UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE BACHARELADO EM SISTEMAS DA INFORMAÇÃO

EMERSON SHIGUEO SUGIMOTO

OTIMIZAÇÃO DE MÉTODOS DE PROVA EM TABLÔS KE ATRAVÉS DA APLICAÇÃO DE UMA HEURÍSTICA BASEADA EM ALGORITMOS GENÉTICOS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2013

EMERSON SHIGUEO SUGIMOTO

OTIMIZAÇÃO DE MÉTODOS DE PROVA EM TABLÔS KE ATRAVÉS DA APLICAÇÃO DE UMA HEURÍSTICA BASEADA EM ALGORITMOS GENÉTICOS

Trabalho de Conclusão de Curso de graduação, apresentado ao Curso Superior de Bacharelado em Sistemas da Informação do Departamento Acadêmico de Informática – DAINF – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Bacharel em Sistemas da Informação.

Orientador:

Nome do orientador: Prof. Dr. Adolfo Gustavo Serra Seca Neto

E-mail do orientador: adolfo@dainf.ct.utfpr.edu.br

Assinatura do orientador:

CURITIBA

2013

RESUMO

SUGIMOTO, Emerson Shigueo. Otimização de métodos de prova em tablôs KE através da aplicação de uma heurística baseada em algoritmos genéticos. 2013. 126 f. Trabalho de Conclusão de Curso - Bacharelado em Sistemas de Informação - Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

Este trabalho apresenta uma abordagem de otimização dos métodos de prova em Sistema KE baseada em Algoritmos Genéticos. O foco do presente trabalho é feito sobre a regra PB (Princípio da Bivalência), que bifurca o tablô e onera a sua prova. A seleção das regras PB candidatas é baseada nos conceitos de Seleção Natural expressos na abordagem de Algoritmos Genéticos, de forma que fórmulas que permitam o aumento da variação de premissas na base de conhecimentos são consideradas mais aptas que outras. A seleção das fórmulas mais aptas é comparada com a metodologia convencional de prova em sistemas KE em um ambiente computacional através de um provador automatizado de teoremas. O aumento da variabilidade da base de conhecimentos propiciada pela pré-seleção das regras PB candidatas demonstra uma otimização dos métodos de prova em comparação com a metodologia convencional. Isto se deve ao fato de que quanto maior o número de premissas diferentes, maior a probabilidade de que uma inconsistência seja gerada na base de conhecimentos e, consequentemente, o tablô seja fechado.

Palavras-Chave: Provador de teoremas automatizado. Sistemas KE. Otimização. Sistemas. Algoritmo Genético.

ABSTRACT

SUGIMOTO, Emerson Shigueo. Proof method optimization in the KE tableau system by applying a heuristic based on genetic algorithms. 2013. 126 p. Trabalho de Conclusão de Curso - Bacharelado em Sistemas de Informação - Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

This work presents an approach for optimizing the proof methods in the KE system based on Genetic Algorithms. The focus of this work is on the PB (Principle of bivalence) rule, which bifurcates the tableau and increases the proof cost. The selection of PB candidates is based on concepts expressed in Natural Selection Genetic Algorithm approach, so that formulas that enable increased variation in the knowledge base are considered more suitable than others. The selection of the most suitable formula is compared with the conventional proof method for KE system in a computing environment via an automated theorem prover. The increased variability of the knowledge base provided by the pre-selection of PB candidates demonstrates an optimization of proof methods in comparison with the conventional methodology. This is due to the fact that the greater the number of different premises it is more likely that an inconsistency is generated in the knowledge base, consequently, the tableau will be closed.

Keywords: Automated theorem prover. KE System. Optimization. Systems. Genetic Algorithms.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AG: Algoritmos Genéticos.

EF: *Environmental Factor*.

HoQ: *House of Quality*.

JVM: *Java Virtual Machine*.

PB: Princípio da Bivalência.

SAT: Satisfazibilidade.

Sse: Se e Somente Se.

Subf: Subfórmula.

TCF: *Technical Complexity Factor*.

UAW: *Unadjusted Actor Weight*.

UCP: *Use Case Points*.

UUCP: *Unadjusted Use Case Point*.

UUCW: *Unadjusted Use Case Weight*.

WBS: *Work Breakdown Structure*.

LISTA DE SÍMBOLOS

f: Função.

fo: Função Objeto.

¬: Negação.

◦: Consistência.

•: Inconsistência.

∧: E.
∨: Ou.
T: Verdade.

F: Falso.

→: Implicação.

↔: Bi-Implica.
$\top$: TOP, Veracidade.

$\bot$: BOTTOM, Falsidade.

Ƥ: Conjunto de símbolos proposicionais.

V: Valoração proposicional.

$\mathcal{L}_{\mathrm{LP}}$: Linguagem da lógica proposicional.

∪: União.

: Subconjunto.

\: Diferença entre conjuntos.

∈: Pertence.

Γ: Conjunto de premissas.

c: Função de complexidade

F_p: Conjunto de Fórmulas da lógica proposicional.
φ: Fórmula.
$\mathbb{N}_0$: Conjunto dos números Naturais incluindo o zero.

SUMÁRIO

1. INTRODUÇÃO

Este trabalho apresenta um método de otimização de prova em sistema KE baseado em Algoritmos Genéticos.

A estratégia de Algoritmos Genéticos (AG) mapeia um estado de um problema em forma de indivíduo. Estes são definidos em forma de "cromossomos" ou "código genético". Os Algoritmos Genéticos criam uma população de indivíduos e selecionam uma porção aleatória, através de uma função de avaliação (função de *fitness*). Esta função procura favorecer indivíduos que apresentem maior valor em uma função de desempenho, ou seja, privilegia os mais aptos, e "reproduz" estes indivíduos através de um processo de *crossover*. Este processo consiste na troca de pares de genes entres os indivíduos.

A seleção dos indivíduos está baseada na Teoria da Seleção Natural, formulada pelo naturalista Charles Darwin em 1858 (LINDEN, 2006). De acordo com esta teoria, o indivíduo mais apto ao ambiente possui uma maior probabilidade de se reproduzir e garantir a permanência dos seus genes. O algoritmo prevê, com base em uma probabilidade aleatória, que ocorram mutações nos genes cromossômicos dos indivíduos. Desta forma os indivíduos gerados, incluindo o conjunto dos indivíduos da geração anterior, podem sofrer mutações, gerando outros indivíduos (RUSSELL; NORVIG, 2003).

Os métodos baseados em tablôs são importantes por existirem para diferentes lógicas (FITTING, 1999). Métodos baseados em tablôs podem ser utilizados para desenvolver procedimentos de prova para lógica clássica assim como para vários tipos de lógicas não clássicas (CARNIELLI; CONIGLIO; MARCOS, 2005). Apesar de parecido com o método dos tablôs analíticos, o sistema KE é um sistema refutacional que não é afetado pelas anomalias dos sistemas livres de corte (MONDADORI e D'AGOSTINO, 1994).

Os métodos de prova definem a aplicação de um conjunto definido de regras em uma Base de Conhecimentos, com a finalidade de permitir a dedução formal de conhecimento. Em geral, estes métodos definem regras, em quais proposições elas se aplicam e o conjunto de fórmulas resultantes provenientes da aplicação das regras.

Porém não definem em que ordem as regras devem ser aplicadas, no intuito de maximizar a eficácia da resolução dos tablôs. Dependendo da ordem da aplicação das regras em uma base de conhecimentos, maior ou menor será o tamanho da prova necessária à prova de um tablô. Smullyan (1995) separa as regras dos tablôs analíticos em regras A e B. As regras A possuem consequências diretas, e regras B bifurcam o tablô. Smullyan cita que a aplicação de regras B onera o tamanho da prova de um tablô. Isto se deve ao fato de que estas regras bifurcam o tablô.

Uma demonstração de otimização de provas de tablôs baseada na seleção de fórmulas no momento da aplicação da regra PB (seleção de fórmulas PB candidatas) da lógica paraconsistente, família C1, é descrita em Sugimoto (2010). A aplicação da regra PB em fórmulas que possuam o conectivo de consistência ◦ adiciona novas fórmulas a base de conhecimento (SUGIMOTO, 2010). Isto permite a resolução do tablô de forma mais eficiente do que apenas a resolução baseada na aplicação da regra PB em sequentes de forma aleatória ou mesmo que siga uma ordem fixa e pré-estabelecida de seleção de fórmulas. Com base nesta premissa, o presente trabalho estuda a aplicação de uma técnica de seleção de regras PB candidatas, do sistema KE, baseada no conceito de Seleção Natural expresso na abordagem de Algoritmos Genéticos. A seleção das premissas procura otimizar a velocidade dos métodos de prova em Sistemas KE.

1.1 Objetivo Geral

Este trabalho tem por objetivo otimizar métodos de prova em tablôs KE, especificamente através da seleção de fórmulas PB candidatas através de um método de escolha baseada nos conceitos de Seleção Natural apresentados pelos Algoritmos Genéticos. Por fórmulas PB candidatas subentende-se, para cada estado do tablô, o conjunto de fórmulas que podem bifurcar o tablô através da aplicação da regra PB (a regra PB no sistema KE está representada na subseção 2.2.8).

A metodologia de avaliação e seleção das fórmulas é baseada na premissa de que quanto mais indivíduos novos (premissas novas) forem adicionados à base de conhecimentos, maior a probabilidade de que o tablô seja fechado e, consequentemente,

mais eficiente seja a prova do ponto de vista de consumo de tempo e recursos computacionais.

1.2 Objetivos Específicos

São objetivos do presente trabalho:

- Desenvolver e comparar técnicas de avaliação e seleção das fórmulas PB candidatas.
- Desenvolver três perspectivas para seleção de fórmulas: estocástica, elitista e híbrida.
- Avaliar a metodologia desenvolvida em conjunto com os métodos convencionais de prova apresentados pelo *software* KEMS.

2. LEVANTAMENTO BIBLIOGRÁFICO E ESTADO DA ARTE

As características de Algoritmos Genéticos estão descritas na seção 2.1. Esta seção descreve a representação dos indivíduos, etapas, função de *fitness*, método de seleção por roleta e aplicações de algoritmos genéticos.

A seção 2.2 descreve conceitos da lógica clássica proposicional, representação computacional de fórmulas, proposições, tamanho de prova, regras de eliminação de parênteses, complexidade de fórmulas, tablôs analíticos, fórmulas assinaladas e por fim tablôs KE.

Os estudos sobre métodos de otimização de Prova em tablôs analíticos estão descritos na seção 2.3.

Os Tablôs são um sistema de dedução que estabelece estruturas e permite a representação e a dedução formal de conhecimento. Tablôs analíticos tratam de problemas de satisfazibilidade, especificamente, verificar a satisfazibilidade de uma base de conhecimentos. Dependendo da ordem da aplicação das regras, provas mais eficientes podem ou não ser obtidas. Como apresentado na seção 2.3, Smullyan (1995), descreve a aplicação de regras do tipo B em tablôs analíticos. Como estas regras bifurcam o tablô, diminuem a eficiência da prova. A partir deste ponto de vista, a seleção das regras B é um ponto de otimização dos métodos de prova.

A técnica de AG será utilizada neste trabalho como método de avaliação e seleção de fórmulas, no intuito de otimizar métodos de prova em sistemas KE.

A metodologia de AG aplicada no presente trabalho está descrita na seção 3. Esta seção descreve a aplicação da avaliação, seleção e cruzamento da abordagem AG aplicada à resolução de problemas lógicos.

Os procedimentos de teste e validação estão descritos na seção 4.

Os resultados e as conclusões estão nas seções 5 e 6, respectivamente. As soluções propostas foram mais eficientes do que a solução convencional apresentada pelo provador automatizado de teoremas KEMS.

2.1 Algoritmos Genéticos

Os algoritmos genéticos possuem origem na década de 40 (LINDEN, 2006), quando os cientistas começaram a tentar se inspirar na natureza para criar o ramo da Inteligência Artificial. No final da década de 50 começou-se a buscar modelos de sistemas genéricos para gerar solução para problemas computacionais. No final da década de 60 surge a descrição formal de Algoritmos Genéticos definida por John Henry Holland em seu livro *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, de 1975. Holland apresenta os algoritmos genéticos como uma metáfora para os processos evolutivos (LINDEN, 2006).

A representação dos indivíduos em AG é feita em codificação binária. Como exemplo segue o cromossomo: 000111010. Cada posição (*locus*) do cromossomo assume um dos possíveis valores binários: 0 ou 1.

As etapas de um algoritmo genético estão representadas na Figura 1.

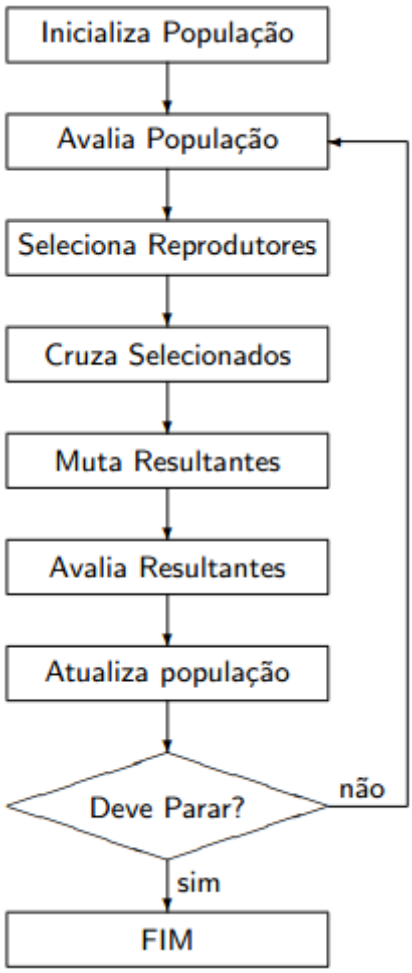


Figura 1 - Estrutura de funcionamento de um AG canônico
Fonte: Lucas (2002).

Descrição das etapas representadas na Figura 1:

- **Inicializa a População**: Nesta etapa os indivíduos são representados em codificação binária, chamados de cromossomos. Todos os indivíduos devem possuir o mesmo tamanho.
- **Avaliação**: avalia-se a aptidão dos indivíduos. A avaliação deve calcular um valor de *fitness* de cada indivíduo. Este valor traduz a aptidão de cada indivíduo de acordo com o problema em estudo.
- **Seleção**: Os indivíduos são selecionados para a reprodução com base em uma probabilidade relativa à sua aptidão (abordagem estocástica). Uma abordagem elitista seleciona os indivíduos mais aptos, ou seja, com um valor de *fitness* maior.

- **Cruzamento**: Ou *crossover*. Características das soluções escolhidas são recombinadas, gerando novos indivíduos. A Figura 2 demonstra um exemplo de *crossover* de 1 ponto dos indivíduos 00101001 e 10011011.

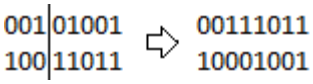


Figura 2 – *Crossover* de 1 ponto
Fonte: Linden (2006).

- **Mutação**: características dos indivíduos são alteradas. Exemplo: para o indivíduo 00101001, uma mutação poderia ser: 0**1**101001, ou 0010100**0**.
- **Atualização**: indivíduos criados na geração são adicionados à população. Uma abordagem convencional seria o descarte de indivíduos menos aptos da geração anterior e adição dos indivíduos criados.
- **Finalização**: verifica-se se o estado objetivo foi alcançado, em caso positivo encerra-se a execução, caso contrário retorna à etapa de avaliação.

2.1.1 Função de *Fitness*

Na etapa de Avaliação (Figura 1) avalia-se a aptidão dos indivíduos. A avaliação deve calcular um valor de *fitness* de cada indivíduo. Este valor traduz a aptidão de cada indivíduo de acordo com o problema em estudo.

A função de Avaliação também é chamada de Função Objeto (fo). Lucas (2002) cita como exemplo a busca do valor máximo de função representada na Figura 3.

$$f(x) = x^3 - x^2 - 10x$$

Figura 3 – Função f(x)
Fonte: Lucas (2002).

$$f_O(G) = f(d(G))$$

Figura 4 – Função Objeto
Fonte: Lucas (2002).

O valor de retorno da Função Objeto (f_O) está representado na Figura 4, onde $f_O(G)$ é a função objeto aplicada sobre o Genoma *G* e *d(G)* a função que decodifica os genes do genoma em um valor x (LUCAS, 2002). Se *G* = [0001010100101001] e *d(x)* é uma função que decodifica números binários em reais, então: *d(G)* = 17,2337, *f(d(G))* = 361,131. Logo, $f_O(G)$ = 361,131.

2.1.2 Seleção por Roleta

A seleção por Roleta é utilizada na abordagem estocástica. Neste método, a seleção das fórmulas deve permitir que premissas menos aptas (fórmulas com uma valoração da função de *fitness* menor) possuam probabilidade de serem escolhidas. Esta probabilidade é relativa ao valor de *fitness* da fórmula. Indivíduos com um *fitness* menor possuem uma probabilidade menor de serem escolhidos, porém não são imediatamente descartados, como na abordagem elitista.

Como exemplo, a Tabela 1 ilustra uma situação descrita em Linden (2006), na qual estão representados os indivíduos, respectivos valores de *fitness*, porcentagem da roleta e o ângulo relativo a cada indivíduo. A Tabela 1 indica que a probabilidade de seleção de um indivíduo menos apto está relacionada diretamente com a sua avaliação. Quanto maior a avaliação do indivíduo, maior o pedaço da roleta e maior será a probabilidade deste indivíduo ser escolhido.

Tabela 1 – Avaliação de Indivíduos.

Indivíduo	Fitness	Fração da Roleta (%)	Fração da Roleta (°)
0001	1	1.61	5.8
0011	9	14.51	52.2
0100	16	25.81	92.9
0110	36	58.07	209.1
Total	62	100.00	360.0

Fonte: Linden (2006).

O pseudocódigo da seleção através da Roleta é apresentado na Figura 5. A função do pseudocódigo é retornar o índice do indivíduo da população sorteado no método de seleção por roleta. Como exemplo, caso o valor sorteado para a variável limite seja de 15.9, o indivíduo selecionado da Tabela 1 seria o indivíduo de índice 2 ou o indivíduo 0011.

```
soma ← valor da soma de todas as avaliações
limite ← valor aleatório entre 0 e soma
i ← 1
aux ← valor de avaliação do indivíduo i
enquanto aux < limite
    i ← i + 1
    aux ← aux + valor de avaliação do indivíduo i
fim enquanto
i ← i – 1
retorne i
```

Figura 5 – Pseudocódigo Roleta.
Fonte: Autoria própria.

2.2 Lógica Clássica Proposicional

De acordo com Silva et al. (2006), a linguagem proposicional envolve proposições (enunciados no qual se pode atribuir um valor verdade - verdadeiro ou falso) e conectivos (ex.: e: '$\wedge$', ou: '$\vee$', negação: '$\neg$', implicação: '$\rightarrow$'), formando fórmulas complexas. O alfabeto da lógica proposicional é composto pelos elementos (SILVA et al., 2006) seguintes:

- Um conjunto infinito e contável de símbolos proposicionais, chamados de átomos, ou de variáveis proposicionais: $\mathcal{P} = \{ p_0, p_1, ... \}$.
- O conectivo unário $\neg$ (negação, lê-se: NÃO).
- Os conectivos binários $\wedge$ (conjunção, lê-se: E), $\vee$ (disjunção, lê-se: OU), e $\rightarrow$ (implicação, lê-se: SE ... ENTÃO ...).
- Os elementos de pontuação, que contêm apenas os parênteses: '(' e ')'.

Os elementos da linguagem $\mathcal{L}_{LP}$ da lógica proposicional são chamados de fórmulas (ou fórmulas bem formadas) e estas são definidas indutivamente como o menor conjunto que satisfaça as seguintes regas de formação (SILVA et al., 2006):

- **Caso básico**: Todos os símbolos proposicionais estão em $\mathcal{L}_{LP}$; ou seja; $\mathcal{P} \subseteq \mathcal{L}_{LP}$. Os símbolos proposicionais são chamados de fórmulas atômicas, ou átomos.
- **Caso indutivo 1**: Se $A \in \mathcal{L}_{LP}$, então $\neg A \in \mathcal{L}_{LP}$.
- **Caso indutivo 2**: Se $A, B \in \mathcal{L}_{LP}$, então $(A \wedge B) \in \mathcal{L}_{LP}$, $(A \vee B) \in \mathcal{L}_{LP}$, $(A \rightarrow B) \in \mathcal{L}_{LP}$.

