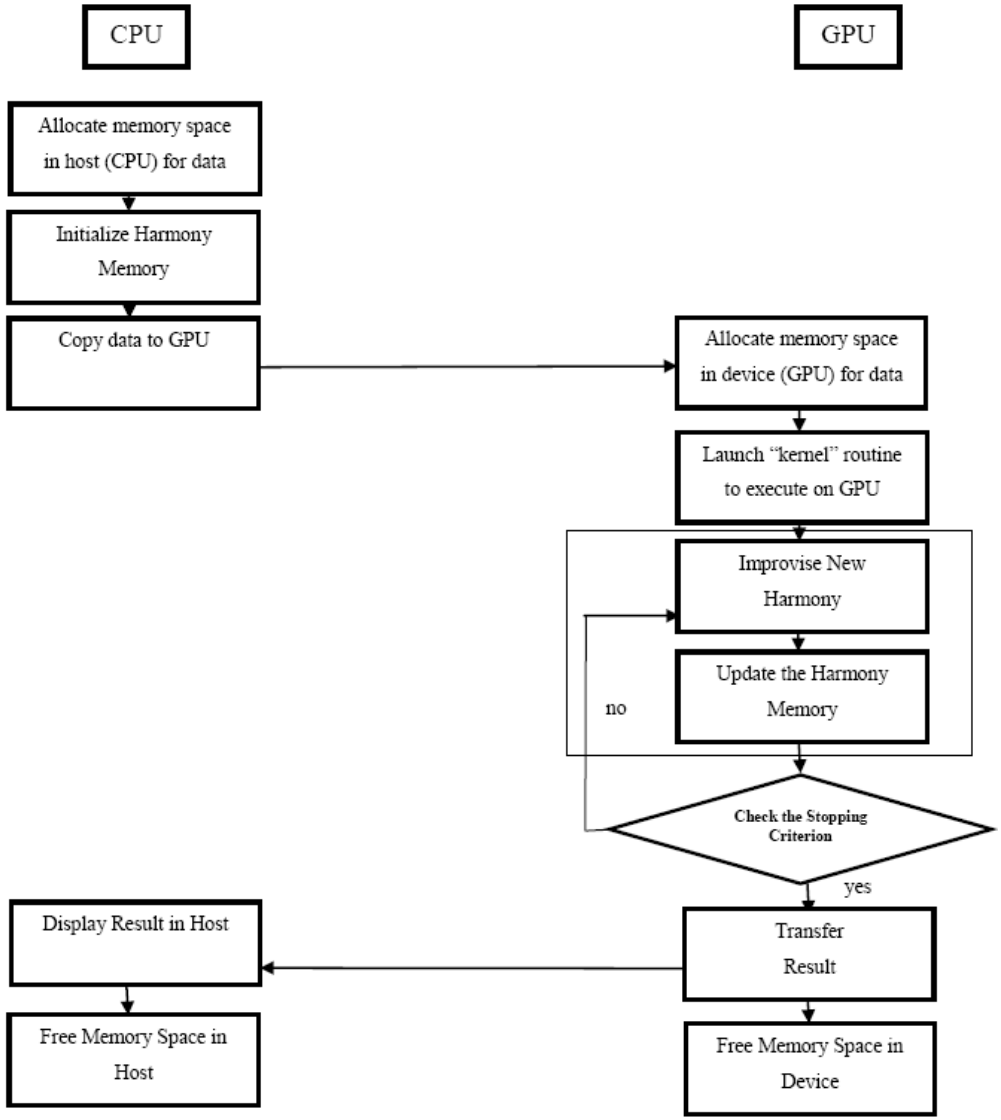
Check the stopping criterion

If the stopping criterion (maximum number of improvisations) is satisfied, computation is terminated. Otherwise, Steps 3 and 4 are repeated.





Where the Massive Parallelism of Harmony Memory being implemented???

