

# Comparação de algoritmos para Análise intraprocedural na otimização de programas Java

José Clavo Tafur<sup>1</sup>

<sup>1</sup>PPGI – Universidade de Brasília (UnB)  
Brasília – DF – Brazil

**Resumo.** *Este projeto descreve o uso de três técnicas diferentes de métodos quantitativos para comparar, analisar e gerar modelos de predição, tendo como alvo dos algoritmos diferentes que realizam uma análise intraprocedural otimizando programas na linguagem Java. Os resultados mostram que a eficiência de executar estas técnicas é um passo a passo completo de como executá-las.*

## 1. O problema

A otimização de programas é o processo de modificar um programa para fazê-lo executar mais eficientemente ou usar menos recursos [Nielson and Hankin, 2004], para realizar este processo no nível de código fonte se usam análises intra e inter procedurais. Nesta pesquisa, nos enfocaremos na comparação de algoritmos que levam a cabo uma análise intraprocedural [Lam and Hendren, 2011], de programas java, a qual é executada no âmbito de um procedimento (método no Java).

Esta comparação fará uso de 3 técnicas sobre métodos quantitativos: comparações pareadas, projeto 2<sup>3</sup> e regressão linear [Jain, 1991].

## 2. Relevância do problema

Apesar de que os programas estão sendo criados por seniores desenvolvedores e seus processos são gerenciados usando modernas metodologias, os resultados não são os esperados em relação a sua eficiência. O que se procura é que um programa de computador seja otimizado para rodar mais rapidamente ou para torná-lo capaz de operar com menos armazenamento de memória ou outros recursos, ou consumir menos energia.

O resultado deste estudo apresentará quão confiáveis são as técnicas usadas na análise intraprocedural através da execução de casos de teste sendo validados usando métodos quantitativos.

## 3. Trabalho relacionado

A principal inspiração para a criação do Jimple Framework foi o Soot [Group, 2020]. O qual é um framework para manipular e otimizar aplicativos Android e Java através de linguagem intermediárias. Além disso, este framework desempenha diferentes tipos de análises como: (Call-graph construction, Points-to analysis, Def/use chains, Intra e inter procedural data-flow analysis, Taint analysis, entre outros).

## 4. Metodologia de trabalho

Inicialmente, a metodologia vai envolver uma revisão do código dos dois algoritmos Jimple Framework [UnB, 2020] e o White Language [UnB, 2021], os quais estão em Github.

Em seguida, conduzimos a coleta dos dados de entrada que serão usados nos testes. Posteriormente, se executaram os casos de testes coletando o tempo de execução deles. Logo depois, se compararam os valores usando os métodos quantitativos. Finalmente, os resultados vão ser apresentados num relatório.

## 5. Solução

### 5.1. Algoritmos

Os dois algoritmos a comparar executam uma análise intraprocedural. O primeiro faz parte do “Jimple Framework” [UnB, 2020] e foi desenvolvido na linguagem Rascal (Figura 1). Por outro lado, o segundo algoritmo faz parte do “White Language” [UnB, 2021] e foi desenvolvido na linguagem Scala (Figura 2).

Para uma melhor compreensão o primeiro algoritmo será chamado de “Algoritmo A” e o segundo de “Algoritmo B”.

