Information Collection Strategies in Memetic Cooperative Neuro-evolution for Time Series Prediction

Gary Wong[1], Anurag Sharma[2], Rohitash Chandra[3]

School of Computing Science, Information and Mathematical Sciences,
The University of the South Pacific, Fiji[1,2].
Centre for Translational Data Science, The University of Sydney, Australia[3]

gary.wong.fiji@gmail.com[1] anuraganand.sharma@usp.ac.fj[2] c.rohitash@gmail.com[3]

Outline

- Introduction
- Background
- Proposed Methods
    1. Sequential Collection
    2. Concurrent Collection
- Experiments / Results
- Discussion & Conclusion

Introduction

Research Objective

- Explore memetic cooperative neuro-evolution methods that features the storage of global solutions/information for local search refinement

Contributions

- Methods that retain features of global search that would otherwise be lost in a single meme sharing scheme
- Improve prediction accuracy
- Pathway for future work in memetic cooperative neuro-evolution

Background: Evolutionary Algorithms

- Evolutionary Algorithms (EA) are successful search and optimization techniques
- EAs used for training neural networks are known as Neuro-Evolutionary algorithms that provide a diversity of solutions
- Drawback is convergence costs as they are black-box optimization methods
- Gradient training methods provide solution intensification and are computationally cheap with regular occurrences of premature convergence

Background: Memetic Algorithms

- Memetic algorithms (MAs) are meta-heuristics that balance exploration and exploitation
- The term ‘meme’ refers to cultural information as opposed to genes
- MAs are capable of tackling largescale real-world problems with better efficiency than canonical evolutionary
- Global search provides diversity while local search provides refinement

Background: Memetic Neuro-evolution

- Previous work implemented a single meme synergy between Cooperative Coevolution and Stochastic Gradient Descent
- Throughout the memetic process, Global and Local Search will take turns refining a single solution according to below parameters
  - LSF : Local Search Frequency
    - How often to apply local search (save a meme in this study)
  - LSI : Local Search Intensity
    - How much refinement time

Proposed Information Collection Strategies

- Information collection refers to the storage of global solutions or memes. How the memes are stored
- This study explores 2 methods;
  1. Sequential Meme Collection
  2. Concurrent Meme Collection

Method 1. Sequential Meme Collection

- The meme collection strategy extracts and concatenates the fittest individuals from all the subpopulations at a uniform rate in sequential order during the entire phase of evolution
- Uses Adaptive LSI – see following slide.

Method 1. Adaptive Local Search Intensity

- Each meme will have different refinement durations according to when the meme was saved.
- Those memes collected <u>closer to the end of evolution </u>will have less refinement time than those collected earlier in the evolution cycle. <u>This is to ensure fair refinement time</u>.
  - T_{max} : Max evaluations allowed
  - $T_{elapsed}$: Evaluations so far

$$\texttt{calculateLSI()} \quad = lsf - \frac{(lsf \times \Gamma_{elapsed})}{\Gamma_{max}}$$

Method 1. Sequential Meme Collection

1. **Initialization Step**
   - Initialize the subpopulations of CC that represent the weights of the neural network
   - Assign fitness
2. **Meme Collection Step**
   - Perform global search for <u>max evaluation time</u>
     - After every LSF evaluations, save the fittest CC solution to the meme collection
3. **Refinement Step**
   - Each meme is refined with varying LSI
   - Compare accuracy of each meme in collection and save the best meme as the current optimal solution

Symbols

Variable	Description	Variable	Description
α	Mutation Rate	ε_t	Test Accuracy.
μ	Population Size	lsf	LS Frequency.
Γ_{max}	Max Evaluations.	lsi	LS Intensity.
$\Gamma_{elapsed}$	Total Evaluations.	δ^*	Best Meme.
λ	Learning Rate.	L	sp Set.
γ	Optimization Time	sp	Sub-population.
i	# Input Neurons.	w_{min}	Lower Weight Limit.
h	# Hidden Neurons.	w_{max}	Upper Weight Limit.
o	# Output neurons.	mc	Meme Collection.
ε_{min}	Required Minimum ε_t.	$e_{counter}$	Elapsed Counter
ec	Elapsed Times	n	Top # of Memes
F	Sub-population Fitness		