O conjunto de subfórmulas é definido de acordo com Silva et al. (2006) como sendo:

1. **Caso básico**: A = p. Subf(p) = {p}, fórmula atômica $p \in \mathcal{P}$.
2. **Caso** A = $\neg$ B. Subf($\neg$B) = {$\neg$B} $\cup$ Subf(B).
3. **Caso** A = B $\wedge$ C. Subf(B $\wedge$ C) = { B $\wedge$ C } $\cup$ Subf(B) $\cup$ Subf(C).
4. **Caso** A = B $\vee$ C. Subf(B $\vee$ C) = { B $\vee$ C } $\cup$ Subf(B) $\cup$ Subf(C).
5. **Caso** A = B $\rightarrow$ C. Subf(B $\rightarrow$ C) = { B $\rightarrow$ C } $\cup$ Subf(B) $\cup$ Subf(C).

O tamanho ou complexidade de uma fórmula A, |A| é um número inteiro positivo (SILVA et al., 2006):

1. $|p| = 1$, fórmula atômica $p \in \mathcal{P}$.
2. $|\neg A| = 1 + |A|$.
3. $|A \circ B| = 1 + |A| + |B|$, para $\circ \in \{ \wedge , \vee , \rightarrow \}$.

Na semântica da lógica clássica existem dois valores-verdade: verdadeiro, representado por 1 e o valor falso representado por 0.

Uma valoração proposicional V é uma função $V : \mathcal{P} \rightarrow \{0,1\}$, que mapeia cada símbolo proposicional em $\mathcal{P}$ em um valor verdade. A valoração $V : \mathcal{L}_{LP} \rightarrow \{0,1\}$ é uma extensão do conceito de valoração anterior. De acordo com Silva et al. (2006) a valoração $V : \mathcal{L}_{LP} \rightarrow \{0,1\}$ é feita da seguinte maneira:

- $V(\neg A) = 1$ sse $V(A) = 0$;
- $V(A \wedge B) = 1$ sse $V(A) = 1$ e $V(B) = 1$;
- $V(A \vee B) = 1$ sse $V(A) = 1$ ou $V(B) = 1$;
- $V(A \rightarrow B) = 1$ sse $V(A) = 0$ ou $V(B) = 1$.

2.2.1 Proposições

De acordo com Azevedo Filho (2010), proposição é toda a declaração cujo contexto de uso comporta um e somente um dos valores lógicos: falso (F) ou verdadeiro (T). Hipóteses são formulados através de proposições e a determinação do seu valor lógico é objeto de pesquisa. O processo de verificar se uma proposição tem valor lógico verdadeiro (T) é também chamado de verificação da validade da proposição.

2.2.2 Tamanho de prova

De acordo com Neto (2007) o tamanho de prova pode ser dado por:

- Complexidade da árvore de prova: soma das complexidades de todas as ocorrências de fórmulas assinaladas nas árvores de prova;
- Altura da árvore de prova (considerando ramos como nós);
- Comprimento do caminho mais longo da raiz até uma folha;
- Quantidade de fórmulas distintas;
- Quantidade de fórmulas assinaladas distintas;

O tamanho de prova usado no presente trabalho é dado pela altura da árvore.

2.2.3 Fórmulas Assinaladas

Uma fórmula assinalada é expressa como T X ou F X, onde X é uma fórmula não assinalada. T X é dita verdadeira se X é verdade e falsa se X é falso. E uma fórmula assinalada F X é dita verdade se X é falso e falso se X é verdade.

Desta forma o valor de T X é o mesmo de X e o valor de F X é o mesmo de ¬ X (SMULLYAN, 1995).

2.2.4 Tablôs KE

O método de Tablô analítico é um descendente direto do cálculo de Gentzen livre de corte e é considerado como um paradigma da noção de dedução analítica na lógica clássica (MONDADORI e D'AGOSTINO, 1994).

O sistema de tablôs adotado no presente estudo utiliza os métodos de prova baseados em KE. De acordo com Mondadori e D'Agostino (1994), o sistema KE foi apresentado como uma melhoria, no aspecto computacional, em relação ao sistema de

tablôs analíticos (SMULLYAN, 1968). Apesar de parecido com o sistema de tablôs analíticos, proposto por Smullyan, o sistema KE é um sistema refutacional que não é afetado pelas anomalias dos sistemas livres de corte.

Mondadori e D'Agostino (1994) desenvolveram o sistema KE como proposta a três anomalias independentes na formalização de tablôs analíticos proposto por Smullyan. São elas:

- Anomalia prova teórica;
- Anomalia semântica; e
- Anomalia computacional;

As anomalias estão descritas nas subsecções 2.2.5, 2.2.6 e 2.2.6.

As Regras do sistema KE estão descritas na subseção 2.2.8.

2.2.5 Anomalia prova teórico

Esta anomalia está relacionada com a propriedade da transitividade das deduções normais. Supondo um conjunto de premissas Γ, sua prova poderia ser uma combinação de duas provas mais simples, de B a partir de Γ e de A a partir de Γ, B.

A noção de derivabilidade formal, baseada no método de tablô não satisfaz o princípio da transitividade (MONDADORI e D'AGOSTINO, 1994).

2.2.6 Anomalia semântica

As noções de verdade e falsidade são regidas por dois princípios básicos: não-contradição (nenhuma proposição pode ser verdadeira e falsa ao mesmo tempo) e bivalência (toda proposição é verdadeira ou falsa, e não existem outras possibilidades).

Para as regras subsequentes de Gentzen, $\Gamma \vdash \Delta$ é válido, se e somente se, para cada modelo M, pelo menos uma fórmula Δ é verdadeira em M sempre que todas as fórmulas em Γ são verdadeiras em M (MONDADORI e D'AGOSTINO, 1994).

A regra de corte do teorema de Hauptsatz, formulado por Gentzen está ilustrada na Figura 6.

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma \vdash \Delta, A}{\Gamma \vdash \Delta}$$

Figura 6 – Regra de corte do Teorema de Hauptsatz.
Fonte: Mondadori e D'Agostino (1994).

De acordo com Mondadori e D'Agostino (1994), se lermos a regra da Figura 6 de cabeça para baixo, seguindo a mesma interpretação semântica que adotamos para as regras operacionais, então o que a regra de corte diz é:

> "Em todos os modelos e para todas as proposições A, ou A é verdadeira ou A é falso".

Que é o princípio da bivalência, que caracteriza a noção de verdade e falsidade. Embora o princípio da não-contradição é claramente incorporado na regra para fechar um ramo, não existe uma regra no método tablô (e em sistemas de corte livres Gentzen), que corresponde ao princípio da bivalência. Ao enumerar todos os casos possíveis, as regras de tablô permitem a possibilidade de algo ser diferente de verdadeiro ou falso.

2.2.7 Anomalia computacional

O suposto aumento de velocidade de provas com corte e de provas sem corte é irrelevante do ponto de vista da dedução automatizada (MONDADORI e D'AGOSTINO, 1994).

2.2.8 Regras do sistema KE

As regras do sistema KE estão representadas na Figura 7.

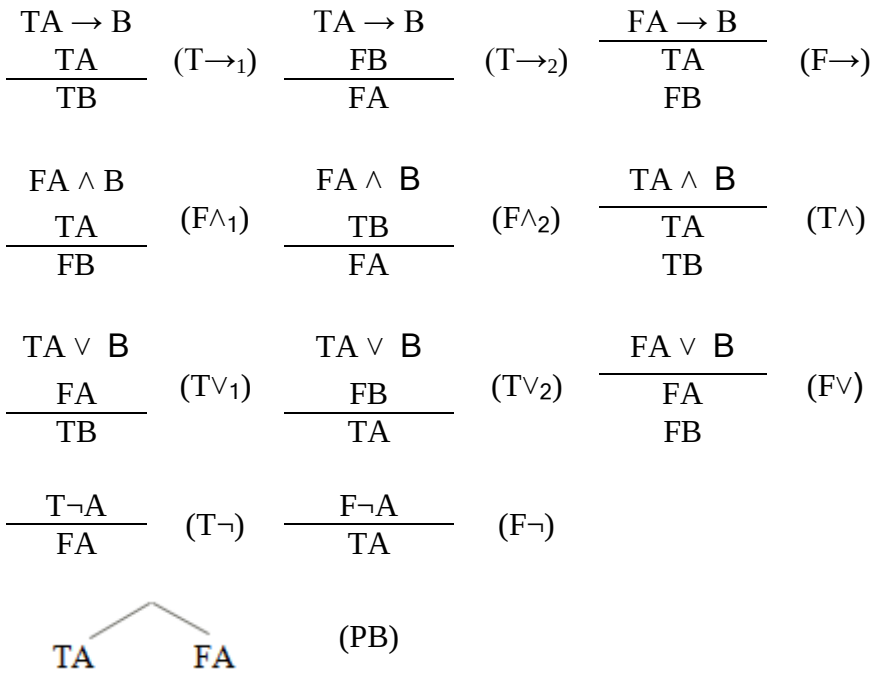


Figura 7 - Regras do sistema de tablôs KE.
Fonte: Neto (2007).

Destaca-se a regra PB (Princípio da Bivalência, representada na Figura 7) que bifurca o tablô e é estudo do presente trabalho.

2.3 Otimização de Prova

Smullyan (1995) separa as regras dos tablôs analíticos em regras A e B. As regras A possuem consequências diretas, e regras B bifurcam o tablô. Smullyan (1995) cita que uma maneira para tratar os tablôs é utilizar as regras de um ramo no sentido de cima para baixo. Nesta abordagem as regras são aplicadas na sequência em que as fórmulas aparecem em um ramo. Porém ele indica que esta abordagem é menos eficiente do que a aplicação prioritária de regras A. Isto se deve ao fato de que a aplicação de regras no sentido de cima para baixo pode aplicar regras B, o que cria novos ramos.

Como exemplo Smullyan (1995) demonstra a prova do sequente [p → (q → r)] → [(p → q) → (p → r)]. As regras A e B aplicadas estão representadas na Figura 8 e Figura 9. A primeira resolução é feita no sentido de cima para baixo (Figura 10) e a segunda privilegia a aplicação da regra A (Figura 11).

Figura 8 - Regra A F →
Fonte: Smullyan (1995).

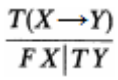


Figura 9 - Regra B T →
Fonte: Smullyan (1995).

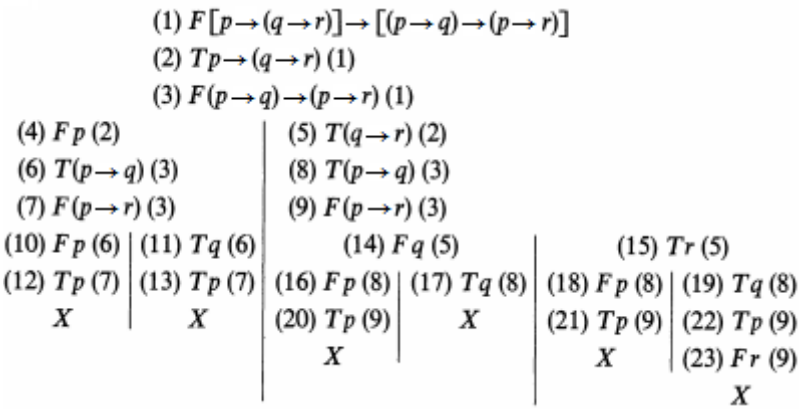


Figura 10 – Demonstração de prova 1
Fonte: Smullyan (1995).

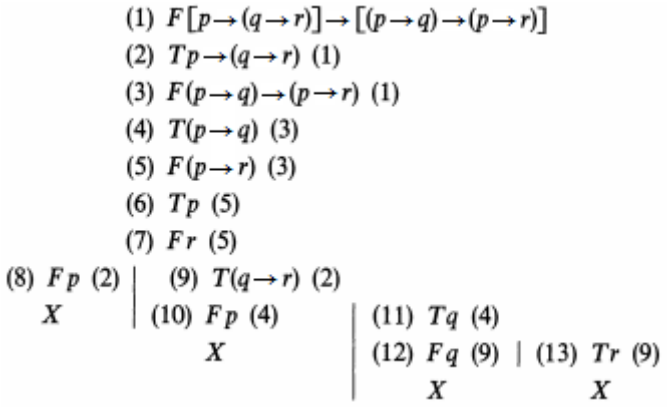


Figura 11 – Demonstração de prova 2
Fonte: Smullyan (1995).

Observa-se que a demonstração de prova que privilegia a aplicação da regra A, representada na Figura 11 (13 linhas), foi mais rápida do que a demonstração 1 da Figura 10 (23 linhas). Isto se deve ao fato de que a aplicação de regras B bifurcam o tablô e desta forma criam novos ramos.

A aplicação de regras que bifurquem o tablô oneram a sua prova devido à bifurcação dos ramos. Desta forma o presente trabalho estuda técnicas de seleção de fórmulas PB candidatas no intuito de otimizar a prova de tablôs no sistema KE.

3. MÉTODO

Como mencionado na subseção 2.3, dependendo da ordem da aplicação das regras, uma prova mais eficiente pode ser obtida (Smullyan, 1995). De acordo com Smullyan (1995) a aplicação de regras B bifurcam o tablô e oneram a sua prova. Em tablôs KE a regra que bifurca o tablô é a regra PB. Esta regra é objeto de estudo do presente trabalho.

Este trabalho apresenta uma abordagem de otimização dos métodos de prova em tablôs KE através da aplicação de uma heurística baseada em algoritmos genéticos (AG). A abordagem baseada em AG é utilizada nas etapas de avaliação e seleção das fórmulas PB candidatas. A aplicação dos AG no presente trabalho está descrita da seção 3.1.

O provador automático de teoremas KEMS será base do desenvolvimento das abordagens de AG elitista, estocástica e híbrida.

3.1 Aplicação de AG

Como mencionado na seção 2.3, dependendo da ordem da aplicação das regras, uma prova mais eficiente pode ser obtida. Com base nesta premissa, uma técnica de seleção fundamentada em AG será utilizada para otimizar métodos de prova baseados em tablôs KE.

As premissas de um tablô são ditas indivíduos para a técnica de AG. A avaliação e seleção são feitas sobre as fórmulas PB candidatas a um determinado estado de prova do tablô. O processo de *crossover* será dado pela aplicação da regra PB na fórmula escolhida na etapa de seleção.

As etapas dos AG, representadas na Figura 1 (seção 2.1), utilizadas no presente trabalho são:

- **Inicializa a População**: A população é formada pelas premissas da base de conhecimento.

- **Avaliação**: A avaliação é dada para cada fórmula PB candidata de um determinado estado do tablô.
- **Seleção**: Três abordagens de seleção são utilizadas neste trabalho: elitista, estocástica e híbrida.
- **Cruzamento (Ou *crossover*)**: Dá-se pela aplicação da regra PB à fórmula selecionada.
- **Mutação**: Não se aplica ao trabalho proposto.
- **Atualização**: As fórmulas resultantes são adicionadas ao ramo do tablô. Não há descarte de gerações.
- **Finalização**: verifica-se se o estado objetivo foi alcançado, no caso, se o ramo está fechado. Em caso positivo, repete-se a etapa de avaliação para um ramo aberto. Encerra-se a execução quando todos os ramos estiverem fechados. Caso o tablô ramo não esteja fechado as etapas de Avaliação, Seleção, Cruzamento e Atualização são repetidas, caso a etapa do Cruzamento não possa ser feita, o tablô é considerado aberto e o processo de prova finalizado.

As etapas de Avaliação, Seleção e Cruzamento de fórmulas estão descritas nas seções 3.1.1, 3.1.4 e 3.1.8, respectivamente.

3.1.1 Avaliação

A avaliação é feita para cada fórmula candidata a aplicação da regra PB. A etapa de avaliação leva em consideração que quanto maior o número de premissas novas adicionadas à base de conhecimento através da aplicação da regra PB, maior será o valor de *fitness* dado à fórmula. A etapa de avaliação serve de base ao processo de seleção das fórmulas (seção 3.1.4) e posterior *crossover* (seção 3.1.8).

As heurísticas adotadas para classificação e escolha de fórmulas PB candidatas envolvem a complexidade das fórmulas, descrita na subseção 3.1.2, e a análise de frequência dos átomos das premissas da base de conhecimento, descrita na subseção 3.1.3.

3.1.2 Avaliação por complexidade de fórmulas

A otimização do procedimento de prova em sistemas KE é baseada no aumento da variabilidade genética da base de conhecimentos. A ideia empregada é a de que o aumento do número de novas fórmulas aumente a probabilidade da ocorrência de inconsistências na base de conhecimentos, o que por sua vez leva ao fechamento do ramo e do tablô.

A aplicação da regra PB em fórmulas com maior complexidade pode gerar novas premissas, aumentando desta forma a variabilidade genética da base de conhecimentos. Desta forma, entre um conjunto de fórmulas PB candidatas, a fórmula de maior *fitness* será a que, dentre as premissas, possuir a maior complexidade. O resultado da função de avaliação é dado pelo cálculo de complexidade da fórmula. Como exemplo seguem as premissas:

Premissa 1: T A ∨ B (complexidade 3).

Premissa 2: T (A ∨ B) → C (complexidade 5)

A aplicação da regra PB na Premissa 1 e Premissa 2, respectivamente, estão representadas na Figura 12 e Figura 13.

T A ∨ B	
F A (PB)	TA (PB)
T B ($T\vee_1$)	

Figura 12 - Aplicação de PB em T A ∨ B
Fonte: Autoria própria.

T (A ∨ B) → C	
F A ∨ B (PB)	T A ∨ B (PB)
F A (F ∨)	T C ($T \rightarrow_1$)
F B (F ∨)	

Figura 13 - Aplicação de PB em T (A ∨ B) → C
Fonte: Autoria própria.

De acordo com a Figura 13, após a aplicação da regra PB em T (A ∨ B) → C, obtêm-se F A ∨ B e T A ∨ B. A partir de F A ∨ B obtêm-se FA e FB (regra F∨). Estas premissas podem ser usadas em conjunto com outras fórmulas e regras e desta forma aumentar a probabilidade de que o ramo e tablô sejam fechados.

Um exemplo de prova está descrito nas figuras: Figura 14 e Figura 15 (subseção 3.1.5).

Cabe ressaltar que esta é uma heurística adotada, não necessariamente o procedimento de prova será otimizado apenas pela escolha de premissas PB candidatas de maior complexidade.

3.1.3 Avaliação por análise frequência átomos

De forma semelhante à avaliação de fórmulas de maior complexidade (subseção 3.1.2), a avaliação por análise de frequência dos átomos tem a premissa de que quanto maior a variabilidade da base de conhecimentos, maior a probabilidade de que uma inconsistência seja gerada.

Especificamente a variabilidade intrínseca à fórmula PB candidata é objeto de estudo do aumento da variabilidade da base de conhecimentos. A aplicação da regra PB em fórmulas com maior valoração da análise de frequência de átomos pode gerar novas premissas, aumentando desta forma a variabilidade genética da base de conhecimentos.

A avaliação de frequências leva em consideração a frequência dos átomos das fórmulas PB candidatas e as demais fórmulas da base de conhecimentos.

Como exemplo, seja A o conjunto de premissas da base de conhecimentos, PB o conjunto de fórmulas PB candidatas (sendo PB $\subset$ A) e B a diferença entre os conjuntos A e PB (A \ PB).

Sendo o conjunto A formado pelas premissas:

A = {TA ∧ C, FA → C ∨ D, T A → C, T C}.

O conjunto PB é subconjunto de A, PB $\subset$ A, e formado pelas premissas:

PB = {TA ∧ C, FA → C ∨ D}.

O conjunto B é dado pela diferença:

B = A \ PB = {T A → C, T C}.