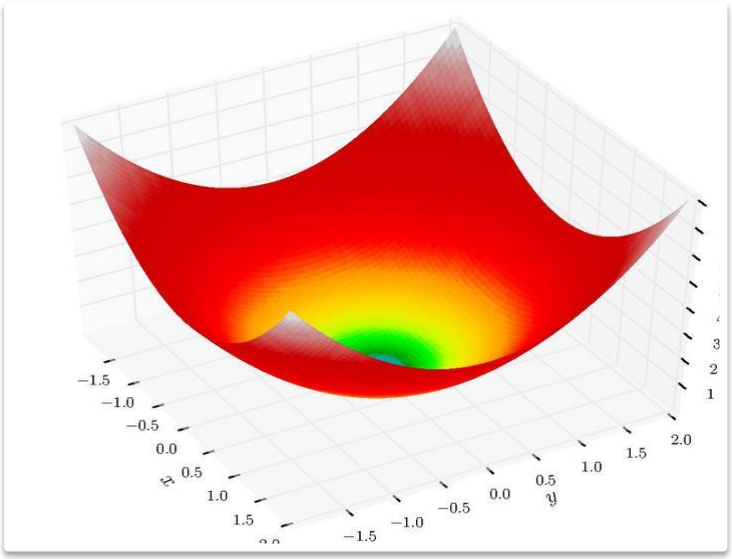


Harmony Search Parameters

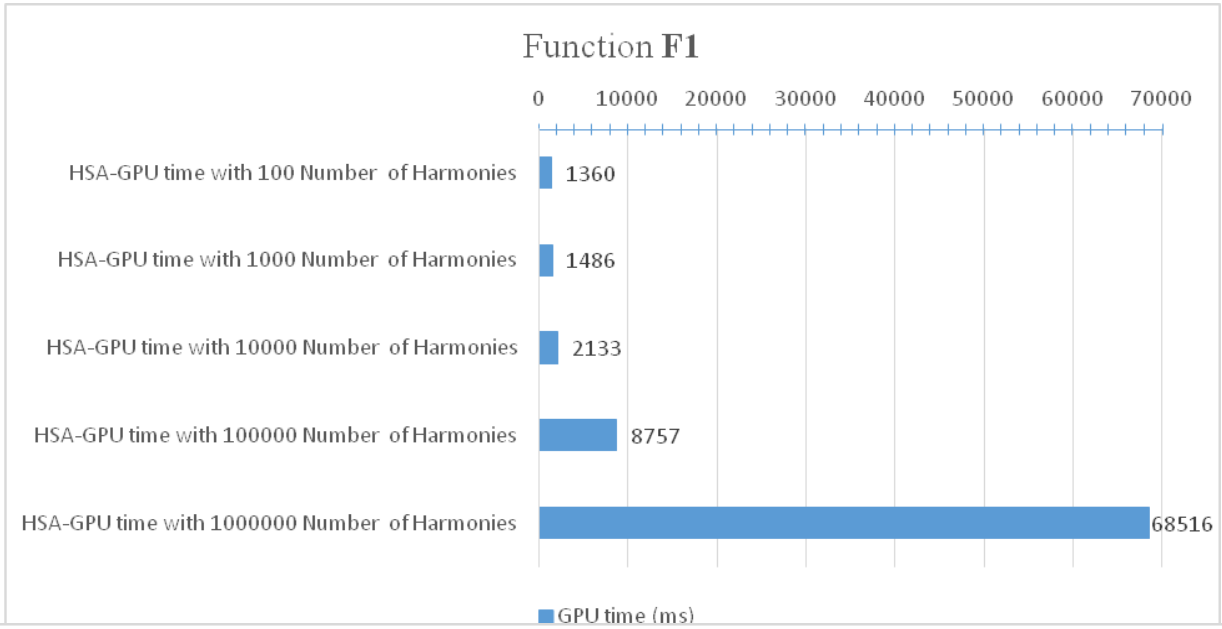
Decision Variables Lower Bound	VarMin	-100
Decision Variables Upper Bound	VarMax	100
Number of Decision Variables	nVar	10000
Maximum Number of Iterations	MaxIt	100000
Harmony Memory Size	HMS	20
Number of New Harmonies	nNew	50
Harmony Memory Consideration Rate	HMCR	0.2
Pitch Adjustment Rate	PAR	0.1