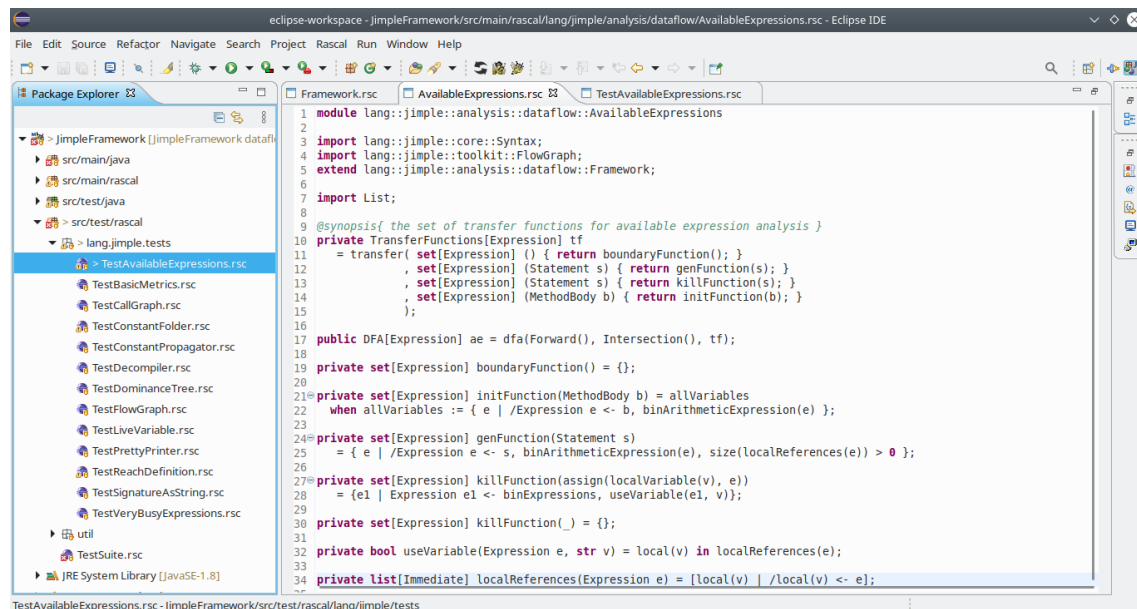


Figure 1. Jimple Framework

Fuente: Elaboración propia.

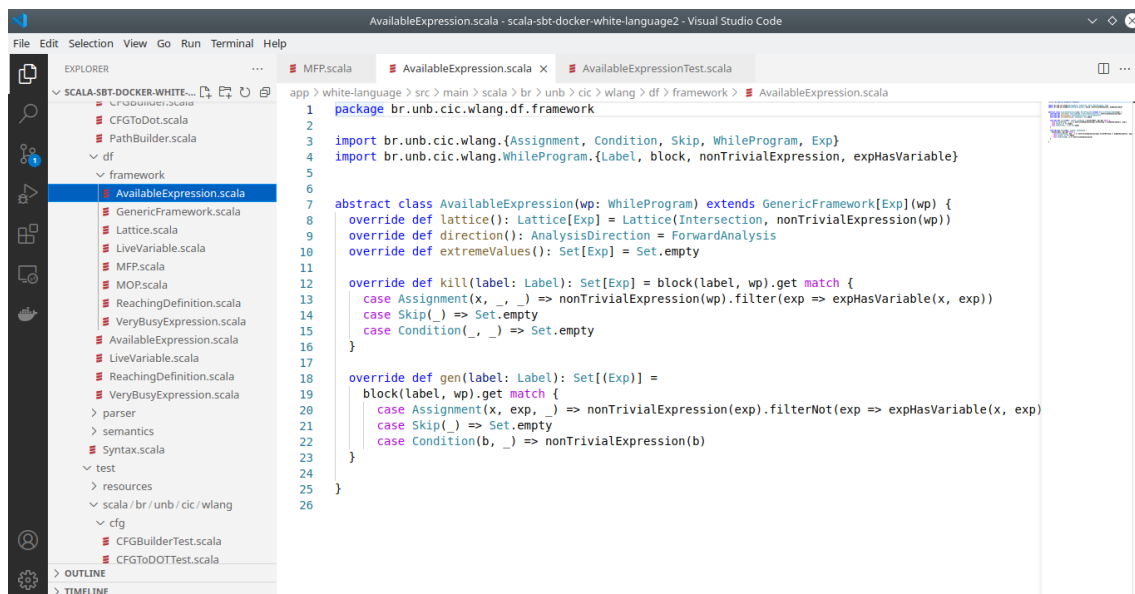
### 5.2. Técnicas de métodos quantitativos

As técnicas de métodos quantitativos ao usar são: comparações pareadas, projeto  $2^3$  e regressão linear.

#### 5.2.1. Comparações Pareadas

Nesta seção, comparamos os desempenho dos dois algoritmos (A e B). Ambos foram testados 8 vezes com a quantidade de tempo mostrado na (Figura 3) y en forma gráfica en la (Figura 4). os dois algoritmos usaram o mesmo padrão para obter os resultados em segundos.

