Evolution of Symbolic Ensemble Forecast Models for Quantitative Precipitation

Amanda S. Dufek, Douglas A. Augusto, Pedro L.S. Dias and Helio J.C. Barbosa

National Laboratory for Scientific Computing, Petrópolis–RJ, Brazil
E-mail: amandasd@lncc.br

Introduction

- In **Meteorology**, we are used to apply a technique known as **Ensemble weather forecast** which consists of a **combination** of several numerical **weather predictions** derived from **different meteorological models**, and **initial** and **boundary conditions**.
- This technique has been proving to be a viable approach to **reduce** the **uncertainties** in **numerical weather predictions**.
- There are some **statistical methods** for **postprocessing ensembles**. It means combining several forecasts to produce a single ensemble forecast. These methods have **worked well** for variables such as **temperature**. However, these approaches have **not worked well** for **quantitative precipitation prediction**.

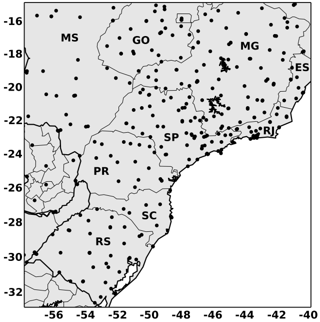
Motivation

The motivation is threefold:

- The **limitations** of the **current methods** for **postprocessing ensembles**, particularly for quantitative precipitation prediction.
- The **difficulty** in **forecasting rainfall amount**.
- The **importance** of an **accurate** and **reliable quantitative precipitation forecast** for the strategic **planning** of several **socio-economic sectors** (such as agricultural production, hydropower generation, water availability for public consumption, flood and landslides controlling, and others).

Context

In this context, we explored an **evolutionary computation algorithm** known as **genetic programming** (**GP**) in order to provide a more accurate and reliable **short-range ensemble forecasts of 24-hour accumulated precipitation** for many real-world data sets over **south**, **southeast** and **central parts** of **Brazil** during the **rainy period** from October to February of 2008 to 2013.



Genetic Programming

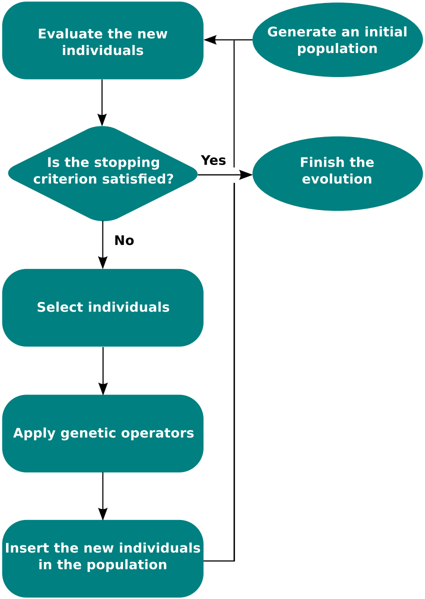
Before showing the results, I will talk a little about **Genetic Programming**.

Genetic Programming

- It is a **stochastic optimization metaheuristic** based on **Darwin's theory** of **evolution** by **natural selection**, commonly referred to as the "**survival of the fittest**": given a population of individuals, the environmental pressure causes natural selection, and so the individuals' fitness tends to rise.
- It evolves a **population** of **computer programs**, usually expressed as **syntax trees**.
- A quite **robust** and **simple** technique in terms of **concept** and **implementation**.
- **Potentially non-linear** technique.
- It evolves **human-interpretable solutions**.
- It exhibits **inherent parallelism**.
- There is the possibility of **introducing specialist knowledge** into the **grammar**.

Genetic Programming

It **starts** with the **generation** of an **initial population** of **candidate solutions**. **Each solution** is **evaluated** according to a **fitness function**. Based on this fitness, some **solutions** are **stochastically selected** from the **population**. The algorithm follows with the **application** of the **genetic operators over** the **selected solutions**. Two of the most important genetic operators are **crossover** and **mutation**. The **new solutions** are **introduced** into the **population**. The **evolutionary process** of **evaluation**, **selection**, **genetic operators**, and **replacement** is **iterated** until a **stopping criterion** is **satisfied**. The **final solution** is the **best solution** of the **last iteration**.