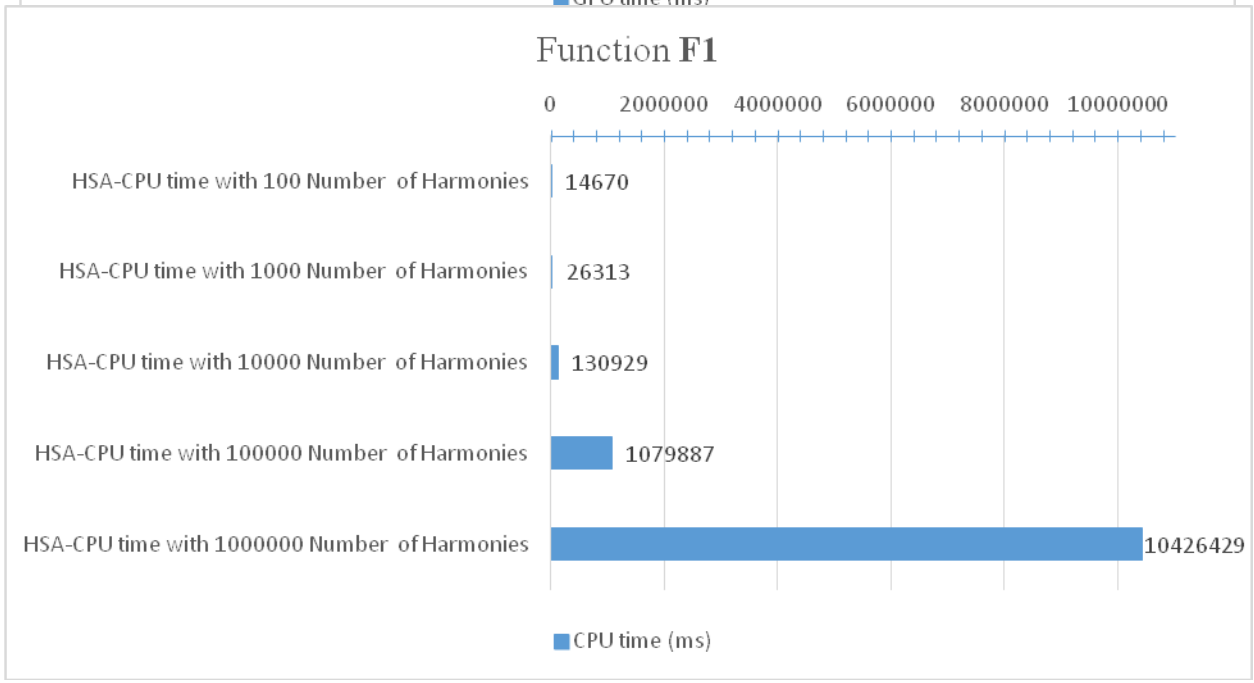
Experimental Results on Benchmark Functions

Test Function	[Range]
$F1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]n$



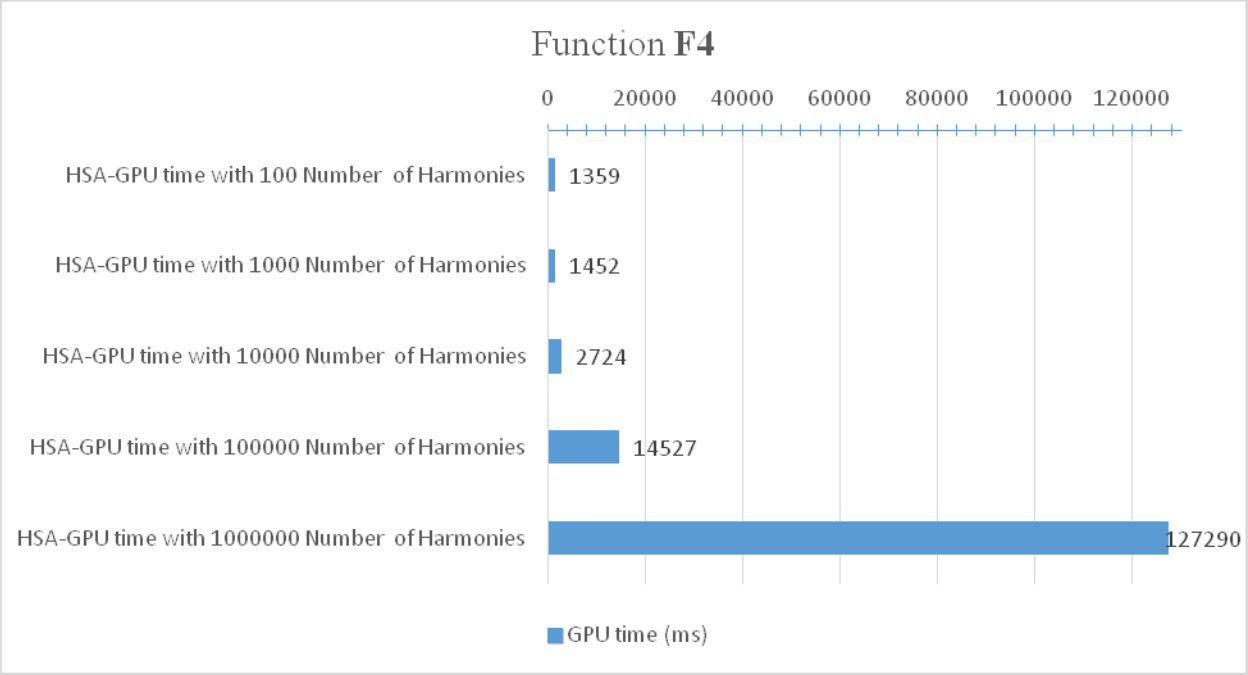
HSA-CPU and HSA-GPU on benchmark function F1