Se compararam com dois valores de intervalos.



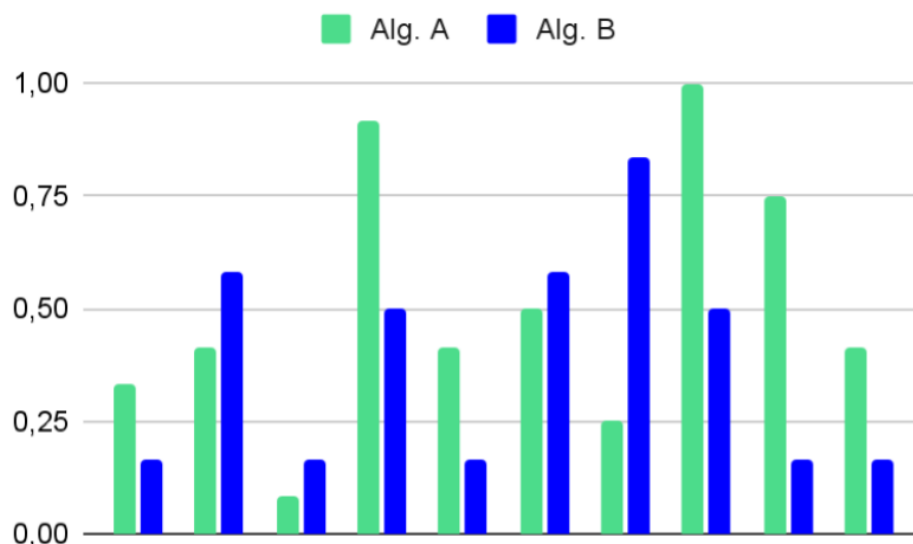
**Figure 2. White Language**

Fuente: Elaboración propia.

Alg. A	0,3333	0,4167	0,0833	0,9167	0,4167	0,5	0,25	1	0,75	0,4167
Alg. B	0,1667	0,5833	0,1667	0,5	0,1667	0,5833	0,8333	0,5	0,1667	0,1667

**Figure 3. Comparações Pareadas : valores**

Fuente: Elaboración propia.



**Figure 4. Comparações Pareadas : gráfico**


Fuente: Elaboración propia.

- **intervalo 90 %** (-0,081 ; 0,331)

Neste intervalo, não se pode rejeitar a hipótese já que tem o 0 entre os valores. Portanto, os algoritmos têm desempenho similar.

- **intervalo 80%** (1,68 ; 1,99)

Neste intervalo, pode-se argumentar que A tem desempenho melhor que B.

Um completo passo a passo da execução da técnica pode ser baixado aqui 

### 5.2.2. Projeto 2<sup>3</sup>

Nesta seção, analisaremos os 3 fatores que maior impacto em nosso projeto os quais são: tipo do algoritmo, tipo do sistema operativo e quantidade de processadores (Figura 5).

Fator	Nível -1	Nível 1
Xa (Algoritmo)	A	B
Xb (S.O)	Windows	Linux
Xc (cpu)	1	4

**Figure 5. Projeto 2<sup>3</sup>: fatores**

Fuente: Elaboración propia.

O projeto 2<sup>3</sup> e seus valores em segundos são mostrado na (Figura 6).

	Xa(-1)		Xa(1)	
	Xc(-1)	Xc(1)	Xc(-1)	Xc(1)
Xb(-1)	0,1628	0,5349	0,2558	0,6744
Xb(1)	0,1163	0,5814	0,3953	0,7612

**Figure 6. Projeto 2<sup>3</sup>: valores**

Fuente: Elaboración propia.


A porção de variação explicada por cada fator e suas interações são mostradas na (Figura 7)

<b>SSA =</b>	14,77 %	
<b>SSB =</b>	1,58 %	
<b>SSC =</b>	81,27 %	<b>&gt; variação</b>
<b>SSAB =</b>	1,58 %	
<b>SSAC =</b>	0,09 %	
<b>SSBC =</b>	0,05 %	
<b>SSABC =</b>	0,66 %	

**Figure 7. Projeto 2<sup>3</sup>: interações**

Fuente: Elaboración propia.