Grammar

There are **different variants** of **GP**. Two of them are: **grammar-based GP** and **grammatical evolution**. Both have the **advantage** of **evolving syntactically correct solutions** in an **arbitrary language** described by a **grammar**.

Six different grammars were designed to tackle the ensemble forecast problem. Now, I will show one of them: the **NLA grammar**.

NLA Grammar

A **grammar** comprises **four entities**: a **start symbol**, the **production rules**, the **terminal symbols**, in italic, and the **non-terminal symbols**, enclosed by brackets. The non-terminal symbols can be replaced by non-terminal or terminal symbols. The **terminal symbols** represent the **operators** and **operands** of the language, and cannot be replaced anymore.

$S =$ *if-then-else* <logical> <ensemble> <ensemble>

$P =$ <ensemble> ::= <model> | <const> | <attribute> |
<binary> <ensemble> <ensemble> |
<unary> <ensemble> |
if-then-else <logical> <ensemble> <ensemble>

<binary> ::= $+ \mid - \mid \times \mid \div$ | *mean* | *max* | *min*

<unary> ::= $-$ | *abs* | $\sqrt{\ }$ | $(\cdot)^2$ | $(\cdot)^3$

<logical> ::= $\vee$ <logical> <logical> |
$\wedge$ <logical> <logical> |
$\neg$ <logical> |
<relational> *pattern* <pattern> |
<relational> <index> <const> |
<relational> <attribute> <const> |
rain | *pattern_change*

<relational> ::= $> \mid < \mid =$

<pattern> ::= $P_1 \mid P_2 \mid P_3 \mid P_4$

<model> ::= $M_1 \mid \ldots \mid M_m$

<index> ::= *K* | *TT* | *SWEAT*

<attribute> ::= $O(1day)$ | $O(2days)$ | *O(mean)* | $O(P)$ |
M(mean) | $M(std)$ | $M(max)$ | $M(min)$ |
$O(lag1+)$ | $O(lag2+)$ | $O(lag3+)$ | *BMA* |
$O(lag1-)$ | $O(lag2-)$ | $O(lag3-)$

NLA Grammar

A **grammar** is a device for generating **sentences — ** a finite sequence of terminal symbols satisfying certain grammatical rules.
The **NLA grammar** enables **linear** and **non-linear combination** of **models**, allows the **use** of some **attributes**, and includes **conditional**, **logical** and **relational operators**.

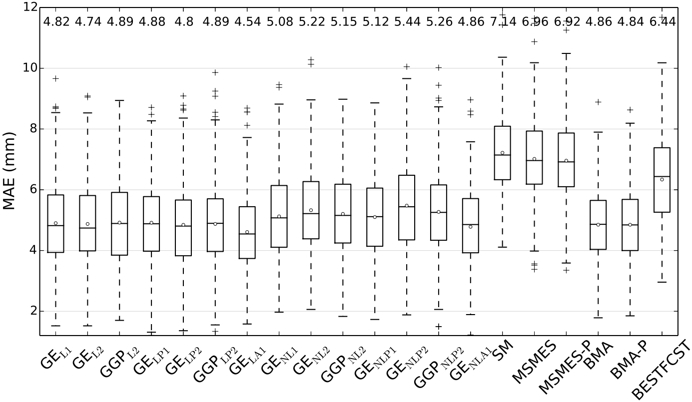
$S =$ *if-then-else* <logical> <ensemble> <ensemble>
$P =$ <ensemble> ::= <model> | <const> | <attribute> |
<binary> <ensemble> <ensemble> |
<unary> <ensemble> |
if-then-else <logical> <ensemble> <ensemble>
<binary> ::= $+ \mid - \mid \times \mid \div \mid$ *mean* | *max* | *min*
<unary> ::= $-$ | *abs* | $\sqrt{\ }$ | $(\cdot)^2$ | $(\cdot)^3$
<logical> ::= $\vee$ <logical> <logical> |
$\wedge$ <logical> <logical> |
$\neg$ <logical> |
<relational> *pattern* <pattern> |
<relational> <index> <const> |
<relational> <attribute> <const> |
rain | *pattern_change*
<relational> ::= $> \mid < \mid =$
<pattern> ::= $P_1 \mid P_2 \mid P_3 \mid P_4$
<model> ::= $M_1 \mid \ldots \mid M_m$
<index> ::= *K* | *TT* | *SWEAT*
<attribute> ::= $O(1day)$ | $O(2days)$ | *O(mean)* | $O(P)$ |
M(mean) | $M(std)$ | $M(max)$ | $M(min)$ |
$O(lag1+)$ | $O(lag2+)$ | $O(lag3+)$ | *BMA* |
$O(lag1-)$ | $O(lag2-)$ | $O(lag3-)$