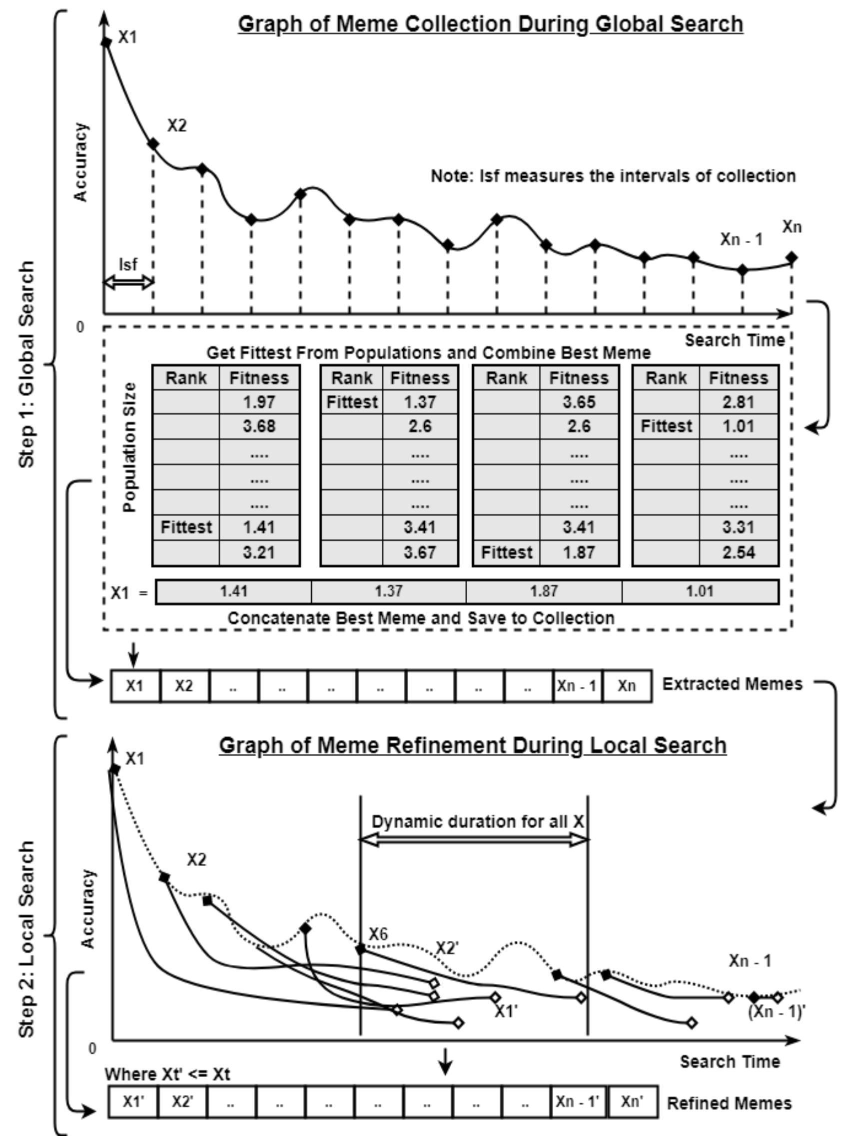
Method 1. Sequential Meme Collection

Algorithm 1: *Sequential* Strategy

Step 1: Population Initialization
s = h + o; $\Gamma_{elapsed} = 0$;
for $y \in \{1,.., s\}$ **do**
    $L(y)$ = rand(μ, w_{max}, w_{min});
    $F(y)$ = eval($L(y)$);
    $\Gamma_{elapsed} = \Gamma_{elapsed} + |L(y)|$;

Step 2: Collection (Global Search)
while $\Gamma_{elapsed} < \Gamma_{max}$ **do**
    while $(\Gamma_{elapsed} - e_{counter}) < (lsf + 1)$ **do**
        for $y \in \{1,..,s\}$ **do**
            for $j \in \{1,..,\mu\}$ **do**
                $L(y)$ =evolve($L(y)$); $\Gamma_{elapsed}$ += μ x $(\gamma + 1)$;
    δ^* = getBestSolution(L);
    $mc = mc \cup \delta^*$;
    $ec = ec \cup \Gamma_{elapsed}$;

Step 3: Refinement (Local Search)
for $u \in \{1,..,|mc|\}$ **do**
    $\Gamma_{elapsed} = ec(u)$;
    $lsi = lsf - \frac{(lsf \times \Gamma_{elapsed})}{\Gamma_{max}}$
    ε_t = bpnn($\delta^*, \lambda, mc(u), lsi$);
evalMemes(mc);