O “**fator C**” com **81,27%** tem a maior variação, portanto a quantidade de processadores são os que geram o maior impacto.

Um completo passo a passo da execução da técnica pode ser baixado aqui 

### 5.2.3. Regressão Linear

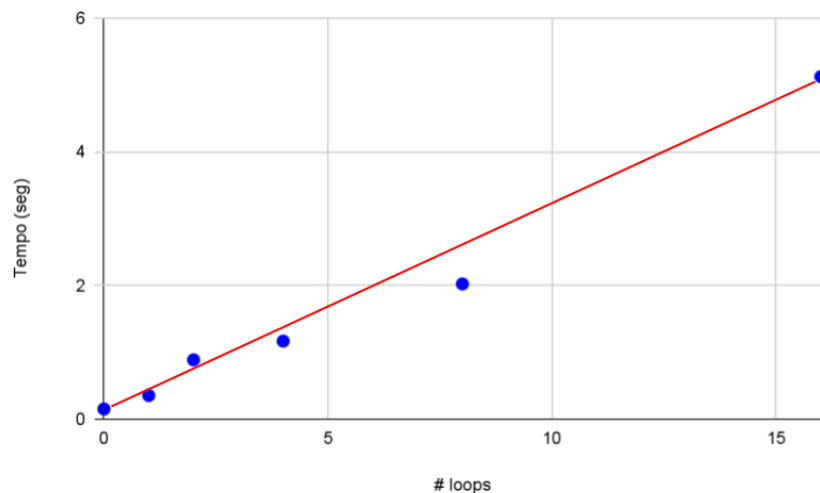
Nesta seção, examinaremos o tempo de execução em segundos do algoritmo A com quantidades diferentes de loops no código (Figura 8), um loop neste representa um “while” ou “for”, para poder realizar uma predição.

x	Loop	0	1	2	4	8	16
y	Tempo	0,151	0,3533	0,8865	1,1672	2,023	5,126

**Figure 8. Regressão Linear: valores**

Fuente: Elaboración propia.

Além disso, se apresenta um gráfico com os parâmetros de estimativa (Figura 9) para uma melhor visualização.



**Figure 9. Regressão Linear: gráfico**

Fuente: Elaboración propia.

A qualidade da regressão medida pelo coeficiente de determinação  $R^2$  é **0,95**, portanto mostra uma alta valor de regressão.

Se efetuou uma predição para um programa com 64 loops, o tempo calculado **T** é **19,27** segundos com **90%** de **IC** é **(18,66;19,88)**.

Um completo passo a passo da execução da técnica pode ser baixado aqui

## 6. Conclusões

As técnicas de métodos quantitativos colocadas em prática mostraram o seguinte: A comparação de observações pareadas com um intervalo de 80% revelou que o algoritmo A é melhor que o B. Além disso, a execução do projeto 2<sup>3</sup> deu como resultado que a quantidade de processadores são os que geram um maior impacto em nosso projeto e finalmente, devido ao resultado das observações pareadas se tomou conta só do algoritmo A com a técnica de regressão linear, a qual gerou um modelo que tem a finalidade de

predizer. Este foi testado para um programa com 64 loops e o tempo calculado foi de 19,27 segundos.

Como conclusão pessoal, as técnicas de métodos quantitativos são umas ferramentas interessantes no momento de comparar, analisar e até fazer uma predição. Aqueles devem ser utilizadas (ou se já estão sendo, devem ser exploradas outras técnicas) em nossos projetos.

## References

- [Group, 2020] Group, S. R. (2020). Soot framework.
- [Jain, 1991] Jain, R. (1991). *Art of Computer Systems Performance Analysis*. Wiley Professional.
- [Lam and Hendren, 2011] Lam, P., B. E. L. O. and Hendren, L. (2011). *The Soot framework for Java program analysis*. CETUS.
- [Nielson and Hankin, 2004] Nielson, F., N. H. R. and Hankin, C. (2004). *Principles of program analysis*. Springer Science and Business Media.
- [UnB, 2020] UnB, P. (2020). Jimple framework.
- [UnB, 2021] UnB, P. (2021). Whilelang.