Sendo a relação átomo e sua respectiva frequência dada por {átomo: frequência}, a frequência de átomos das premissas do conjunto A é:

- TA ∧ C, {A: 1, C: 1}
- FA → C ∨ D, {A: 1, C: 1, D: 1}
- T A → C, {A: 1, C: 1}
- T C, {C: 1}

A frequência de átomos das premissas do conjunto PB é:

- TA ∧ C, {A: 1, C: 1}
- FA → C ∨ D, {A: 1, C: 1, D: 1}

A análise de frequência dos átomos da base de conhecimentos é feita sobre a diferença entre os conjuntos A e PB, ou seja, formado pelo conjunto B:

- T A → C, {A: 1, C: 1}
- T C, {C: 1}

Desta forma a lista de frequências dos átomos da base de conhecimentos é dada pela soma de frequências dos átomos de B:

- {A: 1, C: 2}

A soma das frequências dos átomos das fórmulas PB candidatas com a lista de frequências é dada por:

- TA ∧ C, {A: 2, C: 3}, totalizando-se uma avaliação de 5.
- FA → C ∨ D, {A: 2, C: 3, D: 1}, totalizando-se uma avaliação de 6.

Desta forma a fórmula PB candidata com maior valoração de *fitness* será FA → C ∨ D, de avaliação 6. Esta abordagem tem a premissa de que quanto maior a frequência dos átomos nas fórmulas PB candidatas em relação à frequência dos átomos da base de conhecimentos, maior será a probabilidade de que uma inconsistência seja gerada, e desta forma, o tablô seja fechado.

A função de valoração de uma fórmula PB candidata é dada pela soma das frequências dos átomos das fórmulas PB candidatas e das frequências dos átomos das fórmulas da base de conhecimentos.

3.1.4 Seleção

Três abordagens de seleção baseadas na avaliação das fórmulas PB candidatas (seção 3.1.1) são fontes de estudo do presente trabalho. São elas: seleção elitista, estocástica e híbrida, descritas nas seções 3.1.5, 3.1.6 e 3.1.7, respectivamente.

O presente trabalho estuda as abordagens elitista e estocástica em conjunto com as funções de avaliação baseadas na complexidade das fórmulas e da análise de frequência dos átomos, descritos na seção 3.1.1. Para efeitos comparativos, a seleção elitista é feita sob dois aspectos:

1. Seleção da primeira fórmula PB candidata de maior valoração de *fitness*, independente da existência de outras fórmulas com a mesma valoração;
2. Seleção híbrida: elitista caso não exista mais de uma fórmula com o maior valor de *fitness* e estocástica caso contrário.

A comparação dos dois aspectos da seleção elitista busca verificar a avaliação de eficiência de prova, do consumo de tempo e recursos computacionais requisitados por ambos os aspectos.

3.1.5 Seleção elitista

A abordagem elitista sempre seleciona as premissas com o maior valor de *fitness*. Como exemplo de seleção elitista em conjunto com a avaliação de complexidade de fórmulas (3.1.2), seguem as fórmulas:

Fórmula 1: F $(A \rightarrow B) \rightarrow C$, de complexidade 5.

Fórmula 2: T $((A \wedge B) \rightarrow (B \vee C)) \rightarrow (A \wedge C)$, de complexidade 11.

A Figura 14 demonstra a prova privilegiando-se a aplicação da regra PB em fórmulas de menor complexidade. Na prova descrita na Figura 15, privilegia-se a aplicação da regra PB em fórmulas de maior complexidade.

(1) F (A → B) → C
(2) T ((A ∧ B) → (B ∨ C)) → (A ∧ C)
(3) T A → B (F →, 1)
(4) F C (F →, 1)

(5) TA (PB, 3)

(6) TB (T →, 5 e 3)

(7) T (A ∧ B) → (B ∨ C) (PB, 2)
(8) T A ∧ C (T →, 7 e 2)
(9) TA (T ∧, 8)
(10) TC (T ∧, 8)
X (10 e 4)

(11) F (A ∧ B) → (B ∨ C) (PB, 2)
(12) T A ∧ B (F →, 11)
(13) F B ∨ C (F →, 11)
(14) FB (F ∨, 13)
(15) FC (F ∨, 13)
X (14 e 6)

(16) FA (PB, 3)

(17) T (A ∧ B) → (B ∨ C) (PB, 2)
(18) T A ∧ C (T →, 17 e 2)
(19) TA (T ∧, 18)
(20) TC (T ∧, 18)
X (20 e 4)

(21) F (A ∧ B) → (B ∨ C) (PB, 2)
(22) T A ∧ B (F →, 21)
(23) F B ∨ C (F →, 21)
(24) FB (F ∨, 23)
(25) FC (F ∨, 23)
(26) TA (T ∧, 22)
(27) TB (T ∧, 22)
X (27 e 24)

Figura 14 - Prova 1
Fonte: Autoria própria.

(1) F (A → B) → C
(2) T ((A ∧ B) → (B ∨ C)) → (A ∧ C)
(3) T A → B (F →, 1)
(4) F C (F →, 1)

(5) T (A ∧ B) → (B ∨ C) (PB, 2)
(6) T A ∧ C (T →, 5 e 2)
(7) TA (T ∧, 6)
(8) TC (T ∧, 6)
X (8 e 4)

(9) F (A ∧ B) → (B ∨ C) (PB, 2)
(10) T A ∧ B (F →, 9)
(11) F B ∨ C (F →, 9)
(12) TA (T ∧, 10)
(13) TB (T ∧, 10)
(14) FB (F ∨, 11)
(15) FC (F ∨, 11)
X (14 e 13)

Figura 15 - Prova 2
Fonte: Autoria própria.

A primeira prova (Figura 14) tem 3 bifurcações e 27 linhas. A segunda prova (Figura 15) tem 1 bifurcação e 15 linhas. A diferença entre as duas provas está na ordem da aplicação da regra PB. A primeira prova aplica PB em T $A \rightarrow B$, de complexidade 3, enquanto a segunda aplica PB na premissa T $((A \wedge B) \rightarrow (B \vee C)) \rightarrow (A \wedge C)$, de complexidade 11. Desta forma, a segunda prova aumenta a variabilidade de premissas em cada ramo, o que influencia diretamente no número de possibilidades do ramo ser fechado.

3.1.6 Seleção estocástica

A abordagem estocástica seleciona as premissas de acordo com uma probabilidade relativa à sua aptidão, de forma que, quanto maior o valor de *fitness*, maior a probabilidade de que a premissa seja escolhida. No presente trabalho, a forma de seleção das premissas usa a técnica de seleção por roleta, descrita na seção 2.1.2.

No método estocástico, a seleção deve permitir que premissas menos aptas (premissas com uma valoração da função de *fitness* menor) possuam probabilidade de serem escolhidas. Esta probabilidade é relativa ao seu valor de *fitness*. Indivíduos com um valor de *fitness* menor possuem uma probabilidade menor de serem escolhidos, porém não são imediatamente descartados, como na abordagem elitista. A seleção estocástica é feita através da roleta, descrita na seção 2.1.2. O código em Java está ilustrado na Figura 16.

```
public long GetSomaAvaliacoes(){
        if (getPblist()==null || getPblist().size() <= 0) {return 0;}
        long rt = 0;
        for (Iterator<SignedFormula> sgF = getPblist().iterator();sgF.hasNext();) {
                rt += ((SignedFormula)sgF.next()).getComplexity();
        }
        return rt;
}
public int Roleta(){
        if (getPblist()==null || getPblist().size() <= 0) {return -1;}
        int i; double aux = 0;
        double limite = Math.random() * this.GetSomaAvaliacoes();
        for(i = 0; ( (i < getPblist().size()) && (aux < limite) ); ++i){
                aux += getPblist().get(i).getComplexity();
        }
        i--;
        return i;
}
public SignedFormula GetIndividuoRoleta(int pos){
        if (pos < 0) { return null; }
        return getPblist().get(pos);
}
```

Figura 16 - Roleta Java
Fonte: Autoria própria.

O código de seleção por roleta (Figura 16) sorteia um número entre zero e a soma de todos os valores de *fitness*. Posteriormente usa cada avaliação dos indivíduos

como peso e retorna o índice do indivíduo escolhido. A função GetIndividuoRoleta(**int** pos) retorna o indivíduo a partir do índice.

O principal objetivo da seleção estocástica é estudar soluções alternativas à abordagem elitista. Podem existir casos em que a abordagem estocástica permita a seleção de fórmulas com menor valoração de *fitness*, mas que permitam provas mais eficientes.

3.1.7 Seleção híbrida

A abordagem híbrida trata os casos em que, para a seleção elitista, existe mais de uma fórmula com o maior valor de *fitness* dentre as fórmulas PB candidatas. Para estes casos, a seleção empregada no presente trabalho será a estocástica. A abordagem híbrida combina os aspectos da seleção elitista e estocástica para os casos de premissas com a mesma valoração

Cabe ressaltar que a abordagem híbrida por seleção elitista em conjunto com a seleção estocástica é adotada neste trabalho, porém outras abordagens podem ser adotadas, como exemplo a seleção elitista com avaliação de complexidade e posterior seleção elitista por análise de frequência de átomos.

Como exemplo, seguem as premissas:

- T A → (C ∨ D), de complexidade 5.
- T B → C, de complexidade 3.
- F A ∨ D ∧ B, de complexidade 5.

Existem duas premissas com a mesma valoração de complexidade 5, T A → (C ∨ D) e F A ∨ D ∧ B. A abordagem híbrida tem por objetivo tratar estes casos, de forma a propiciar uma seleção mais próxima ao conceito de seleção natural apresentada pelos algoritmos genéticos. O algoritmo híbrido implementa a seleção estocástica entre as premissas T A → (C ∨ D) e F A ∨ D ∧ B. A seleção estocástica pode utilizar como função de avaliação a complexidade das fórmulas (subseção 3.1.2) ou análise de frequência de átomos (subseção 3.1.3).

3.1.8 *Crossover*

O cruzamento (*crossover*) é dado pela aplicação da regra PB na premissa escolhida na etapa de seleção (seção 3.1.4).

Como representado na Figura 13, com a aplicação da regra PB em T (A ∨ B) → C, obtêm-se F A ∨ B e T A ∨ B. A premissa F A ∨ B é adicionada ao ramo da direita e T A ∨ B é adicionado ao ramo da esquerda.

Linden (2006) destaca o problema da convergência genética como sendo uma das principais desvantagens dos algoritmos genéticos. Isto se dá, pois rapidamente ele pode ficar preso em um máximo local e demorar ou não encontrar o máximo global devido à restrição gênica imposta a ele em virtude da abordagem elitista. Como o presente trabalho não aborda a reprodução dos indivíduos através do *crossover* convencional, ou seja, troca de segmentos cromossômicos, a restrição da convergência genética não se aplica na resolução dos problemas de busca propostos.

3.2 Pseudocódigo

Os pseudocódigos das abordagens estão representados na Figura 17 e na Figura 18. O pseudocódigo da roleta é descrito na seção 2.1.2, Figura 5.

```
enquanto Tablô Estiver Aberto
    para cada ramo : lista ramos
      se ramo.Fechado == true
            interrompa
      fim se
      formulas ← lista de fórmulas PB candidatas ( ramo )
      FormulaEscolhida ← null
      para cada formula : formulas
            formula.CalcularFitness()
            se FormulaEscolhida == null
                FormulaEscolhida ← formula
            se não
                se FormulaEscolhida.ValorFitness < formula.ValorFitness
                      FormulaEscolhida ← formula
                fim se
            fim se
      fim para
      se FormulaEscolhida == null
            interrompa
      fim se
      AplicarPB ( FormulaEscolhida )
    fim para
fim enquanto
```

Figura 17 - Pseudocódigo Geral da abordagem elitista.
Fonte: Autoria própria.

```
enquanto Tablô Estiver Aberto
  para cada ramo : lista ramos
    se ramo.Fechado == true
      interrompa
    fim se
    formulas ← lista de fórmulas PB candidatas ( ramo )
    FormulaEscolhida ← null
    para cada formula : formulas
      formula.CalcularFitness()
      se FormulaEscolhida == null
        FormulaEscolhida ← Roleta (formulas)
      fim se
      se FormulaEscolhida == null
        interrompa
      fim se
    fim para
    AplicarPB ( FormulaEscolhida )
  fim para
fim enquanto
```

Figura 18 - Pseudocódigo Geral da abordagem estocástica.
Fonte: Autoria própria.

4. PROCEDIMENTOS DE TESTE E VALIDAÇÃO

Para análise e testes das classes desenvolvidas, a tecnologia JUNIT de testes automatizados em Java foi utilizada. Através dela podem ser criados testes em lote para validar os métodos das classes desenvolvidas.

O componente JUNIT é utilizado em conjunto com o ambiente de desenvolvimento em Java das estratégias AG, Eclipse. Isto facilita a automatização e verificação dos resultados dos testes.

Os objetos de teste, que representam as premissas, da classe SignedFormula (Figura 14), são criados de forma a contemplar todos os tipos de conectivos pertencentes ao sistema KE (seção 2.2.4). O conjunto de testes da abordagem proposta é feito sobre instâncias de fórmulas atômicas e complexas da classe SignedFormula.

Incialmente os teste foram feitos para as técnicas de avaliação (*fitness*) de fórmulas por complexidade (subseção 3.1.2) e análise de frequência de átomos (subseção 3.1.3).

Posteriormente os testes das classes que envolvem as etapas de avaliação e seleção foram feitos. As classes que envolvem estas etapas são: AGEstrategia (Figura 17), AGElitistaFrequenciaAtomos (Figura 21), AGEstocasticoFrequenciaAtomos (Figura 22), AnaliseNumeroAtomos (Figura 23), AGElitistaMaiorComplexidade (Figura 24), AGEstocasticoMaiorComplexidade (Figura 25), AGHibFreqAtomosElitEsto (Figura 26) e AGHibrMaiorComplexidadeElitEsto (Figura 27).

Os testes foram realizados em uma máquina com as seguintes características: capacidade de processamento de 2 GB de memória RAM, processador Intel(R) Core(TM) 2 Quad Q6600 @ 2.40GHz e sistema operacional Windows 7, conforme descrição da seção 1.

5. RESULTADOS

A avaliação é feita através da comparação dos resultados da abordagem baseada em AG e da metodologia convencional de prova. Os critérios analisados foram:

- Número de Bifurcações;
- Tamanho de Prova;
- Tempo de Prova;
- Consumo de Memória;
- Número de Nós;

Com relação aos critérios analisados, quanto menor os resultados para: número de bifurcações, tamanho da prova, tempo utilizado, número de nós e consumo de memória, mais eficiente é considerada a abordagem. As abordagens que reunirem o maior conjunto de melhores resultados serão consideradas mais eficientes.

Os testes foram realizados em uma máquina com as características capacidade de processamento de 2 GB de memória RAM, processador Intel(R) Core(TM) 2 Quad Q6600 @ 2.40GHz e sistema operacional Windows 7, conforme descrição da seção 1.

As famílias de problemas em sistema KE utilizadas para os testes estão descritas na seção 5.1.

As abordagens de AG testadas incluem as técnicas de seleção elitista, estocástica e híbridas em conjunto com as técnicas de avaliação de análise de frequência de átomos e de complexidade de fórmulas. Todas as abordagens propostas de AG são comparadas com a metodologia convencional de prova apresentada pelo *software* KEMS.

A configuração do KEMS para todas as abordagens está ilustrada na Figura 19. As configurações padrões são utilizadas, sendo a estratégia utilizada a ConfigurableSimpleStrategy e para o método de comparação de fórmulas o comparador InsertionOrderComparator. O tempo máximo estipulado para cada prova foi de 20 minutos.

Os testes de comparação das abordagens foram feitos em lote, através de linha de comando com o arquivo executável jar do KEMS. A configuração do *software* KEMS para execução em lote está descrita no APÊNDICE D – KEMS - Execução em lote.

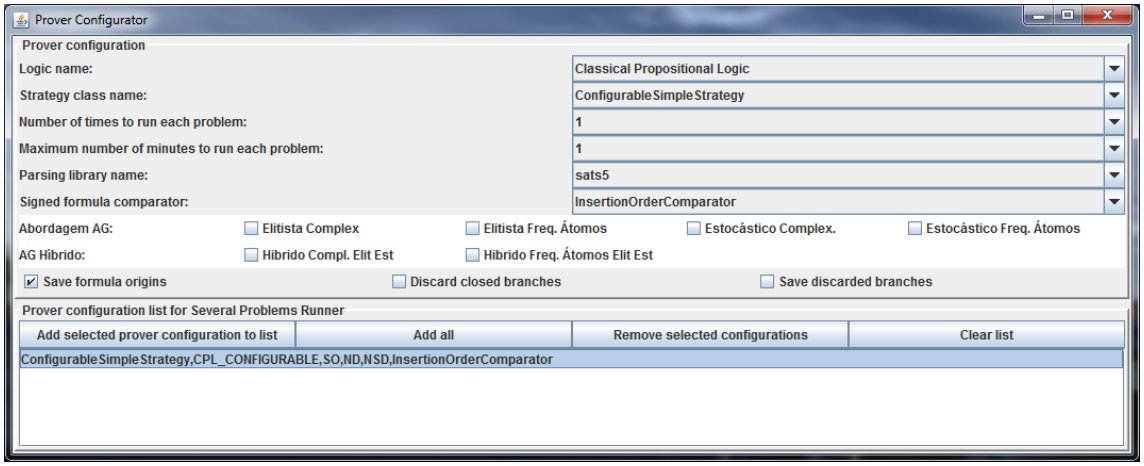


Figura 19 – Configurações de prova do KEMS
Fonte: Autoria própria.

Os resultados de prova estão descritos nas seções: 5.2, 5.3, 5.4 e 5.5.

Cabe ressaltar que não foi encontrada apenas uma solução genérica ótima para todas as famílias. Cada família possui características próprias, de forma que, cada tipo de família de problemas possui uma, ou um conjunto de soluções ótimas.

5.1 Famílias de problemas

As famílias de problemas em sistemas KE utilizadas para os testes são:

- Gamma, instâncias de 1 a 10;
- Statman, instâncias de 1 a 9, 13, 17, 21, 25 e 29;
- H, instâncias de 1 a 7; e
- PHP, instâncias 1 a 6.

Estas famílias foram selecionadas por serem as mais utilizadas para resolução de tablôs (NETO, 2007).

A instância 4 de todas as famílias de problemas utilizadas estão representadas nas figuras: Figura 20, Figura 21, Figura 22 e Figura 23.

```
T(a1|b1)
T(a1->(a2|b2))
T(b1->(a2|b2))
T(a2->(a3|b3))
T(b2->(a3|b3))
T(a3->(a4|b4))
T(b3->(a4|b4))
T(a4->(a5|b5))
T(b4->(a5|b5))
F(a5|b5)
```

Figura 20 – Família Gamma instância 4
Fonte: Autoria própria.

```
T(c0|d0)
T(((c0|d0)->c1)|((c0|d0)->d1))
T((((c0|d0)&(c1|d1))->c2)|(((c0|d0)&(c1|d1))->d2))
T((((c0|d0)&(c1|d1)&(c2|d2))->c3)|(((c0|d0)&(c1|d1)&(c2|d2))->d3))
F(c3|d3)
```

Figura 21 – Família Statman instância 4
Fonte: Autoria própria.

```
F(((!p1)&(!p2)&(!p3)&(!p4))|(p1&(!p2)&(!p3)&(!p4))|((!p1)&p2&(!p3)&(!p4))|(p1&p
2&(!p3)&(!p4))|((!p1)&(!p2)&p3&(!p4))|(p1&(!p2)&p3&(!p4))|((!p1)&p2&p3&(!p4))|
(p1&p2&p3&(!p4))|((!p1)&(!p2)&(!p3)&p4)|(p1&(!p2)&(!p3)&p4)|((!p1)&p2&(!p3)&
p4)|(p1&p2&(!p3)&p4)|((!p1)&(!p2)&p3&p4)|(p1&(!p2)&p3&p4)|((!p1)&p2&p3&p4)|
(p1&p2&p3&p4))
```

Figura 22 – Família H instância 4
Fonte: Autoria própria.

```
T (((p0,0|p0,1)|p0,2)|p0,3)
T (((p1,0|p1,1)|p1,2)|p1,3)
T (((p2,0|p2,1)|p2,2)|p2,3)
T (((p3,0|p3,1)|p3,2)|p3,3)
T (((p4,0|p4,1)|p4,2)|p4,3)
F (p0,0&p1,0)
F (p0,0&p2,0)
F (p0,0&p3,0)
F (p0,0&p4,0)
F (p1,0&p2,0)
F (p1,0&p3,0)
F (p1,0&p4,0)
F (p2,0&p3,0)
F (p2,0&p4,0)
F (p3,0&p4,0)
F (p0,1&p1,1)
F (p0,1&p2,1)
F (p0,1&p3,1)
F (p0,1&p4,1)
F (p1,1&p2,1)
F (p1,1&p3,1)
F (p1,1&p4,1)
F (p2,1&p3,1)
F (p2,1&p4,1)
F (p3,1&p4,1)
F (p0,2&p1,2)
F (p0,2&p2,2)
F (p0,2&p3,2)
F (p0,2&p4,2)
F (p1,2&p2,2)
F (p1,2&p3,2)
F (p1,2&p4,2)
F (p2,2&p3,2)
F (p2,2&p4,2)
F (p3,2&p4,2)
F (p0,3&p1,3)
F (p0,3&p2,3)
F (p0,3&p3,3)
F (p0,3&p4,3)
F (p1,3&p2,3)
F (p1,3&p3,3)
F (p1,3&p4,3)
F (p2,3&p3,3)
F (p2,3&p4,3)
F (p3,3&p4,3)
```

Figura 23 – Família PHP instância 4
Fonte: Autoria própria.