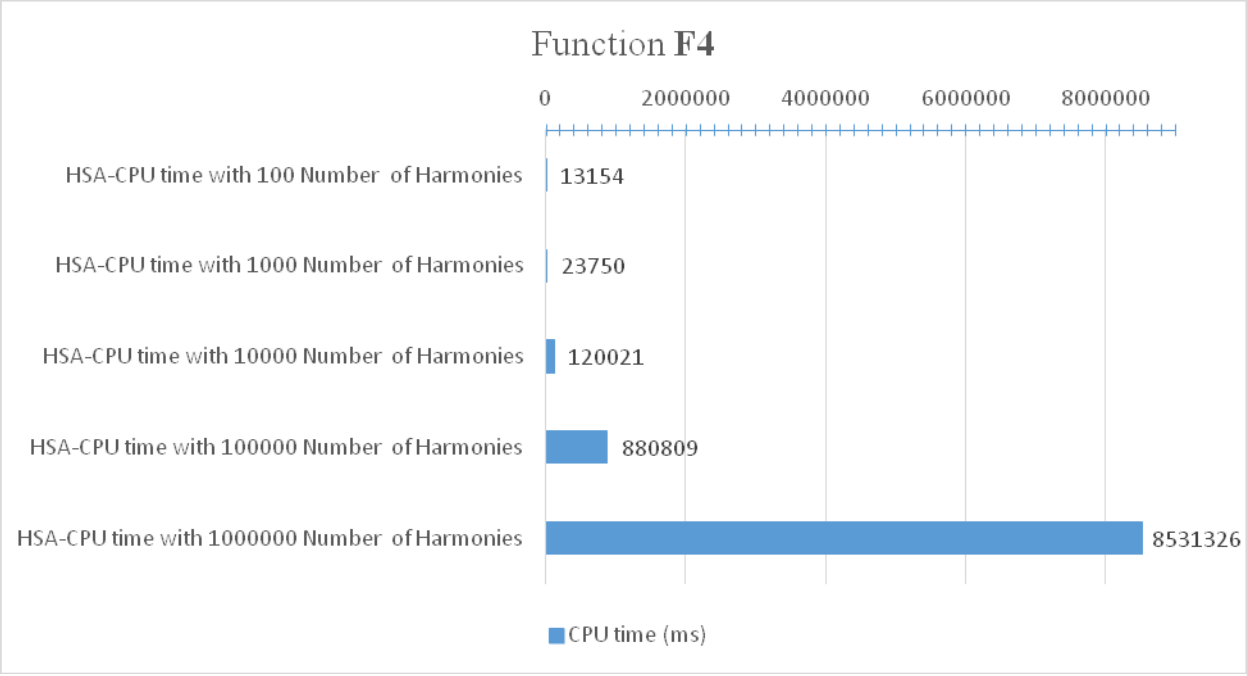




HSA-CPU and HSA-GPU on benchmark function F4

$$F4(x) = \sum_{i=1}^{n-1} [(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$





Conclusion

We presented our work on the implementation of GPU for big N-dimensional harmony memory of aesthetics searching algorithms, i.e., harmony search algorithm. Experiments have shown with big number of harmonies, the GPU is always faster than the CPU. These findings can be implemented on big dimensional of real data for searching the optimal solutions aesthetically, i.e., domain oriented accordingly.

THANK YOU

JOIN THE CONVERSATION

#GTC15