Method 2. Concurrent Meme Collection

- This meme collection strategy collects a list of the fittest individuals from the subpopulations <u>at the same time</u> at the end of the exploration phase

Method 2. Concurrent Meme Collection

1. **Initialization Step**
   - Initialize the subpopulations of CC that represent the weights of the neural network
   - Assign fitness
2. **Meme Collection Step**
   - Perform global search for <u>max evaluation time</u>
   - At the end of max evaluations, save the best *N* solutions from CC populations into the meme collection
3. **Refinement Step**
   - Each meme is refined with same LSI
   - Compare accuracy of each meme in collection and save the best meme as the current optimal solution

Method 2. Concurrent Meme Collection

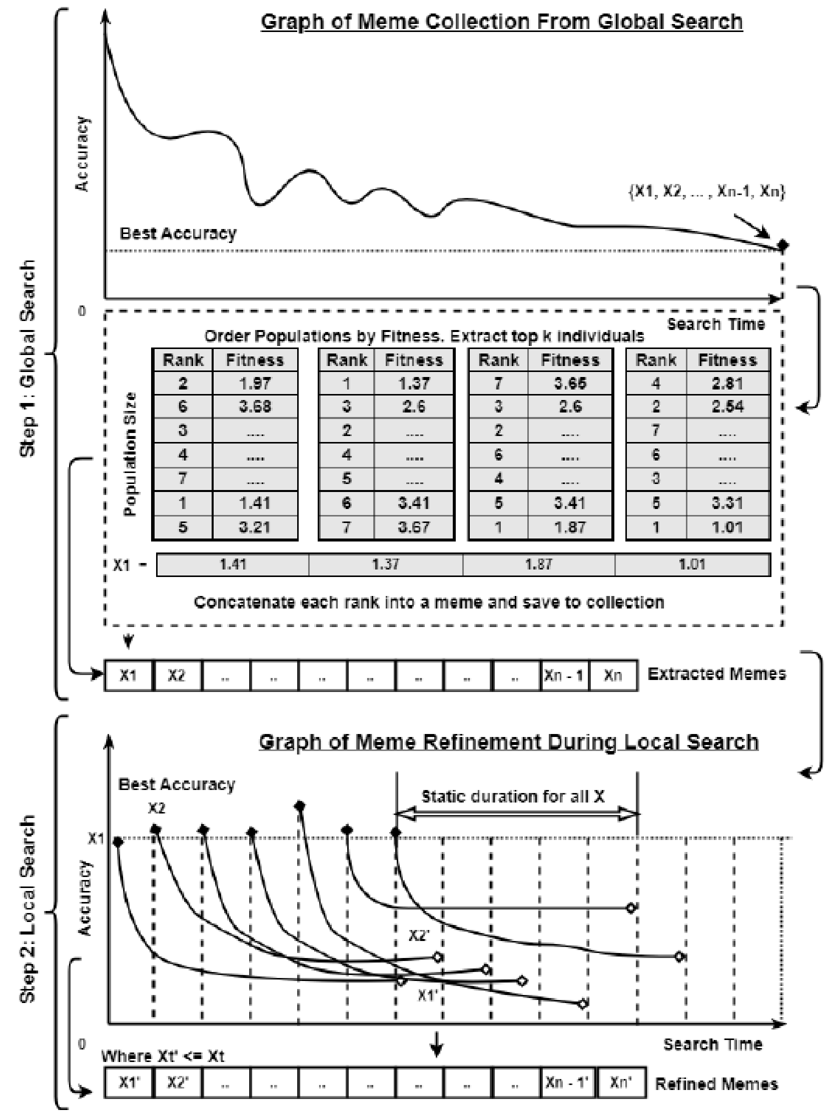
Algorithm 2: *Concurrent* Strategy

Step 1: Population Initialization
s = h + o; $\Gamma_{elapsed} = 0$;
for $y \in \{1,.., s\}$ **do**
  $L(y)$ = `rand` (μ, w_{max}, w_{min});
  $F(y)$ = `eval` $(L(y))$;
  $\Gamma_{elapsed} = \Gamma_{elapsed} + |L(y)|$;

Step 2: Collection (Global Search)
while $\Gamma_{elapsed} < \Gamma_{max}$ **do**
  while $(\Gamma_{elapsed} - e_{counter}) < (lsf + 1)$ **do**
    for $y \in \{1,..,s\}$ **do**
      for $j \in \{1,..,\mu\}$ **do**
        $L(y)$ =`evolve` $(L(y))$; $\Gamma_{elapsed}$ += μ x $(\gamma + 1)$;