Dentre as famílias de problemas utilizadas, destaca-se a família PHP devido à sua complexidade de prova.

5.2 Resultados família Gamma

Os critérios de avaliação de tempo, tamanho de prova, número de nós, número de bifurcações e uso de memória estão representados em forma gráfica nas figuras: Figura 24, Figura 25, Figura 26, Figura 27 e Figura 28, respectivamente. A discussão dos resultados está descrita na subseção 5.2.1.

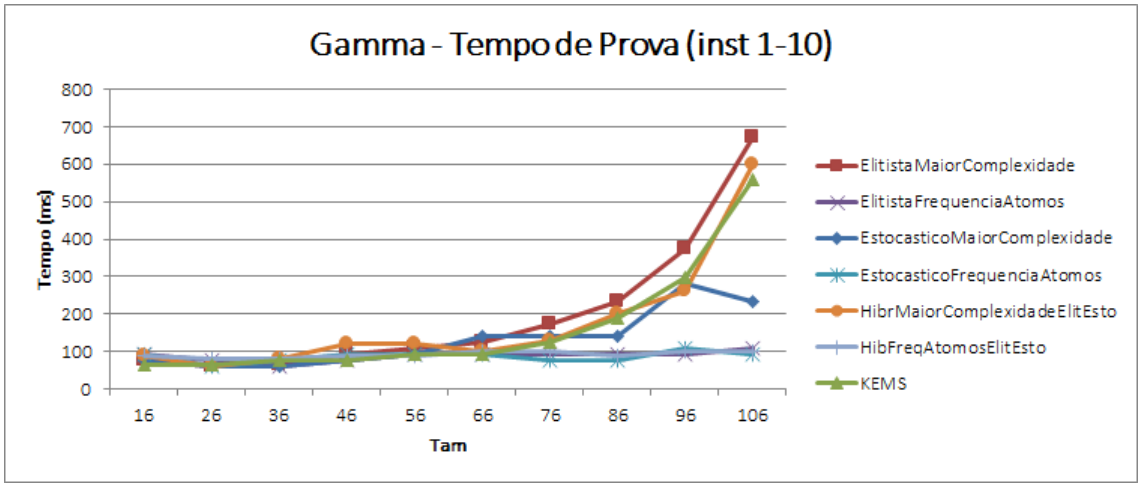


Figura 24 – Na comparação por tempo de prova para a família Gamma, as estratégias: análise de Frequência Átomos Estocástico, Elitista e Híbrida mostraram-se mais eficientes.
Fonte: Autoria própria.

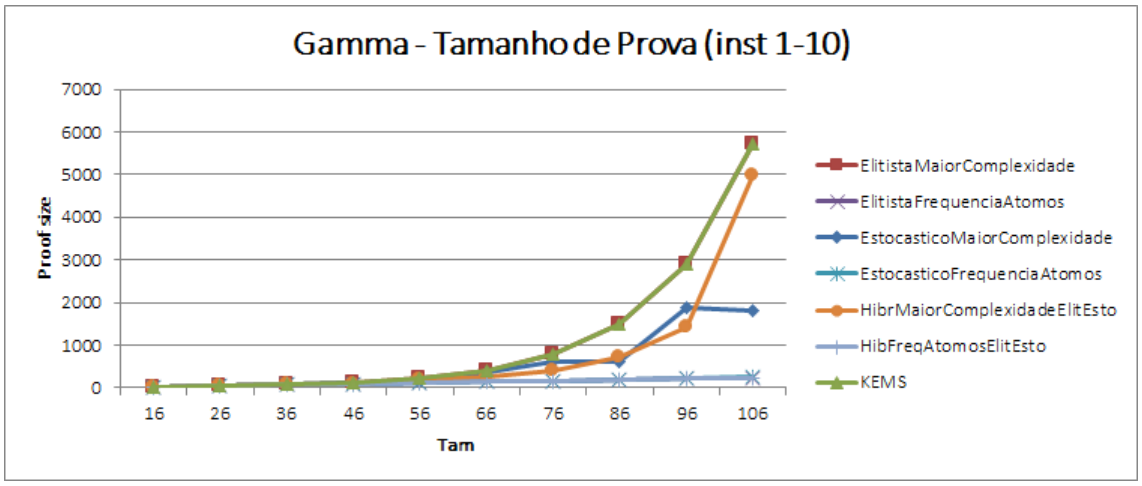


Figura 25 – Na comparação do tamanho da prova para a família Gamma, as estratégias: análise de Frequência Átomos Estocástico, Elitista e Híbrida mostraram-se mais eficientes.
Fonte: Autoria própria.

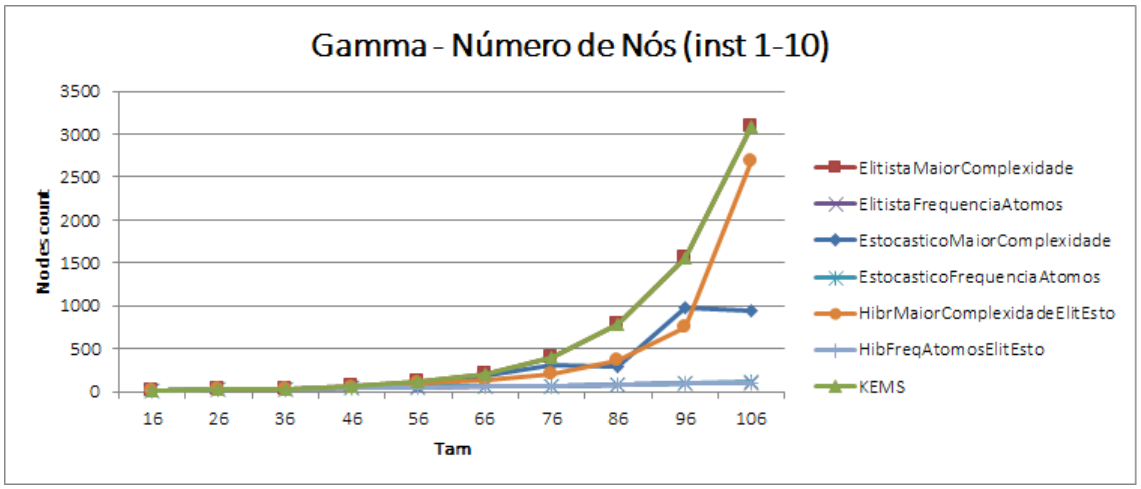


Figura 26 – Na comparação do número de nós para a família Gamma, as estratégias: análise de Frequência Átomos Estocástico, Elitista e Híbrida mostraram-se mais eficientes.
Fonte: Autoria própria.

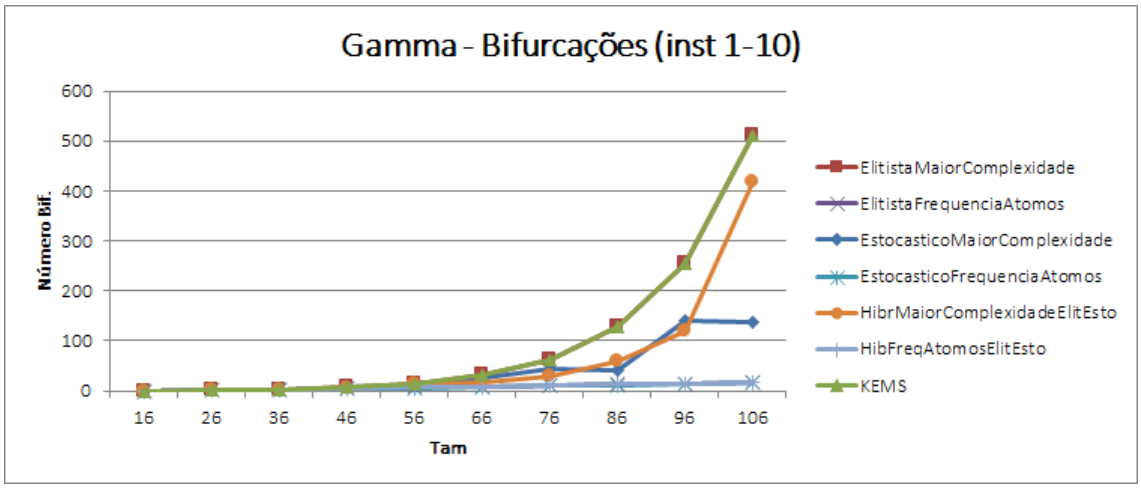


Figura 27 – Na comparação do número de bifurcações para a família Gamma, as estratégias: análise de Frequência Átomos Estocástico, Elitista e Híbrida mostraram-se mais eficientes.
Fonte: Autoria própria.

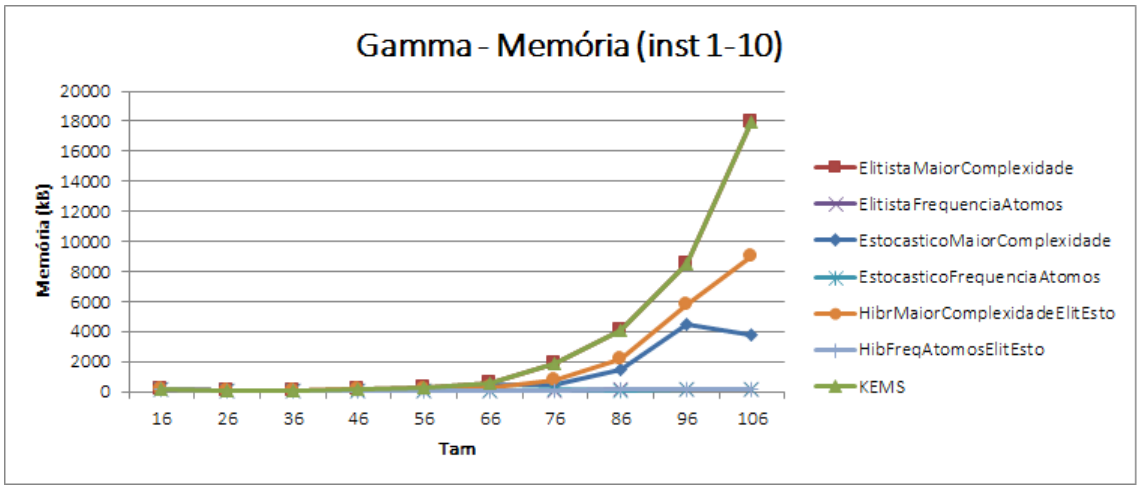


Figura 28 – Na comparação do uso de memória para a família Gamma, as estratégias: análise de Frequência Átomos Estocástico, Elitista e Híbrida mostraram-se mais eficientes.
Fonte: Autoria própria.

5.2.1 Discussão dos resultados família Gamma

Com relação ao consumo de tempo (Figura 24), tamanho de prova (Figura 25), número de nós (Figura 26) e número de bifurcações (Figura 27), as abordagens Estocástica, Elitista e Híbrida de frequência de átomos foram mais eficientes. As demais abordagens ficaram próximas a abordagem convencional apresentada pelo KEMS.

O consumo de memória (Figura 28) da abordagem elitista com função de avaliação por complexidade de fórmula foi o mesmo que o da abordagem convencional. As demais abordagens foram mais eficientes.

Com relação as abordagens com função de avaliação por complexidade de fórmulas, destaca-se a abordagem estocástica. Como a seleção das fórmulas PB candidatas é dada pela roleta, a abordagem encontrou uma solução mais eficiente do que a abordagem elitista com a mesma função de avaliação.

Para a família de problemas Gamma, destacam-se as abordagens Estocástica, Elitista e Híbrida com função de avaliação por frequência de átomos. Estas foram mais eficientes para todos os critérios analisados.

5.3 Resultados família Statman

Os critérios de avaliação de tempo, tamanho de prova, número de nós, número de bifurcações e uso de memória estão representados em forma gráfica nas figuras: Figura 29, Figura 30, Figura 31, Figura 32 e Figura 33. A discussão dos resultados está descrita na subseção 5.3.1.

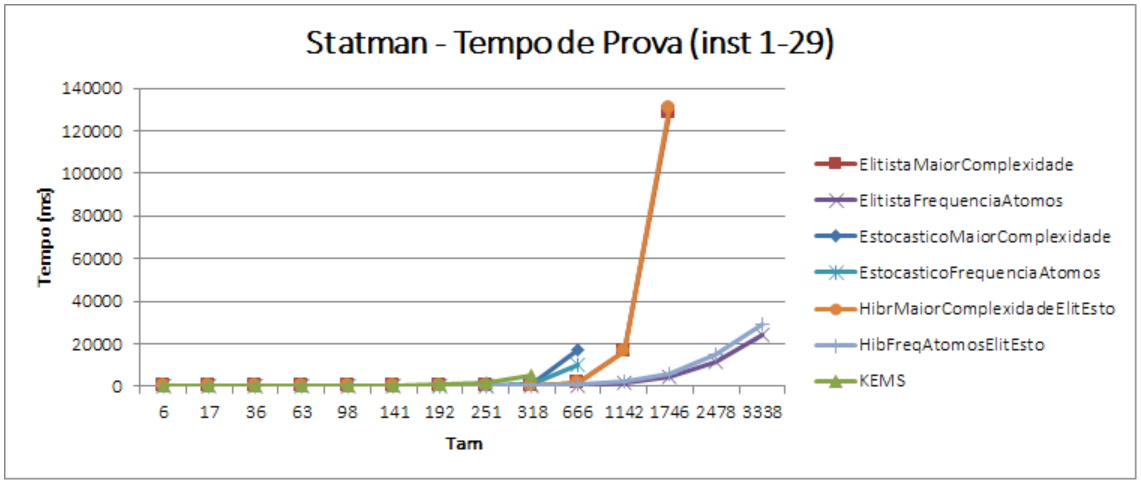


Figura 29 – Na comparação do tempo da prova para a família Statman, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

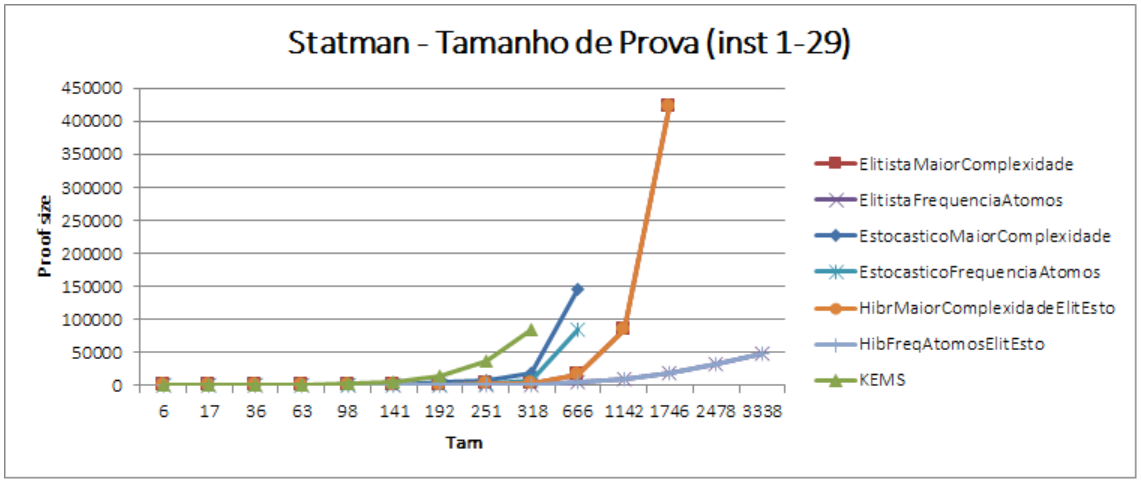


Figura 30 – Na comparação do tamanho da prova para a família Statman, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

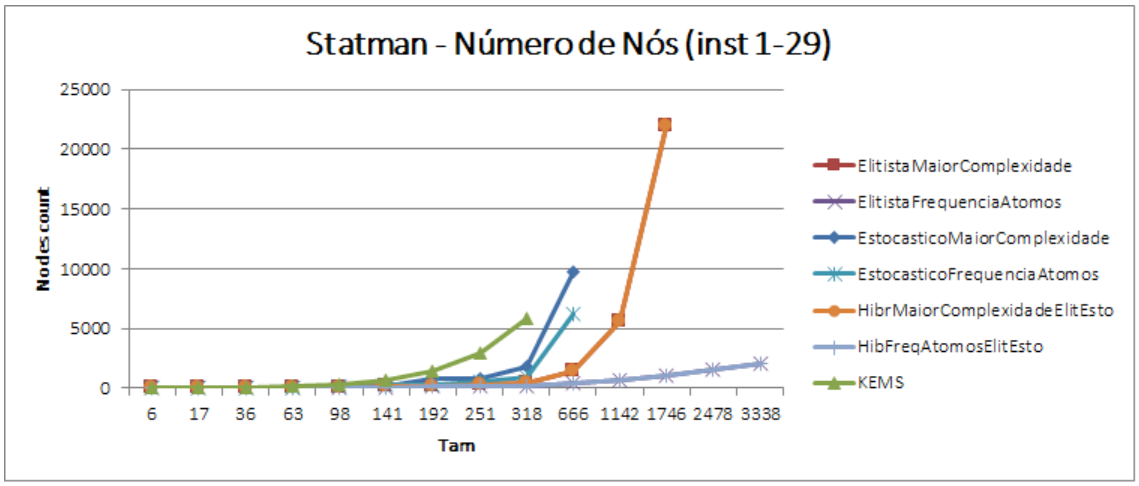


Figura 31 – Na comparação do número de nós para a família Statman, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

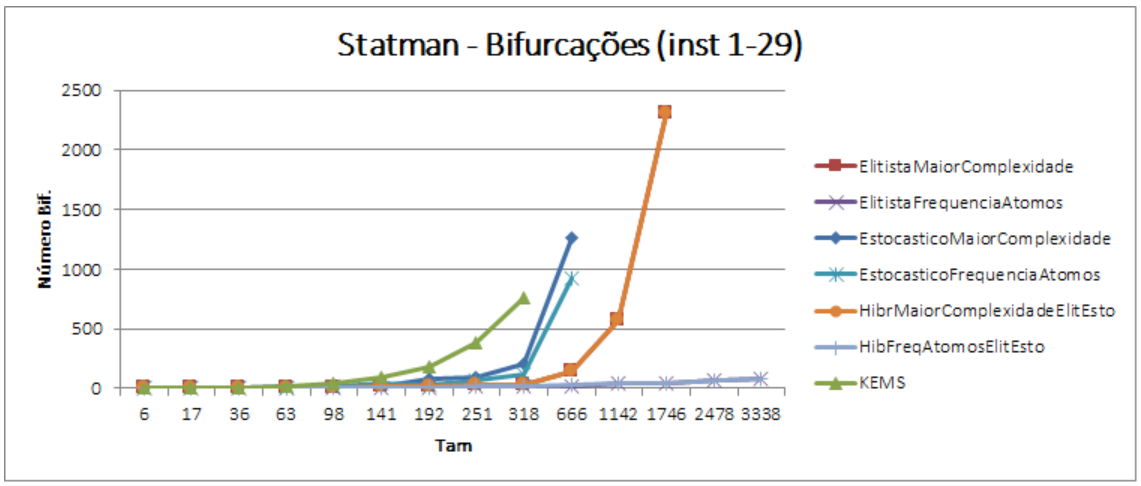


Figura 32 – Na comparação do número de bifurcações para a família Statman, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

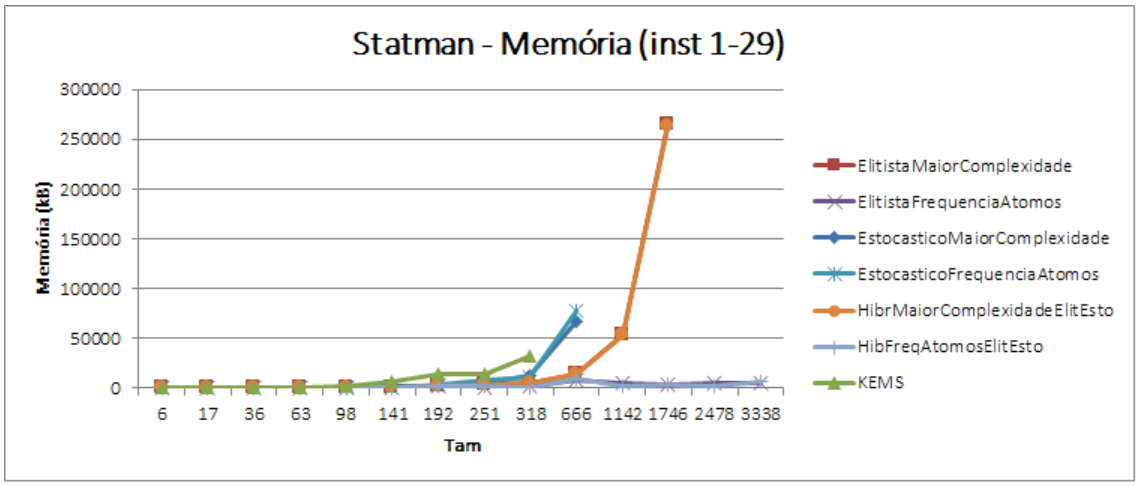


Figura 33 – Na comparação do uso de memória para a família Statman, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

5.3.1 Discussão dos resultados família Statman

Para o tempo máximo de 20 minutos, a metodologia convencional apresentada pelo *software* KEMS provou até a instância 9 da família de problemas Statman. As abordagens estocástica com função de avaliação por complexidade de fórmulas e análise de frequência de átomos provaram até instância 13. As abordagens elitista e híbrida com função de avaliação por complexidade de fórmulas provaram até a instância 21. As abordagens elitista e híbrida com função de avaliação por análise de frequência de átomos provaram até a instância 29.