Results

A **comparison** between some **traditional statistical techniques** and a **set** of **GP experiments** was performed. And now I will show the **results**.

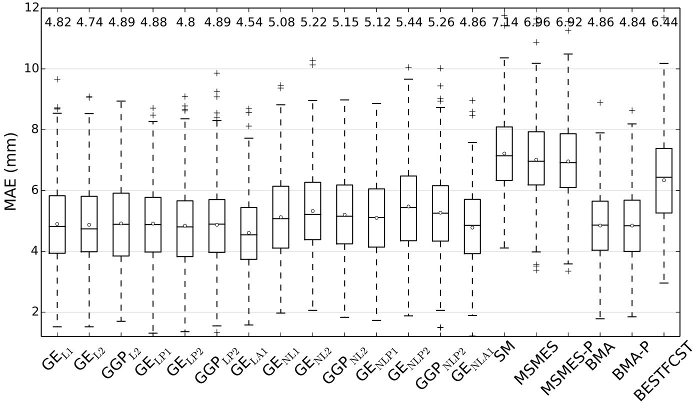
Results

Box plots of the **Mean Absolute Error** (MAE) of the **three-day ensemble forecast** for many **locations** over **Brazil**. The first fourteen boxes are GP experiments with different grammars and different GP versions. The last box is the best ensemble member.

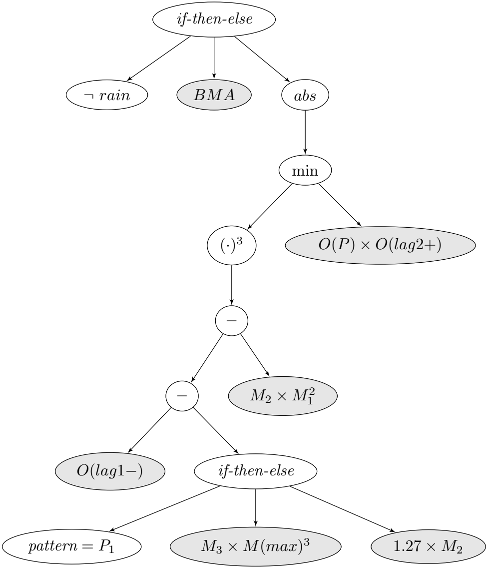


Results

GP obtained a **higher performance** relative to three traditional statistical techniques, with **errors 27–57% lower** than **simple ensemble mean's** and the **MASTER super model ensemble system's**, and is also **superior** to the **best individual forecasts** in **34–42%**. On the other hand, **GP** had a **similar performance** to each other and to the **Bayesian model averaging**, but **GP** is a technique **far more versatile**.



Results



Example of the **program** corresponding to **1-day ensemble quantitative precipitation forecast** for **Franca-SP**, with MAE of 6.98mm on the training set and 5.73mm on the test set. The filled-in gray ellipses are leaf nodes. The expression tree is read from left to right, starting at the top and working down.

In addition to improving the quantitative precipitation forecasts, we can also **extract knowledge** from the best **solutions**.

Conclusions

- The **experiments** showed the **potential** of the **GP** approach, with a clear **advantage** over the **traditional statistical techniques**.
  - GP achieved **more accurate ensemble forecasts**.
  - GP offers **human-interpretable solutions**.
  - Allows the **incorporation** of **specialist knowledge** through a formal grammar.
  - Grammar-based GP can **evolve expressions** of **arbitrary complexity**.
- **Further investigation** on the **improvement** of the **technique** is a **promising line** of **research**.