$n = \dfrac{\Gamma_{elapsed}}{lsf}$;
L = `orderAsc` (L);
mc = `getTopSolutions` (L, n);

Step 3: Refinement (Local Search)
for $u \in \{1,..,\mu\}$ **do**
  ε_t = `bpnn` $(\delta^*, \lambda, mc(u), \ lsi)$;
`evalMemes` (mc);



Benchmark Problems

- We apply the proposed methods to 5 time series benchmark problems
  1. Sunspot Time Series Dataset
  2. Santa Fe Laser Time Series Competition Data
  3. Mackey Glass Dataset
  4. Lorenz Dataset
  5. Taiwan Trading Index

Dataset	Samples	Dim. and Time Lag	*Train/Val/Test*
Laser	1000	D: 3, T: 2	194 / 166 / 166
Lorenz	1000	D: 3, T: 2	299 / 99 / 99
MackeyGlass	1000	D: 3, T: 2	299 / 99 / 99
Sunspot	2000	D: 5, T: 3	399 / 132 / 132
TWIExchange	304	D: 5, T: 1	177 / 55 / 55

Experiment Setup

- *CC Population size: 300*
- *Max Evaluations: 100,000*
- *SGD Learning Rate: 0.1*
- *Method 1 - Seq. Strategy*
  - *LSF: Save meme at every 5000 evaluations*
  - *LSI: Adaptive*
- *Method 2 - Con. Strategy*
  - *LSF: Save meme at the end of evolution*
  - *LSI: 2000 epochs*
- *Feed-forward neural network used*

Measuring Accuracy

- *Fitness is measured via the Root Mean Squared Error (RMSE)*
  - Yi : Actual Output
  - Yi^: Predicted output

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2)}$$

Results and Analysis

PERFORMANCE FOR THE **SUNSPOT** PROBLEM

Method	Mean ε_t	Best ε_t	Worst ε_t	Eval. $\Gamma_{elapsed}$
Sequential	0.0127696	0.0107341	0.0195412	205,039
Concurrent	0.019353	0.014647	0.02512103	121,200
MCNE [22]	0.0478444	0.0246412	0.0671124	100,000

PERFORMANCE FOR THE **TWI EXCHANGE** PROBLEM [31]

Method	Mean ε_t	Best ε_t	Worst ε_t	Eval. $\Gamma_{elapsed}$
Sequential	0.0394227	0.035412	0.0412148	272,318
Concurrent	0.0397674	0.0363142	0.0432614	121,200
MCNE [22]	0.0852743	0.0745214	0.0912457	100,000

PERFORMANCE FOR THE **SANTA FE LASER** PROBLEM [29]

Method	Mean ε_t	Best ε_t	Worst ε_t	Eval. $\Gamma_{elapsed}$
Sequential	0.069533	0.0571243	0.072142	269,421
Concurrent	0.0768557	0.0634781	0.0793412	121,200
MCNE [22]	0.194982	0.147142	0.2188464	100,000

Results and Analysis

PERFORMANCE FOR THE MACKEY GLASS PROBLEM [28]

Method	Mean ε_t	Best ε_t	Worst ε_t	Eval. $\Gamma_{elapsed}$
Sequential	0.00454625	0.00192641	0.0057482	271,031
Concurrent	0.00595269	0.00320041	0.00671213	121,200
MCNE [22]	0.0252556	0.012321489	0.03451222	100,000

PERFORMANCE FOR THE LORENZ PROBLEM [30]

Method	Mean ε_t	Best ε_t	Worst ε_t	Eval. $\Gamma_{elapsed}$
Sequential	0.073145	0.071354	0.078321	260,668
Concurrent	0.34457	0.32148871	0.3811421	121,200
MCNE [22]	0.0747062	0.075321	0.0793321	100,000

Discussion and Conclusion

- The sequential strategy had the best generalization performance in all the problems tested
- Adapting LSI seems to be useful in providing a better balance for solutions collected at different points of the global search
- Improved accuracy than the standalone methods but computationally expensive
- Using collected information/memes with later refinement can be useful in a memetic structure
- Refining solutions collection during evolution seems to be a better approach than those collected post evolution
- Future work can implement multiple local search methods on the pool of memes with a metaheuristic for controlling when and how to apply each local search method
- Other work can try reversing the roles where global search would provide refinement

The End. Thank you