Em comparação a metodologia convencional de prova, todas as abordagens foram mais eficientes para os critérios analisados.

Para a família de problemas Statman, destacam-se as abordagens elitista e híbrida com função de avaliação por análise de frequência de átomos, as quais provaram todas as instâncias de teste informadas.

5.4 Resultados família H

Os critérios de avaliação de tempo, tamanho de prova, número de nós, número de bifurcações e uso de memória estão representados em forma gráfica nas figuras: Figura 34, Figura 35, Figura 36, Figura 37 e Figura 38. A discussão dos resultados está descrita na subseção 5.4.1.

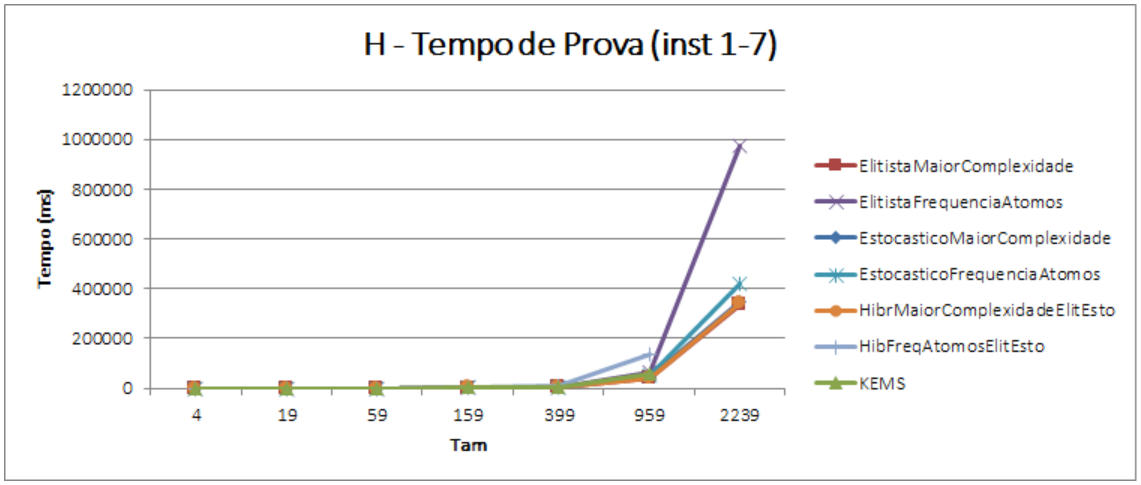


Figura 34 – Na comparação do tempo de prova para a família H, todas as estratégias (exceto a abordagem híbrida com função de avaliação análise de frequência de átomos) mostraram-se mais eficientes.
Fonte: Autoria própria.

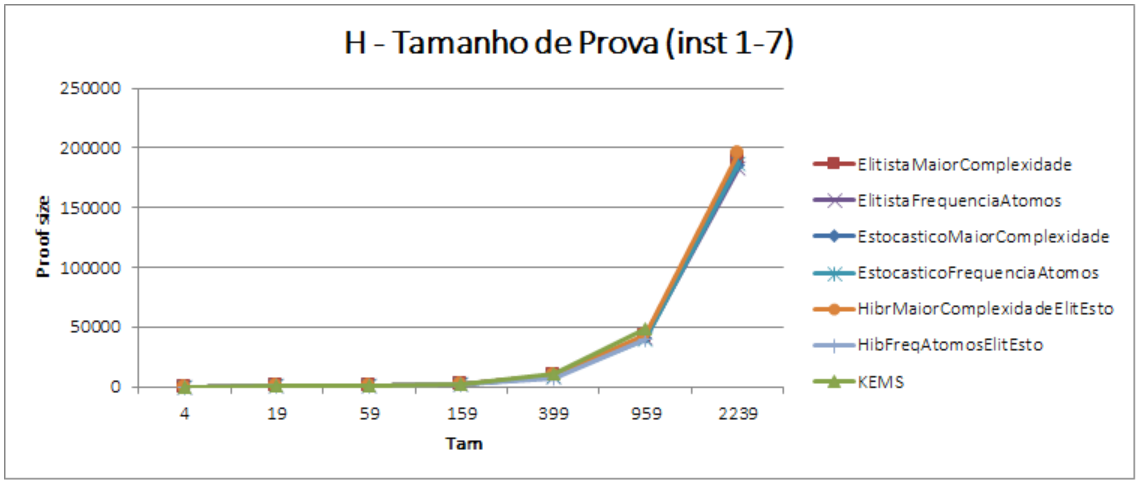


Figura 35 – Na comparação do tamanho da prova para a família H, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

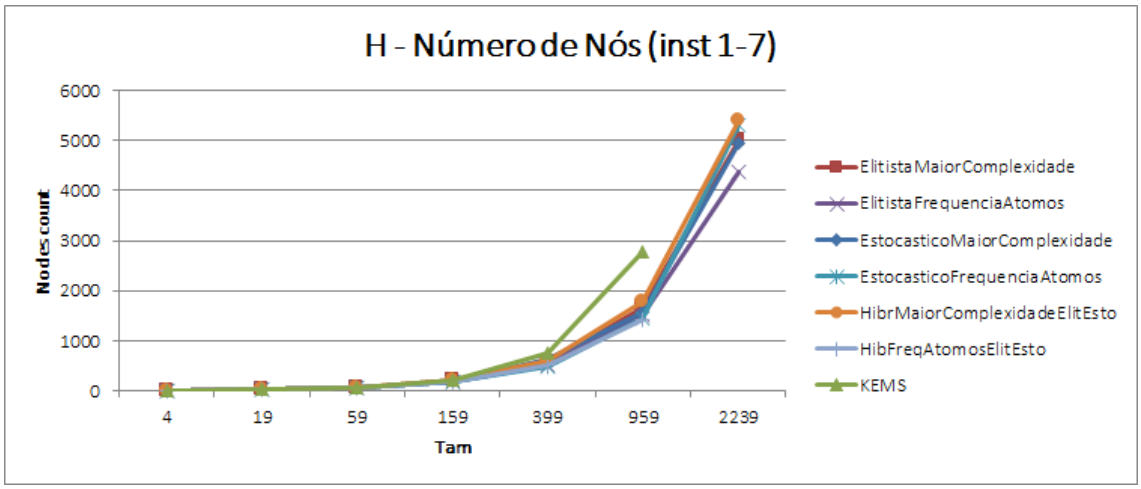


Figura 36 – Na comparação do número de nós para a família H, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

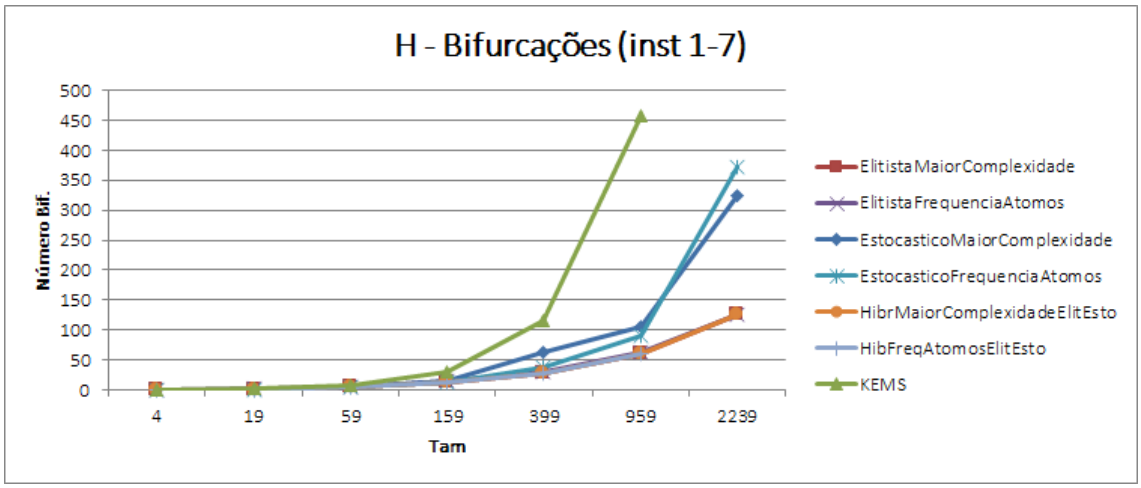


Figura 37 – Na comparação do número de bifurcações para a família H, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

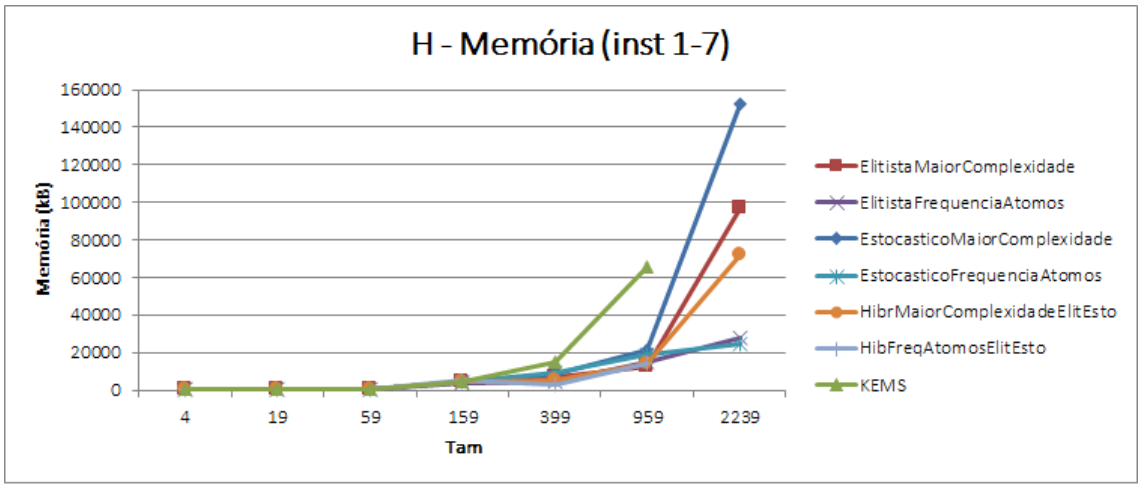


Figura 38 – Na comparação do uso de memória para a família H, todas as estratégias (exceto a abordagem híbrida com função de avaliação análise de frequência de átomos) mostraram-se mais eficientes.
Fonte: Autoria própria.

5.4.1 Discussão dos resultados família H

Para o tempo máximo de 20 minutos, a metodologia convencional apresentada pelo *software* KEMS e a abordagem híbrida com função de avaliação análise de frequência de átomos provaram até a instância 6 da família de problemas H. Todas as demais abordagens propostas provaram até a instância 7.

Para todos os critérios de avaliação, exceto o tempo, a abordagem híbrida com função de avaliação análise de frequência de átomos foi mais eficiente do que a abordagem apresentada pelo KEMS. O tempo de resposta utilizado pela abordagem híbrida com função de avaliação análise de frequência de átomos foi superior a todas as demais abordagens. Isto é devido a homogeneidade das fórmulas PB candidatas, o que onerou a seleção híbrida da abordagem.

Para a família de problemas H, com exceção da abordagem híbrida com função de avaliação análise de frequência de átomos, todas as abordagens foram mais eficientes para os critérios analisados do que a metodologia de prova convencional.

5.5 Resultados família PHP

Os critérios de avaliação de tempo, tamanho de prova, número de nós, número de bifurcações e uso de memória estão representados em forma gráfica nas figuras: Figura 39, Figura 40, Figura 41, Figura 42 e Figura 43. A discussão dos resultados está descrita na subseção 5.5.1.

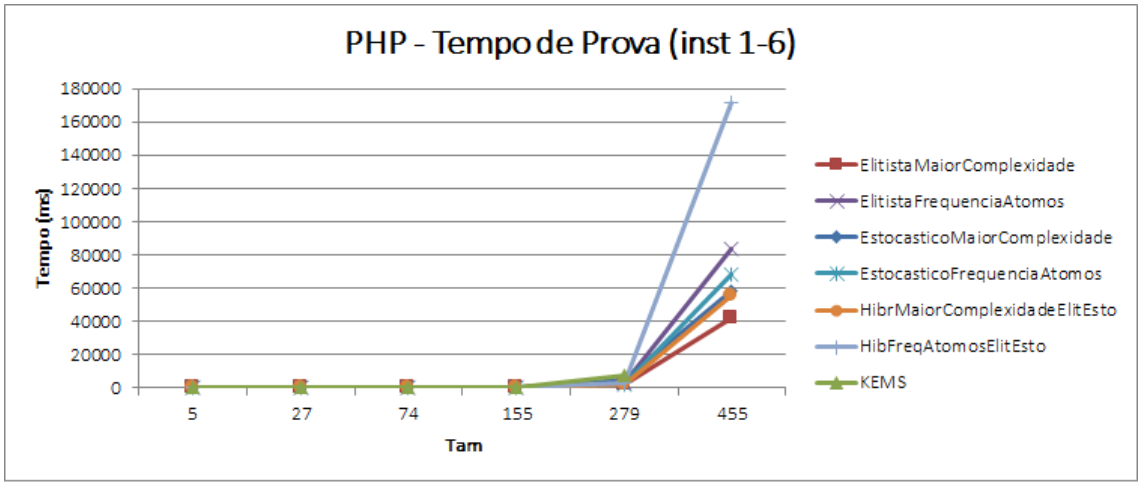


Figura 39 – Na comparação do tempo de prova para a família PHP, todas as estratégias (exceto a abordagem híbrida com função de avaliação análise de frequência de átomos) mostraram-se mais eficientes.
Fonte: Autoria própria.

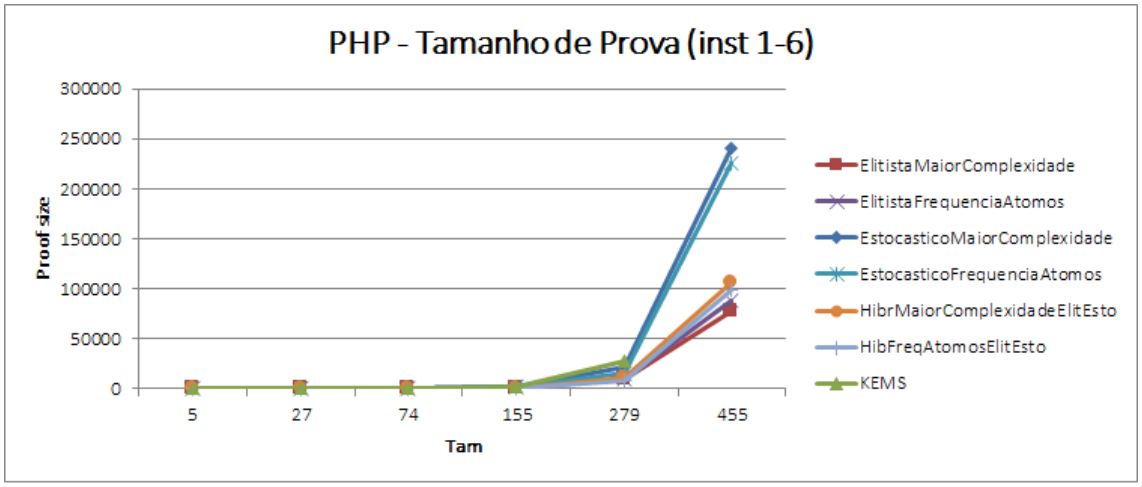


Figura 40 – Na comparação do tamanho da prova para a família PHP, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

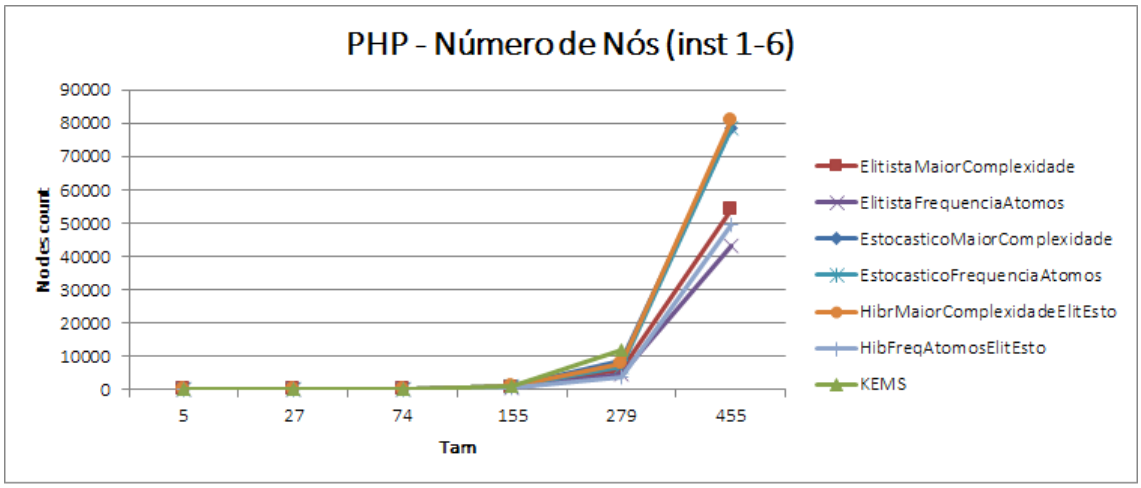


Figura 41 – Na comparação do número de nós para a família PHP, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