Underway Work

- The **main drawback** of the **GP** approach is the **high computational cost** of its **fitness function**. It would **preclude** its **operational implementation**, particularly with regard to the **practice** of **weather forecasting**.
- The **applicability** of **GP** to deal with the ensemble forecast problem can be **seriously compromised** with the **increase** of the **volume** of the **input data**.
- For instance, consider a **scenario** of providing **operational long-range ensemble forecasts** of **quantitative precipitation** on the **global scale** using **TIGGE** data; it would probably **take days running**.
- On the other hand, one of the **major advantages** of **GP** is its **high degree** of **parallelism**.
- In this context, we are **currently working** on a **parallel version** of **GP** in order to **reduce computational time** and **improve** the **solution quality**.

Parallel Models of Genetic Programming

I will talk about our **parallel decomposition** of **GP**.

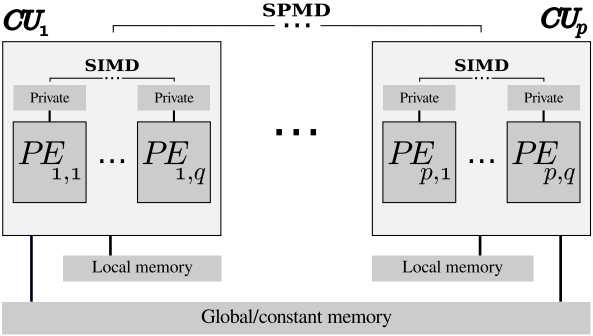
Parallel Models of Genetic Programming

- **GP** can be **decomposed** into **three complementary parallel models**: **algorithmic-level**, **iteration-level** and **solution-level**. The **three parallel models** were **designed** in **a hierarchical multiplicative way**.

- **Regular** and **massive workloads** should be **processed** on **accelerators** whereas **irregular** and **mostly sequential workloads** should be processed on **CPU**. Basically, this means that **CPU manages** the **evolutionary process** and **performs** in a **serial way** the **selection**, **reproduction** and **replacement** steps, while an **accelerator** is responsible for **evaluating** the **solutions** and **finding** the **best solution** at **each generation**.

Iteration- and Solution-Level Parallel Model

The **solutions** are **distributed** among the **compute units** (CU), and the **processing elements** (PE) within each compute unit **take care**, in **parallel**, of the **whole training dataset**.

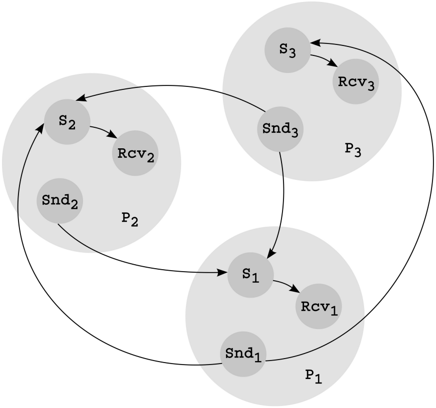


Conceptual compute device architecture.

Algorithmic-Level Parallel Model

- **Different settings** of the **algorithm** are **launched** in a **parallel cooperative way**.
- **Each** independent **run** of the **algorithm** is **assigned** to a **different local** or **remote processor**.
- To fully **exploit** the **available computational resources**, the **number** of **running algorithms** should be roughly **equivalent** to the **number** of **CPUs cores**.
- The **communication** among the **algorithms** follows the **client-server model** based on **sockets**.
  - Each algorithm has its own socket, and the **lightweight processes** of **message passing** are assigned to **multi-threads running** in **parallel**.
  - The **communication topology** can be **modified on-the-fly**.
  - The **message passing** is carried on via **local network** or **Internet**.
  - A **failure** on a **processor** or on the **communication channel cannot bring down** the **whole system**.
  - The **flow** of **message** is **moderate** and the **data transfer speed** is **flexible**.