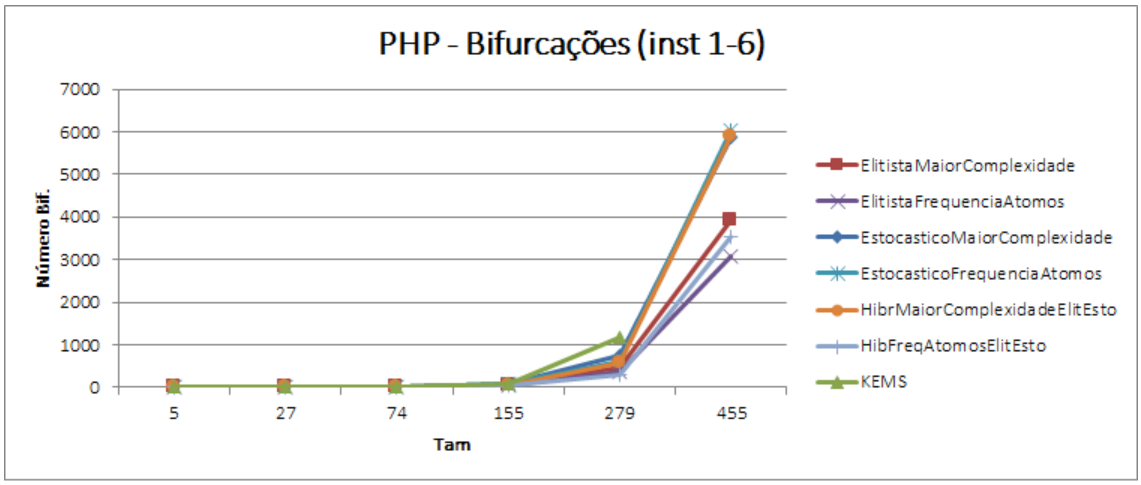


Figura 42 – Na comparação do número de bifurcações para a família PHP, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

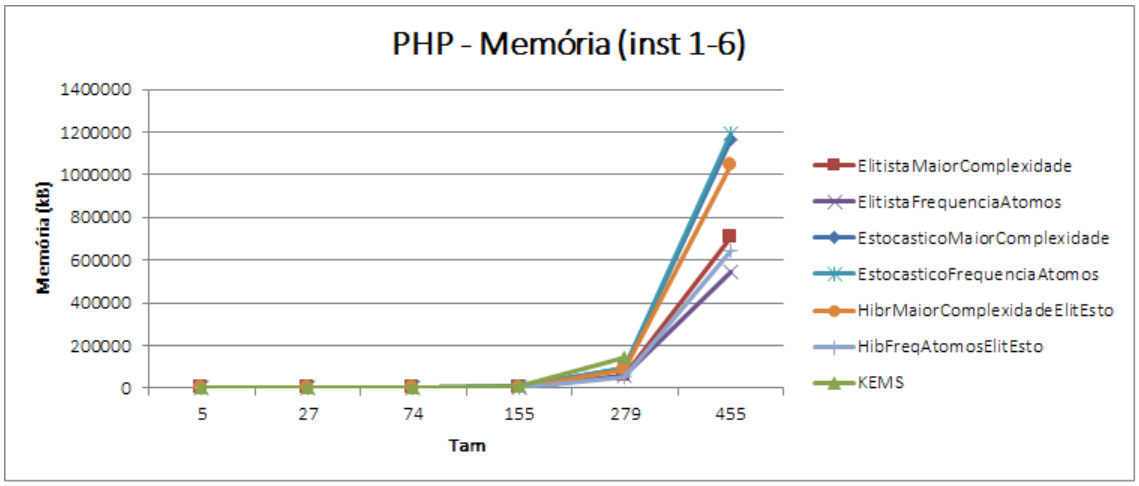


Figura 43 – Na comparação do uso de memória para a família PHP, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

5.5.1 Discussão dos resultados família PHP

Para o tempo máximo de 20 minutos, a metodologia convencional apresentada pelo *software* KEMS provou até a instância 5 da família de problemas PHP. Todas as demais abordagens provaram até a instância 6.

Para a instância 6, a abordagem híbrida com avaliação por análise de frequência de átomos utilizou um maior tempo de resposta. Isto é devido a homogeneidade das premissas da árvore de prova em relação as fórmulas PB candidatas, o que onerou a seleção híbrida da abordagem.

Para a família de problemas PHP todas as abordagens foram mais eficientes, para os critérios analisados, do que a metodologia de prova convencional.

6. CONCLUSÕES

Em resolução de tablôs existem muitas combinações de aplicações de regras em uma base de conhecimento. Cabe ressaltar que a ordem de aplicação das regras influencia diretamente na capacidade de prova de um *software* provador de teoremas, pois dependendo da ordem em que as regras são aplicadas, mais ou menos passos e bifurcações serão necessários para a sua resolução. Isto interfere diretamente nos recursos de memória e tempo de prova gastos na resolução de um tablô.

Este trabalho apresentou três abordagens baseadas em AG que têm como metodologia a seleção de fórmulas PB candidatas que aumentem a probabilidade de que o tablô seja fechado. A metodologia possui uma tendência a aumentar a capacidade de prova de provadores automatizados de teoremas, através de um menor consumo dos recursos de tempo e memória usados na resolução de tablôs.

A seleção elitista parte da premissa que a seleção de fórmulas mais aptas (fórmulas que aumentem a variabilidade da base de conhecimentos) aumente a probabilidade de que o tablô seja fechado. Porém esta técnica de seleção pode limitar-se a uma região do espaço de buscas de forma a prejudicar a eficiência na solução do tablô. Neste sentido a seleção estocástica apresenta uma técnica que permite a escolha de fórmulas menos aptas. O conjunto de classes desenvolvido prevê a ampliação das técnicas de seleção. A interface IEstrategiaAG (Figura 16) permite que novas técnicas de seleção baseadas em AG sejam incorporadas ao projeto de forma simplificada.

A abordagem híbrida combina aspectos das abordagens elitista e estocástica. Neste trabalho, esta abordagem encadeia a seleção elitista e estocástica com a mesma função de avaliação. Outras combinações de encadeamentos podem ser feitos, como, por exemplo, o encadeamento por seleção elitista com função de avaliação por frequência de átomos, seleção elitista com função de avaliação de complexidade e posterior seleção estocástica.

A heurística empregada no presente trabalho baseia-se em AG, porém outras técnicas de busca podem ser utilizadas. Entre elas pode-se citar a busca em feixe. Através desta técnica é possível realizar uma busca em diversas regiões de um espaço de estados, e desta forma, otimizar a processo de busca. Outra técnica que pode ser empregada é a busca de Têmpera simulada. Esta pode simular a seleção estocástica

empregada no presente trabalho, pois de acordo com a temperatura, maior ou menor será a probabilidade de que fórmulas menos aptas sejam escolhidas.

As funções de avaliação propostas no presente trabalho envolvem a análise por frequência de átomos e a avaliação de complexidade das fórmulas. Uma técnica de avaliação mais ampla, e também mais custosa do ponto de vista computacional, é a avaliação de fórmulas baseada no aumento da variabilidade da base de conhecimentos. Por aumento da variabilidade subentende-se a inserção de novas fórmulas à base de conhecimentos através da aplicação da regra PB. Esta técnica de avaliação consiste na aplicação da regra PB em todas as fórmulas PB candidatas e posterior aplicação de todas as regras até que o ramo seja fechado ou que a única regra aplicável seja a regra PB. A avaliação deve levar em consideração o aumento da variabilidade das premissas na base de conhecimentos propiciada pela inserção das novas premissas provenientes da aplicação da regra PB e das regras aplicadas sobre as novas premissas.

Uma simplificação de avaliação baseada na complexidade das fórmulas e de frequência de átomos é utilizada no presente trabalho. Optou-se por esta simplificação por causa das limitações de memória e tempo. A busca e avaliação do aumento da variabilidade implicam diretamente no aumento do consumo do tempo e dos recursos computacionais de processamento e memória.

O presente trabalho abordou a resolução de problemas em sistemas KE, porém não limita-se apenas a este tipo de sistema. Outros sistemas e métodos de prova podem ser incluídos. A técnica empregada baseia-se na premissa de que novas fórmulas aumentam a probabilidade de que uma inconsistência possa ser alcançada. Esta abordagem pode ser aplicada e analisada em outros sistemas.

O objeto de estudo foi a regra PB, porém a técnica de seleção apresentada pode ser ampliada para regras alfa e beta. Regras alfa são as que partem de apenas uma premissa e regras beta de duas. O foco na regra PB é devido ao fato de que ela bifurca o tablô e desta forma aumenta diretamente o tamanho da prova (SMULLYAN, 1995). Desta forma uma técnica de seleção de fórmulas PB candidatas pode influenciar diretamente na otimização de prova de um tablô.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ABE, Jair Minoro; SCALZITTI, Alexandre; SILVA FILHO, João Inácio da. **Introdução à Lógica para a Ciência da Computação**. 2ª edição. 247 p. Editora Arte & Ciência, 2001.

AZEVEDO FILHO, Adriano. **Princípios de Inferência Dedutiva e Indutiva: Noções de Lógica e Métodos de Prova**. 1ª Edição 2010, Scotts Valley: CreateSpace, 148p.

BOUGHACI, Dalila; BENHAMOU, Belïd; DRIAS, Habiba. Scatter Search and Genetic Algorithms for MAX-SAT Problems. **Journal of mathematical modelling and algorithms**, v. 7, n. 2, 2007. Disponível em: <http://www.springerlink.com/content/ak203713447r610j/>. Acesso em: 29 mar. 2012.

CRAWFORD, Kelly D. Solving the n-queens problem using genetic algorithms. **Proceeding SAC '92 Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's**. 1992. Kansas City. Missouri. United States. **Proceedings**… Disponível em: <http://dx.doi.org/10.1145/130069.130128>. Acesso em: 09 mar. 2012.

CARNIELLI, Walter; CONIGLIO, Marcelo E.; Marcos, João. **Logics of formal inconsistency**. São Paulo: Springer-Verlag, 2007 p. 15-107. Disponível em: <ftp://logica.cle.unicamp.br/pub/e-prints/vol.5,n.1,2005-revised.pdf>. Acesso em: 28 mar. 2012.

DEB, Kalyanmoy. **Genetic Algorithm in Search Optimization The Technique and Applications In Proceedings of International Workshop on Soft Computing and Intelligent Systems**. 1997. Calcutá. Índia. **Proceedings of International Workshop on Soft Computing and Intelligent Systems**. Disponível em: <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.5371>. Acesso em: 10 mar. 2012.

DEL CASTANHEL, Lucas; SUGIMOTO, Emerson Shigueo. **Solução para jogo de captura e busca de reféns baseado em redes neurais e planejamento clássico**. 2011. Disponível em: <http://code.google.com/p/si-emerson-lucas/downloads/list>. Acesso em: 22 mar. 2012.

FITTING, M. Introduction. In: D'AGOSTINO ET AL (Eds). **Handbook of Tableau Methods**. Kluwer Academic Press, 1999, p. 1–44.

GOUVEIA, P.; DIONÍSIO F.M.; MARCOS, J. Lógica Computacional. DMIST, 2000.

HEITKOETTER, J.; BEASLEY, D. **The hitch-hiker's guide to evolutionary computation**, 2001. In: LUCAS, Diogo C. **Algoritmos Genéticos: uma Introdução**. UFRGS. 2002. p. 28. Disponível em: <http://www.inf.ufrgs.br/~alvares/INF01048IA/ApostilaAlgoritmosGeneticos.pdf>. Acesso em: 10 mar. 2012.

HOLLAND, John Henry. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.** The MIT Press, 1975. 211 p.

KARNER, Gustav. **Resource Estimation for Objectory Projects**. LINKÖPING. 1993. Disponível em: <http://www.bfpug.com.br/Artigos/UCP/Karner%20-%20Resource%20Estimation%20for%20Objectory%20Projects.doc>. Acesso em: 16 jun. 2012.

KONDO, Tadashi. Feedback GMDH-type neural network using prediction error criterion and its application to 3-dimensional medical image recognition. In: SICE ANNUAL CONFERENCE, 2008, TOKYO. **Proceedings**… Disponível em: <http://dx.doi.org/10.1109/SICE.2008.4654811>. Acesso em: 09 mar. 2012.

LINDEN, Ricardo. **Algoritmos Genéticos – Uma importante ferramenta de Inteligência Computacional**. Editora BRASPORT, Rio de Janeiro, 2006. 1ª Edição. 348 páginas.

LUCAS, Diogo C. **Algoritmos Genéticos: uma Introdução**. UFRGS. 2002. Disponível em: <http://www.inf.ufrgs.br/~alvares/INF01048IA/ApostilaAlgoritmosGeneticos.pdf>. Acesso em: 10 mar. 2012.

MONDADORI, Marco; D'AGOSTINO, Marcello. **The Taming of the Cut. Classical Refutations with Analytic Cut**. Journal of Logic and Computation, vol. 4, number 3, pp. 285-319, 1994.

NETO, Adolfo Gustavo Serra Seca. **Um provador de teoremas multi-estratégia**. 2007. 155 f. Tese (Doutorado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2007. Disponível em: <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04052007-175943/>. Acesso em: 27 mar. 2012.

PACHECO, Marco Aurélio Cavalcanti. **Algoritmos genéticos: princípios e aplicações**. ICA. Laboratório de Inteligência Computacional. Departamento de Engenharia Elétrica. 1999. Rio de Janeiro. Disponível em: <http://www.ica.ele.puc-rio.br/downloads/38/ce-apostila-comp-evol.pdf>. Acesso em: 10 mar. 2012.

PRESSMAN, Roger S. **Engenharia de Software**. 6ª edição. 720 p. Editora McGraw-Hill, 2006.

ROBINSON, John Alan; VORONKOV, Andrei (Eds.). **Handbook of Automated Reasoning**. Volumes 1 e 2. Elsevier and MIT Press, 2001. Editora Gulf Professional Publishing, 2001. 2128 p.

RODRÍGUEZ, M. Andrea. JARUR, Mary Carmen. A Genetic Algorithm for Searching Spatial Configurations. **IEEE Transactions on Evolutionary Computation**. 2005. Chile. **Proceedings**… Disponível em: <http://dx.doi.org/10.1109/TEVC.2005.844157>. Acesso em: 10 mar. 2012.

RUSSELL, S. J. NORVIG, P. **Artificial Intelligence: A Modern Approach**. Pearson Education. 2003.

SILVA, Flávio S. C. da; FINGER, Marcelo; MELO, Ana C. V. de. **Lógica para computação**. São Paulo: Thomson Learning, 2006.

SMULLYAN. **First-Order Logic**. Publicações Dover, 1995. Disponível em: <http://web.doverpublications.com/cgi-bin/toc.pl/0486683702>. Acesso em: 28 mar. 2012.

SOUZA, João Nunes de. **Lógica para ciência da computação: fundamentos de linguagem, semântica e sistemas de dedução**. Rio de Janeiro: Editora Campus, 2002.

SUGIMOTO, Emerson Shigueo. **Implementação de uma Estratégia Eficiente para a Lógica C1 em um Provador de Teoremas Multi-Estratégia**. 2010. 40 f. Relatório final de Atividades - Programa Institucional de Iniciação Científica (PIBIC), Universidade Tecnológica Federal do Paraná. Curitiba, 2010.

_____. **Representação de Fórmulas Lógicas através de Estruturas de Dados**. 2011. 14 f. Relatório final de Atividades - Programa Institucional de Iniciação Científica (PIBIC), Universidade Tecnológica Federal do Paraná. Curitiba, 2011.

THAKUR, Reen. SINGH, Vinay Kr. SINGH, Manu Pratap. Evolutionary design of fuzzy logic controllers with the techniques artificial neural network and genetic algorithm for cart-pole problem. **Computational Intelligence and Computing Research (ICCIC), 2010 IEEE International Conference on**. 2011. Índia. **Proceedings**… Disponível em: <http://dx.doi.org/10.1109/ICCIC.2010.5705756>. Acesso em: 10 mar. 2012.

TANOMARU, Julio. Motivação, Fundamentos e Aplicações de Algoritmos Genéticos. **II Congresso Brasileiro de Redes Neurais. III Escola de Redes Neurais**. Disponível em: <http://bogdano.irreais.net/crap/pub/tutorial_ag.pdf>. Acesso em: 10 mar. 2012.

TURKYL, Ayad M. AHMAD, Mohd Sharifuddin. Using Genetic Algorithm for Solving N-Queens Problem. **Information Technology (ITSim), 2010 International Symposium**. 2010. Iraque. **Proceedings**… Disponível em: <http://dx.doi.org/10.1109/ITSIM.2010.5561604>. Acesso em: 09 mar. 2012.

ZHANG, Zhen-Zhen. HUANG, Wei-Hsiu. LIN, Jyun-Jie. CHANG, Pei-Chann. WU, Jheng-Long. A Puzzle-Based Artificial Chromosome Genetic Algorithm for the Traveling Salesman Problem. **Technologies and Applications of Artificial Intelligence (TAAI), 2011 International Conference on**. 2011. China. **Proceedings**… Disponível em: <http://dx.doi.org/10.1109/TAAI.2011.58>. Acesso em: 10 mar. 2012.

APÊNDICE A – RECURSOS DE HARDWARE E *SOFTWARE*

Os recursos de *hardware* estão descritos na seção 1 e os recursos de *software* na seção 1.1.

1. Recursos de *Hardware*

Para análise da estratégia baseada em Algoritmos Genéticos através do KEMS, será necessária uma máquina com as características capacidade de processamento semelhantes a 2 GB de memória RAM, processador Intel(R) Core(TM) 2 Quad Q6600 @ 2.40GHz e sistema operacional Windows ou Linux.

1.1 Recursos de *Software*

Este trabalho utiliza como base computacional o *software* (programa) KEMS que é um provador de teoremas multi-estratégia baseado no método KE (MONDADORI e D'AGOSTINO, 1994) implementado em Java[1] e AspectJ[2]. Java é uma linguagem de programação orientada a objetos (JAVA, 2012) e AspectJ uma linguagem de programação que estende Java com os conceitos da orientação a aspectos.

Para o desenvolvimento foi utilizado o *software* Eclipse[3], que permite a edição e compilação de código fonte Java.

A Máquina Virtual do Java (JVM – *Java Virtual Machine*[4]) foi utilizada para desenvolvimento em Java.

---

[1] JAVA. Linguagem de Programação. Disponível em: <http://java.com/pt_BR/about/>. Acesso em: 25 mar. 2012.

[2] ASPECTJ. Documentação da linguagem orientada à aspectos. Disponível em: <http://www.eclipse.org/aspectj/doc/released/index.html#paths>. Acesso em: 28 abr. 2010.

[3] ECLIPSE. *Software* para de desenvolvimento em linguagem Java. Disponível em <http://www.eclipse.org/>. Acesso em: 28 mar. 2012.

Para o controle de versões do *software* KEMS, foi utilizado o sistema de controle de versões Git[5]. O site Github[6] foi utilizado para controle de versões através de tecnologia Git. Este site permite de forma gratuita a hospedagem de arquivos e o controle de versões de projetos de forma *online* e entre diversos colaboradores.

Para análise e testes das classes implementadas no *software* KEMS, a tecnologia JUNIT[7] de testes padronizados em Java foi utilizada. Através dela podem ser criados testes em lote para validar as classes desenvolvidas.

---

[4] JVM. Java Virtual Machine. Disponível em: <http://java.sun.com/javase/6/docs/technotes/guides/vm/index.html?intcmp=3170>. Acesso em: 3 mar. 2012

[5] GIT. Controle de Versionamento de Projetos. Disponível em: <http://git-scm.com/>. Acesso: em 25 mar. 2012.

[6] GITHUB. Site que oferece um serviço online gratuito de hospedagem de arquivos e de controle de versões. Disponível em: <http://github.com/>. Acesso em: 20 mar. 2012.

[7] JUNIT. Testes automatizados em Java. Disponível em: <http://www.junit.org/>. Acesso em: 4 mar. 2012.

APÊNDICE B – PROJETO DE *SOFTWARE*

Os requisitos funcionais e não funcionais estão descritos nas seções 2 e 2.1, respectivamente.

A seção 3 descreve o diagrama de casos de uso, a seção 3.1 descreve os pontos de casos de uso, a seção 4 descreve o diagrama de Classes, a seção 5 descreve o diagrama de objetos e a seção 6 descreve o diagrama de sequências.

2. Requisitos Funcionais

O *software* deve ser capaz de:

- Identificar se um ramo esta fechado ou não.
- Conhecer o conjunto de premissas de cada ramo.
- Listar um conjunto de regras aplicáveis ao conjunto de fórmulas de um ramo, baseado no sistema em estudo.
- Conhecer o conjunto de fórmulas resultantes da aplicação das regras.
- Permitir que a seleção pelo usuário de qual abordagem será aplicada.
- Avaliar a complexidade computacional das premissas.
- Avaliar e selecionar uma fórmula a partir de um conjunto de fórmulas.

2.1 Requisitos Não Funcionais

Os requisitos não funcionais são:

- A seleção das fórmulas deve basear-se no valor de avaliação das mesmas.
- A abordagem elitista deve selecionar fórmulas que possuam o maior valor de *fitness*.

- A abordagem estocástica deve utilizar a roleta como forma de seleção de um indivíduo.
- O Sistema deve, para ambas as abordagens, fornecer uma resposta em tempo finito menor ou igual ao tempo de prova pelos métodos convencionais.
- O método de seleção por roleta deverá indicar um indivíduo com base na valoração de *fitness*.

3. Diagrama de Casos de Uso

O diagrama de casos de uso **Selecionar Fórmula** está representado na Figura 1. Este diagrama representa a interação entre o *software* KEMS e a abordagem proposta. O Quadro 1 apresenta a descrição do caso de uso.

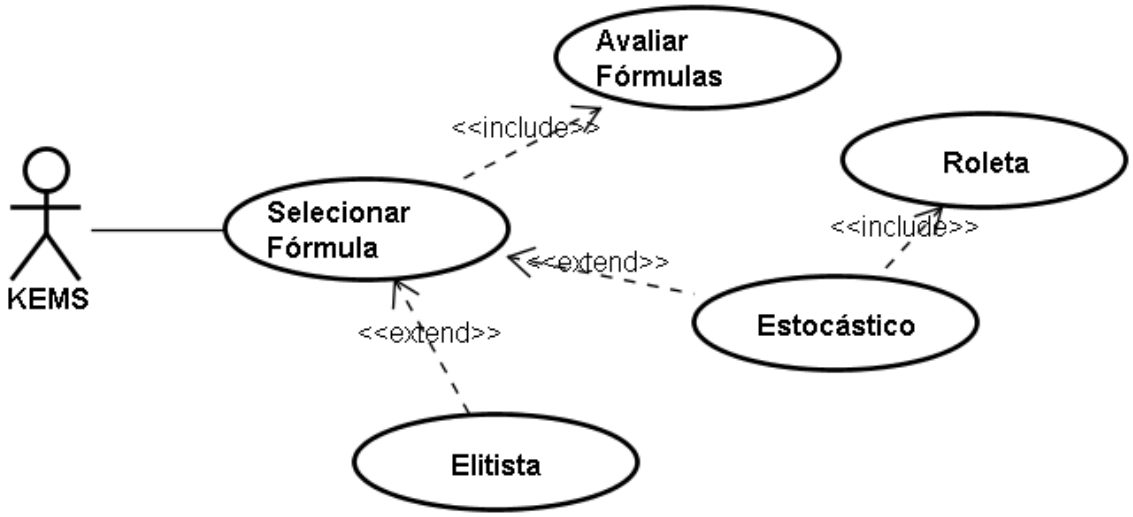


Figura 1 – Caso de Uso selecionar fórmula
Fonte: Autoria própria.

Nome do Caso de Uso	Selecionar Fórmula
Caso de Uso Geral	
Ator Principal	KEMS
Atores Secundários	
Resumo	Este caso de uso descreve os passos necessários para a seleção de uma fórmula
Pré-Condições	Conjunto de Fórmulas não nulo
Pós-Condições	Fórmulas Avaliadas Fórmula Selecionada
Ações do Ator	Ações do Sistema
1. Informa conjunto de Fórmulas	
2. Informa a abordagem AG que será usada	
	3. Usa o Caso de Uso Avaliar Fórmulas
	4. Usa o Caso de Uso Estocástico ou Elitista e retorna uma fórmula
Restrições/Validações	1. Conjunto de Fórmulas não nulo.

Quadro 1 – Selecionar fórmula
Fonte: Autoria própria.

O fluxo básico é descrito na Figura 2.

1. *Software* KEMS solicita a seleção de uma fórmula (ator), e informa os seguintes dados:
   a. A estratégia AG que será usada (elitista, estocástica ou híbrida).
   b. Lista de Fórmulas PB candidatas (interface IFormulas)
2. O caso de uso AvaliarFormula avalia as Fórmulas (sistema). O fluxo é direcionado para 2a ou 2b.
   a. Acesso ao caso de uso Elitista. (sistema).
   b. Acesso ao caso de uso Estocástico. (sistema).
3. O sistema retorna uma fórmula selecionada (interface IFormulas) (sistema)

Figura 2 – Fluxo básico
Fonte: Autoria própria.

Os fluxos alternativos estão descritos nas figuras: Figura 3 e Figura 4.

1. Usuário Seleciona a abordagem Elitista. (ator)
2. O fluxo básico segue para 2a no fluxo básico (Figura 2). (sistema)

Figura 3 – Fluxo alternativo 1
Fonte: Autoria própria.

3. Usuário Seleciona a abordagem Estocástica. (ator)
4. O fluxo básico segue para 2b no fluxo básico (Figura 2). (sistema)

Figura 4 – Fluxo alternativo 2
Fonte: Autoria própria.

3.1 *Use Case Points*

O *Use Case Points* (UCP) é uma técnica para medição do tamanho funcional de sistemas proposta por Gustav Karner (1993).

O valor de *Unadjusted Actor Weight* (UAW) (KARNER, 1993) do diagrama de casos de uso da Figura 1 é calculado no Quadro 2. Este valor traduz a complexidade do caso de uso com base nos atores.

Tipo de Ator	**Peso**	**Número de Atores**	**Resultado**
Simples	1	0	0
Médio	2	0	0
Complexo	3	1	3
Total de UAW			3

Quadro 2 – Cálculo de UAW
Fonte: Autoria própria.

O valor de *Unadjusted Use Case Weight* (UUCW) (KARNER, 1993) do diagrama de casos de uso da Figura 1 é calculado no Quadro 3.

Complexidade	**Peso**	**Número de Casos de Uso**	**Resultado**
Casos de Uso Simples	5	4	20
Casos de Uso Médio	10	1	10
Casos de Uso Complexo	15	0	0
Total de UUCW			30

Quadro 3 – Cálculo de UUCW
Fonte: Autoria própria.

O cálculo de *Unadjusted Use Case Point* (UUCP) é representado na Figura 5.

UUCP = *Unadjusted Actor Weight* (UAW) + *Unadjusted Use Case Weight* (UUCW)

Figura 5 – Cálculo de UUCP
Fonte: KARNER (1993).

Desta forma o valor de UUCP para o presente projeto é de 33. O Fator de Complexidade Técnica (*Technical Complexity Factor* - TCF) é calculado utilizando-se a fórmula representada na Figura 6.

TCF = 0,6 + (0,01 x TFactor)

Figura 6 – Cálculo de TCF
Fonte: KARNER (1993).

O cálculo de TFactor está descrito no Quadro 4. O valor calculado de TCF (Figura 6) é de 0,92.

Fator	**Fatores de Complexidade**	**Peso**	**Valor**	**Total**
F1	Sistemas Distribuídos	2	0	0
F2	Tempo de Resposta	1	3	3
F3	Eficiência para o usuário final (*online*)	1	2	2
F4	Processamento interno complexo	1	2	2
F5	Código reusável	1	5	5
F6	Facilidade de Instalação	0,5	2	1
F7	Facilidade de uso (facilidade operacional)	0,5	2	1
F8	Portabilidade	2	3	6
F9	Facilidade de Mudança	1	5	5
F10	Concorrência (acesso simultâneo à aplicação)	1	5	5
F11	Recursos de segurança	1	1	1
F12	Fornece acesso direto para terceiros	1	0	0
F13	Requer treinamento especial para o usuário	1	1	1
TFactor				32

Quadro 4 – Cálculo de TFactor
Fonte: Autoria própria.

O Fator de Complexidade de Ambiente (*Environmental Factor* - EF) é calculado utilizando-se a fórmula representada na Figura 7.

EF = 1.4 + (-0.03 x Efactor)

Figura 7 – Cálculo de TCF
Fonte: KARNER (1993).

O cálculo de EFactor está descrito no Quadro 5. O valor calculado de EF (Figura 7) é de 0,68.

Fator	Fatores de Complexidade	Peso	Valor	Total
F1	Familiaridade da equipe com o processo formal de desenvolvimento adotado	1,5	4	6
F2	Colaboradores de meio período	-1	0	0
F3	Capacidade do líder do projeto em análise de requisitos e modelagem	0,5	4	2
F4	Experiência da equipe em desenvolvimento de aplicações do gênero em questão	0,5	4	2
F5	Experiência em Orientação a Objetos	1	5	5
F6	Motivação da Equipe	1	3	3
F7	Dificuldade com a linguagem de programação	-1	0	0
F8	Requisitos estáveis	2	3	6
EFactor				24

Quadro 5 – Cálculo de EFactor
Fonte: Autoria própria.

O valor total do sistema em *Use Case Points* (UCP) ajustados é dado pela fórmula representada na Figura 8.

UCP = UUCP x TCF x EF

Figura 8 – Cálculo de UCP
Fonte: KARNER (1993).

O valor de UCP calculado é de 20,6. Karner (1993) sugere estimar o tempo necessário para o desenvolvimento de um projeto com uma média de 20 horas de trabalho por Ponto de Caso de Uso (UCP).

O cálculo do tempo de trabalho estimado é de UCP x 20. Desta forma o tempo estimado é de 413 horas de trabalho.

4. Diagrama de Classes

O diagrama de classes da estratégia AG proposta está representada na Figura 9. O diagrama de objetos está representado na seção 5.

Os diagramas de classes da estrutura de fórmulas utilizado pelo *software* KEMS estão ilustrados na Figura 10 e Figura 11.

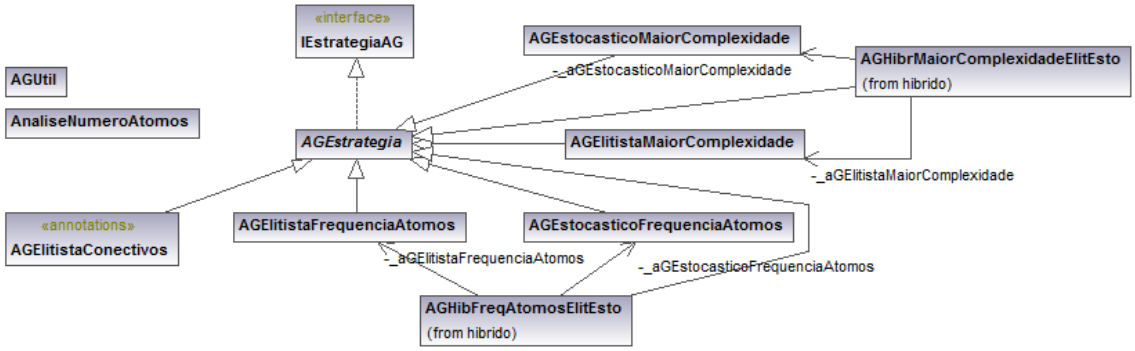


Figura 9 - Diagrama de Classes Estratégia AG
Fonte: Autoria própria.

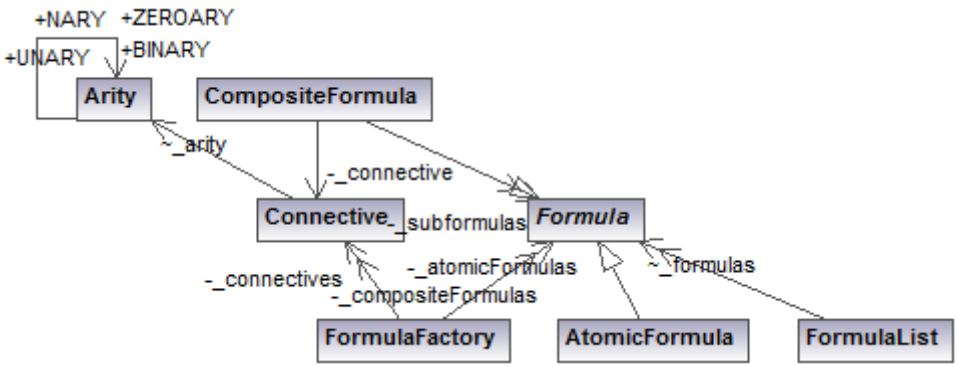


Figura 10 - Diagrama de Classes Formula KEMS
Fonte: Adaptado do *software* KEMS (NETO, 2007).

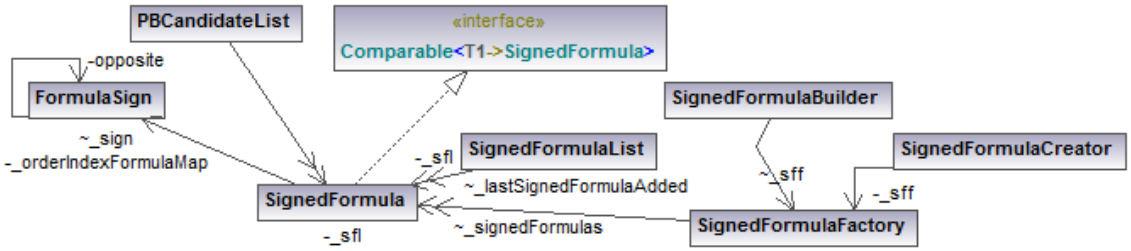


Figura 11 - Diagrama de Classes SignedFormula KEMS
Fonte: Adaptado do *software* KEMS (NETO, 2007).

As principais classes que representam as premissas utilizadas pelo KEMS são as classes SignedFormula (Figura 14), PBCandidateList (Figura 15), CompositeFormula (Figura 13) e AtomicFormula (Figura 12).

A classe AtomicFormula (Figura 12) representa os átomos das fórmulas lógicas, como por exemplo: A, B, C, etc.

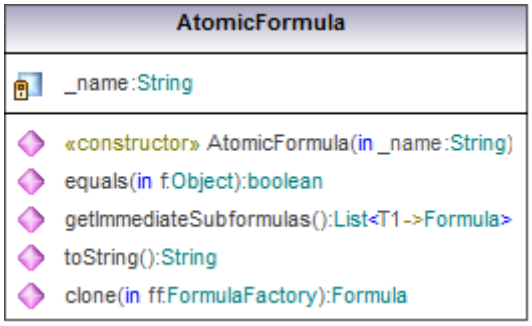


Figura 12 – Classe AtomicFormula
Fonte: Adaptado do *software* KEMS (NETO, 2007).

A classe CompositeFormula (Figura 13) representa a formação de fórmulas compostas, formadas por um conectivo e outras fórmulas atômicas ou compostas. Exemplo: A → B, C ∨ D, (A → B) ∨ C ∨ D.

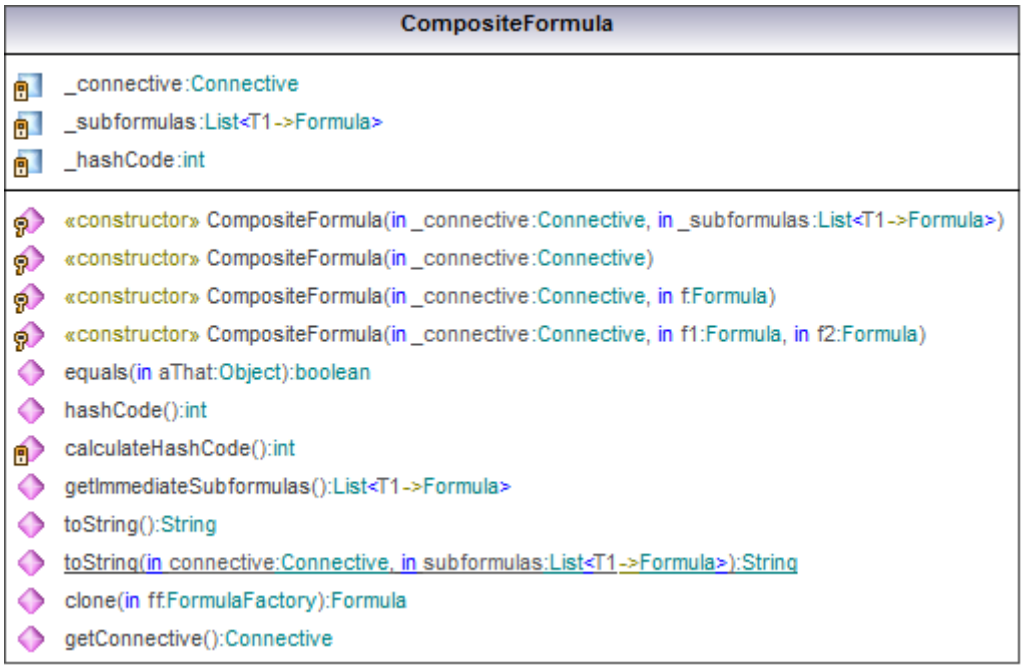


Figura 13 – Classe CompositeFormula
Fonte: Adaptado do *software* KEMS (NETO, 2007).

A classe SignedFormula (Figura 14) representa fórmulas assinaladas. Como por exemplo: T A → B, F C, etc.

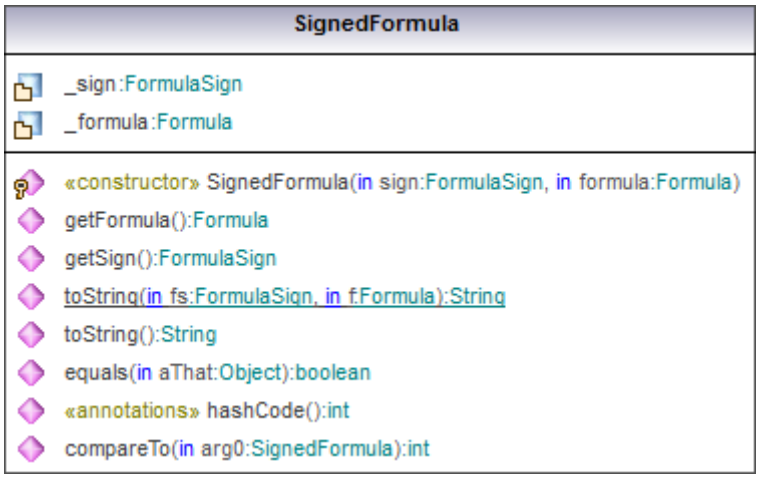


Figura 14 – Classe SignedFormula
Fonte: Adaptado do *software* KEMS (NETO, 2007).

A classe PBCandidateList (Figura 15) contém a lista de fórmulas PB candidatas de um ramo de prova. A lista de fórmulas PB candidatas utiliza a classe SignedFormula (Figura 14) para representar as premissas.

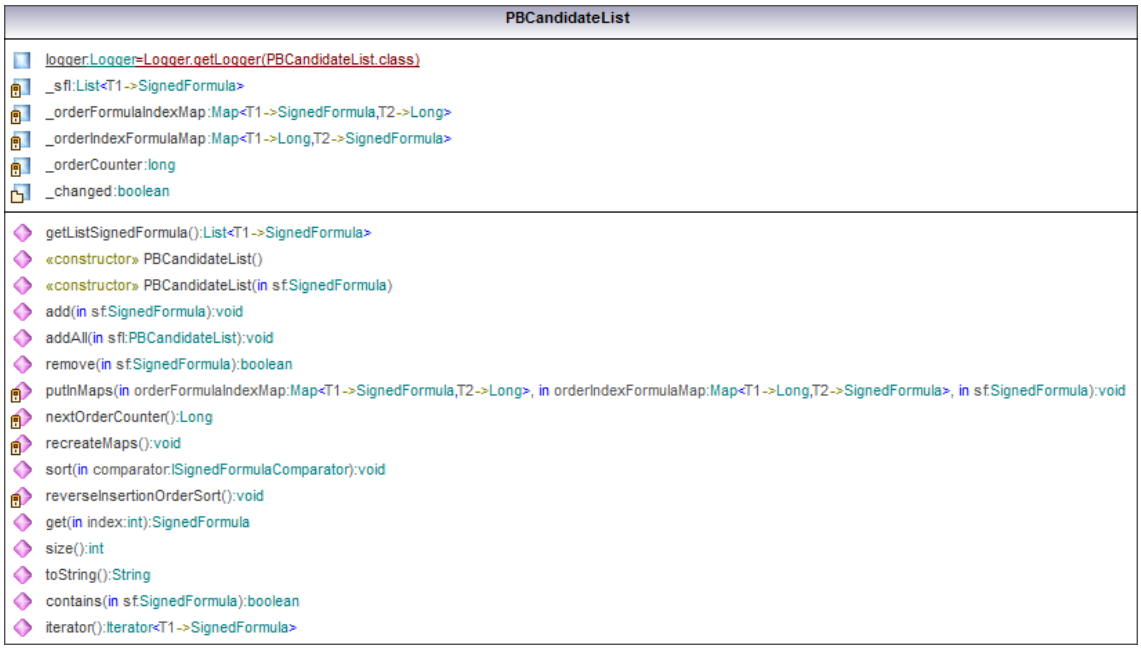


Figura 15 – Classe PBCandidateList
Fonte: Adaptado do *software* KEMS (NETO, 2007).