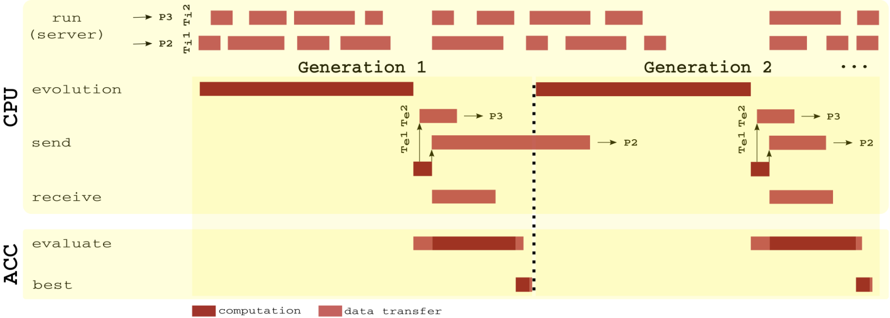
Algorithmic-Level Parallel Model



An **illustrative example** of a **communication topology** between **three processes** or **algorithms**: $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_3}$. Each **process P** is represented by a **server S** and by **sending Snd** and **receiving Rcv message operations**.

Parallel Models of Genetic Programming

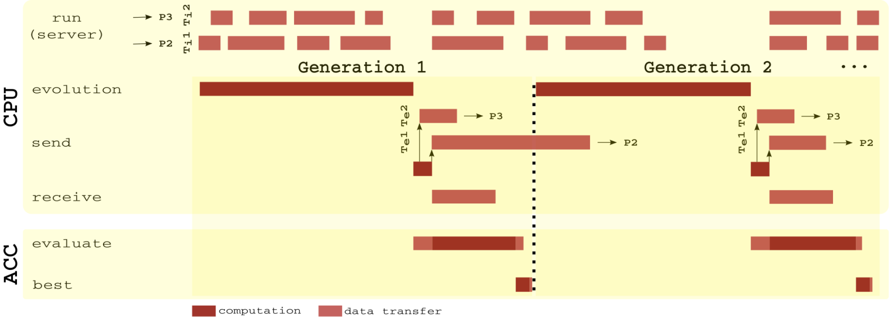
Whenever a **client** is **connected** to a **server**, a **thread automatically starts** in **background** and **receives** the **message**.



Sequence of execution of communication and evolutionary tasks by P_1.

Parallel Models of Genetic Programming

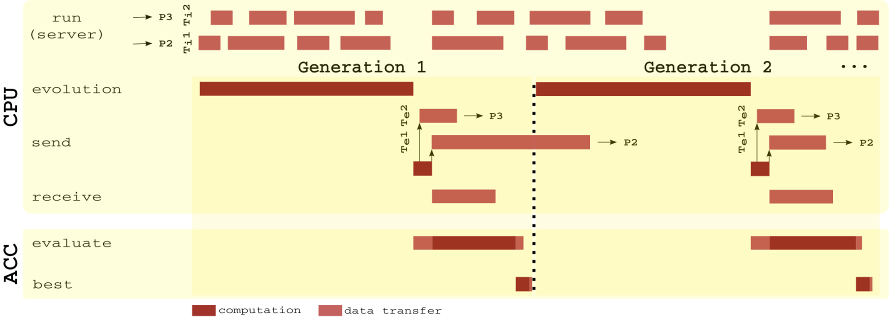
During the **execution** of the **tasks** on **accelerators**, some **communication operations**, such as **sending emigrants** to servers and **transferring immigrants** to current **population**, are done by **CPUs cores** in **background**. That is, the strategy **overlaps computation** and **communication**, which in practice fully **hides** the **communication effort**. Both the **communication** and **evolutionary tasks** are **concurrent**, i.e., the **communication operations** are done **asynchronously** in **background** and do **not interrupt** the **execution** of the **algorithm**.



Sequence of execution of communication and evolutionary tasks by P_1.

Parallel Models of Genetic Programming

The **CPU** and **GPU idle time** and the **impact** of the **communication operations** on the **execution time** of the **whole system** should be **minimal**.



Sequence of execution of communication and evolutionary tasks by P_1.

Final Remarks

Right now we are finishing some implementation details, and we will soon begin the tests of time and solution quality.

!Muchas Gracias!