A interface pública dos métodos das estratégias de AG desenvolvidas no presente trabalho é descrita pela interface IEstrategiaAG (Figura 16). O dicionário de informações está descrito no Quadro 6. Esta interface abstrai publicamente os métodos das estratégias de AG.

Figura 16 – Interface IEstrategiaAG
Fonte: Autoria própria.

Interface:	IEstrategiaAG	
Método	Descrição	Retorno
getPblist	O objeto PBCandidateList contém o conjunto de fórmulas PB candidatas.	Retorna a lista do tipo PBCandidateList
setPblist(PBCandidateList pblist)	Atualiza o objeto PBCandidateList.	
getStrategy	A Estratégia contém os ramos de prova.	Retorna a Estratégia: ISimpleStrategy
setStrategy(ISimpleStrategy strategy)	Atualiza a Estratégia	
getListaFormulasJaSelecionadas	Lista de controle das fórmulas já escolhidas.	ArrayList<SignedFormula>
setListaFormulasJaSelecionadas(ArrayList<SignedFormula> listaFormulasJaSelecionadas)	Atualiza a lista de fórmulas já escolhidas	
getSignedFormula	Retorna a fórmula da lista de fórmulas PB candidatas (PBCandidateList) de acordo com a estratégia AG.	SignedFormula

Quadro 6 - Dicionário da interface IEstrategiaAG
Fonte: Autoria própria.

Os métodos comuns das abordagens AG da interface IEstrategiaAG (Figura 16) são implementados pela classe abstrata AGEstrategia (Figura 17). A classe abstrai a função getSignedFormula(), que é intrínseca a cada estratégia AG específica. O dicionário de informações está descrito no Quadro 7.

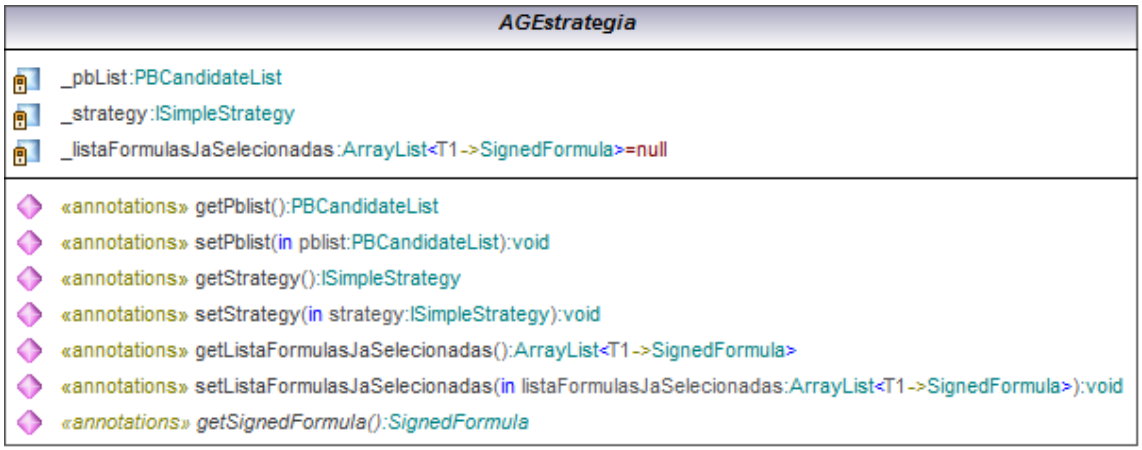


Figura 17 – Classe abstrata AGEstrategia
Fonte: Autoria própria.

Classe:	AGEstrategia	
Atributo	Descrição	Tipo
_pbList	Lista de fórmulas PB candidatas	PBCandidateList
_strategy	Estratégia	ISimpleStrategy
_listaFormulasJaSelecionadas	Lista de fórmulas selecionadas	ArrayList<SignedFormula>
Método	Descrição	Retorno
getSignedFormula	Método abstrato	Retorna a fórmula da lista de fórmulas PB candidatas (PBCandidateList) de acordo com a estratégia AG.

Quadro 7 - Dicionário da classe abstrata AGEstrategia
Fonte: Autoria própria.

A classe AGElitistaConectivos (Figura 18) implementa o método abstrato getSignedFormula() da interface IEstrategiaAG (Figura 16). O dicionário de informações está descrito no Quadro 8. Esta classe seleciona a fórmula PB candidata que possuir o maior número de conectivos. Como exemplo, para o conjunto de premissas:

- TA ∧ C, 1 conectivo.
- FA → C ∨ D, 2 conectivos.

A premissa escolhida é FA → C ∨ D, com dois conectivos.

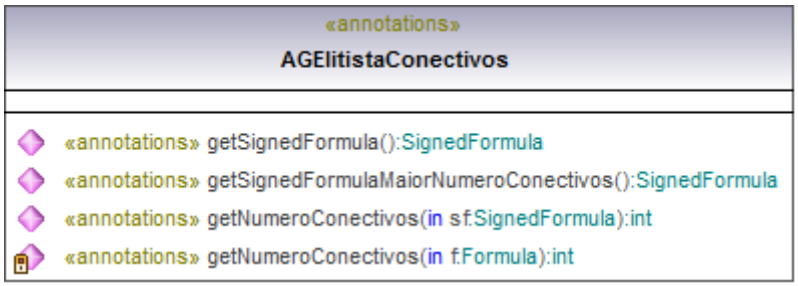


Figura 18 – Classe AGElitistaConectivos
Fonte: Autoria própria.

Classe:	AGElitistaConectivos	
Método	Descrição	Retorno
getSignedFormula	Seleciona a fórmula com maior número de conectivos da lista PBCandidateList	SignedFormula
getSignedFormulaMaiorNumeroConectivos	Analisa o número de conectivos das fórmulas da lista PB.	Retorna a fórmula com o maior número de conectivos da lista PBCandidateList
getNumeroConectivos	Conta o número de conectivos de uma Formula	Int: número de conectivos

Quadro 8 - Dicionário da classe AGElitistaConectivos
Fonte: Autoria própria.

A classe AGUtil (Figura 19) possui métodos comuns as estratégias de AG propostas. O dicionário de informações está descrito no Quadro 9.

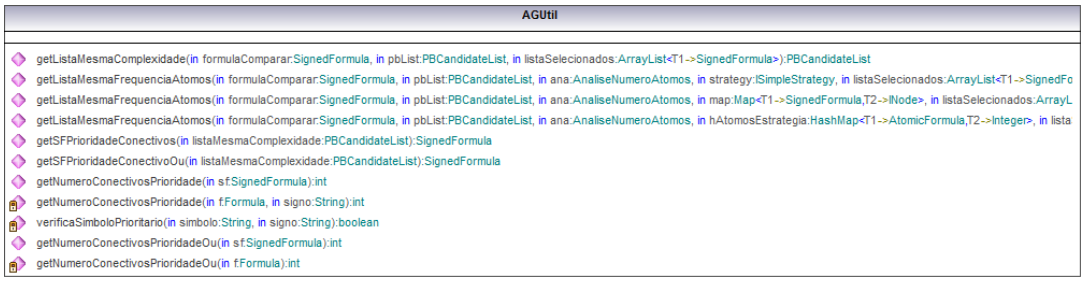


Figura 19 – Classe AGUtil
Fonte: Autoria própria.

Classe:	AGUtil	
Método	Descrição	Retorno
getListaMesmaComplexidade(SignedFormula formulaComparar, PBCandidateList pbList, ArrayList<SignedFormula> listaSelecionados)	Retorna uma lista de fórmulas que possuem a mesma valoração de *fitness* de complexidade da fórmula formulaComparar	PBCandidateList
getListaMesmaFrequenciaAtomos(SignedFormula formulaComparar, PBCandidateList pbList, AnaliseNumeroAtomos ana, ISimpleStrategy strategy, ArrayList<SignedFormula> listaSelecionados)	Retorna lista de fórmulas que possuem a mesma valoração de *fitness* de análise de frequência da átomos.	PBCandidateList

Quadro 9 - Dicionário da classe AGUtil
Fonte: Autoria própria.

A classe AGConfiguration (Figura 20) atua como *factory* de estratégias AG. A interface de estratégias IEstrategiaAG está descrita na Figura 16. O dicionário de informações está descrito no Quadro 10.

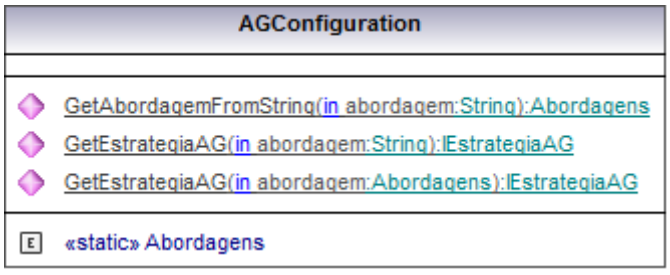


Figura 20 – Classe AGConfiguration
Fonte: Autoria própria.

Classe:	AGConfiguration	
Método	Descrição	Retorno
GetAbordagemFromString(String abordagem)	Recupera a abordagem a partir da String com o nome da abordagem	enum Abordagens
GetEstrategiaAG(String abordagem)	Retorna a estratégia a partir da String com o nome da abordagem	IEstrategiaAG
GetEstrategiaAG(Abordagens abordagem)	Retorna a estratégia a partir do enum Abordagens	IEstrategiaAG

Quadro 10 - Dicionário da classe AGConfiguration
Fonte: Autoria própria.

A estratégia de AG baseada na frequência dos átomos das fórmulas PB candidatas está representada na seção 4.1.

A estratégia de AG baseada na complexidade das fórmulas PB candidatas está representada na seção 4.2.

4.1 Estratégia AG Frequência de Átomos

A classe AGElitistaFrequenciaAtomos (Figura 21) seleciona a fórmula PB candidata que possuir a maior avaliação dada pela análise de frequência de átomos. A avaliação por análise de frequência de átomos é descrita na subseção 3.1.3. A seleção elitista está descrita na subseção 3.1.5. O dicionário de informações está descrito no Quadro 11.

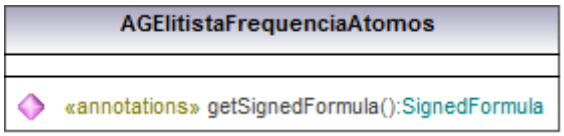


Figura 21 – Classe AGElitistaFrequenciaAtomos
Fonte: Autoria própria.

Classe:	AGElitistaFrequenciaAtomos	
Método	Descrição	Retorno
getSignedFormula	Seleciona com base na frequência de átomos das fórmulas PB candidatas e das fórmulas da árvore de prova	SignedFormula

Quadro 11 - Dicionário da classe AGElitistaFrequenciaAtomos
Fonte: Autoria própria.

A classe AGEstocasticoFrequenciaAtomos (Figura 22) utiliza a função de avaliação baseada na análise de frequência de átomos de forma estocástica através da roleta. A avaliação de cada fórmula PB candidata exerce um peso na probabilidade da sua escolha. A seleção estocástica é descrita na subseção 3.1.6. O dicionário de informações está descrito no Quadro 12.

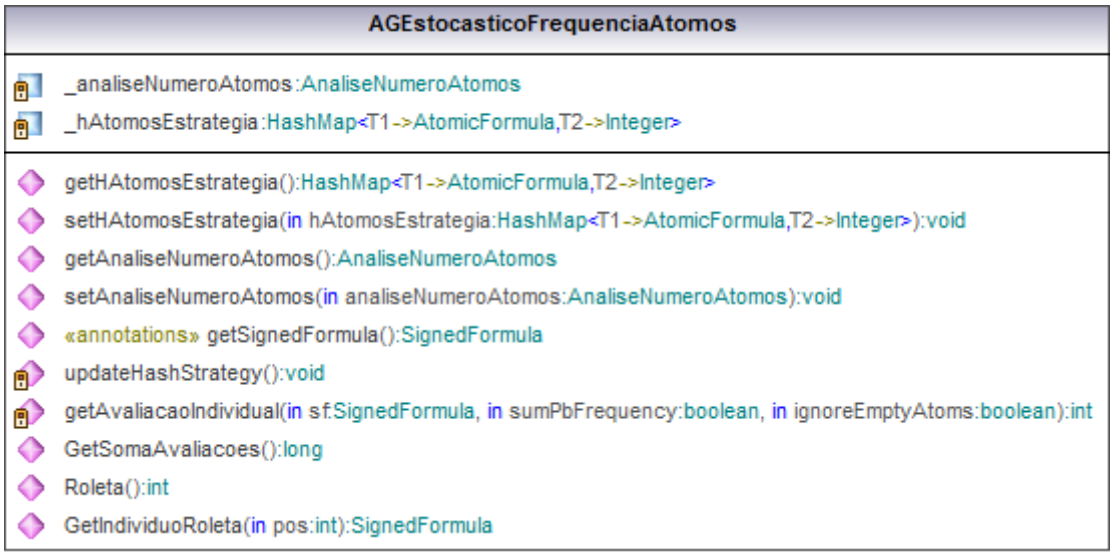


Figura 22 – Classe AGEstocasticoFrequenciaAtomos
Fonte: Autoria própria.

Classe:	AGEstocasticoFrequenciaAtomos	
Método	Descrição	Retorno
getSignedFormula	Retorna a Fórmula de acordo com a roleta	SignedFormula
updateHashStrategy	Contém a frequência dos átomos das fórmulas PB candidatas	
getAvaliacaoIndividual(SignedFormula sf)	A avaliação é feita com base na frequência dos átomos das fórmulas PB candidatas e na frequência dos átomos das fórmulas da árvore de prova	Avaliação de uma Fórmula.
GetSomaAvaliacoes	Soma as avaliações individuais de cada fórmula	Soma de avaliação
Roleta	Sorteia a posição do indivíduo na roleta. Leva em consideração a avaliação de cada fórmula.	Posição do indivíduo na roleta
GetIndividuoRoleta(int pos)	Recupera o indivíduo a partir da sua posição da roleta.	SignedFormula

Quadro 12 - Dicionário da classe AGEstocasticoFrequenciaAtomos
Fonte: Autoria própria.

A classe AnaliseNumeroAtomos (representada na Figura 23) possui funções de suporte as classes AGElitistaFrequenciaAtomos (Figura 21) e AGEstocasticoFrequenciaAtomos (Figura 22). Sua principal função é analisar as frequências dos átomos das fórmulas das premissas da base de conhecimentos, das fórmulas PB candidatas e selecionar uma fórmula com base na avaliação da análise de frequências dos átomos (subseção 3.1.3). O dicionário de informações está descrito no Quadro 13.

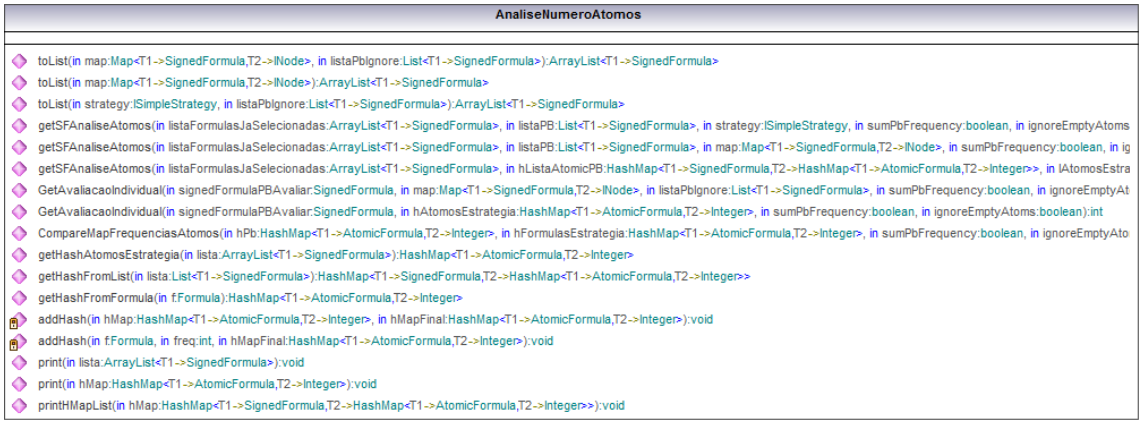


Figura 23 – Classe AnaliseNumeroAtomos
Fonte: Autoria própria

Classe:	AnaliseNumeroAtomos	
Método	Descrição	Retorno
toList(Map<SignedFormula, INode> map, List<SignedFormula> listaPbIgnore)	Cria a lista de fórmulas da árvore de prova	ArrayList<SignedFormula>
getSFAnaliseAtomos(ArrayList<SignedFormula> listaFormulasJaSelecionadas, List<SignedFormula> listaPB, ISimpleStrategy strategy)	Retorna a fórmula PB candidata com maior valoração para a análise de frequência de átomos.	Fórmula
GetAvaliacaoIndividual	Retora a avaliação indivídual de uma fórmula	int: *fitness* de uma fórmula
CompareMapFrequenciasAtomos	Compara a frequência de dois *hashs*	int: análise de frequências dos átomos das fórmulas de cada *hash*.
getHashFromFormula(Formula f)	Cria um *hash* de frequências dos átomos de uma fórmula	HashMap<AtomicFormula, Integer>

Quadro 13 - Dicionário da classe AnaliseNumeroAtomos
Fonte: Autoria própria.

4.2 Estratégia AG Maior Complexidade

A classe AGElitistaMaiorComplexidade (Figura 24) seleciona a fórmula PB candidata com a maior complexidade. A avaliação por complexidade de fórmulas está descrita na subseção 3.1.2. A seleção elitista está descrita na subseção 3.1.5. O dicionário de informações está descrito no Quadro 14.

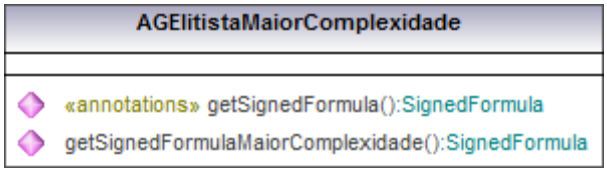


Figura 24 – Classe AGElitistaMaiorComplexidade
Fonte: Autoria própria.

Classe:	AGElitistaMaiorComplexidade	
Método	Descrição	Retorno
getSignedFormula	Retorna a Fórmula de maior *fitness*	SignedFormula

Quadro 14 - Dicionário da classe AGElitistaMaiorComplexidade
Fonte: Autoria própria.

A classe AGEstocasticoMaiorComplexidade (Figura 25) utiliza a mesma função de avaliação que a abordagem usada na classe AGElitistaMaiorComplexidade (Figura 24). A diferença está na aplicação da seleção das fórmulas PB candidatas através da roleta, de forma semelhante a classe AGEstocasticoFrequenciaAtomos (Figura 22). A seleção estocástica é descrita na subseção 3.1.6. O dicionário de informações está descrito no Quadro 15.

Figura 25 – Classe AGEstocasticoMaiorComplexidade
Fonte: Autoria própria.

Classe:	AGEstocasticoMaiorComplexidade	
Método	Descrição	Retorno
getSignedFormula	Retorna a Fórmula de acordo com a roleta	SignedFormula
GetSomaAvaliacoes	Soma as avaliações individuais de cada fórmula	Soma de avaliação
Roleta	Sorteia a posição do indivíduo na roleta. Leva em consideração a avaliação de cada fórmula.	Posição do indivíduo na roleta
GetIndividuoRoleta(int pos)	Recupera o indivíduo a partir da sua posição da roleta.	SignedFormula

Quadro 15 - Dicionário da classe AGEstocasticoMaiorComplexidade
Fonte: Autoria própria.

4.3 Estratégia AG Híbrida

A classe AGHibFreqAtomosElitEsto (Figura 26) utiliza uma função de seleção mista. A primeira etapa avalia as fórmulas PB candidatas de acordo com a análise de frequência dos átomos descrita na seção 4.1. Caso existam fórmulas com o mesmo valor de *fitness*, a seleção será estocástica entre as fórmulas PB candidatas de mesma valoração, caso contrário, a fórmula de maior valoração será a escolhida. A seleção híbrida é descrita na subseção 3.1.7. O dicionário de informações está descrito no Quadro 16.

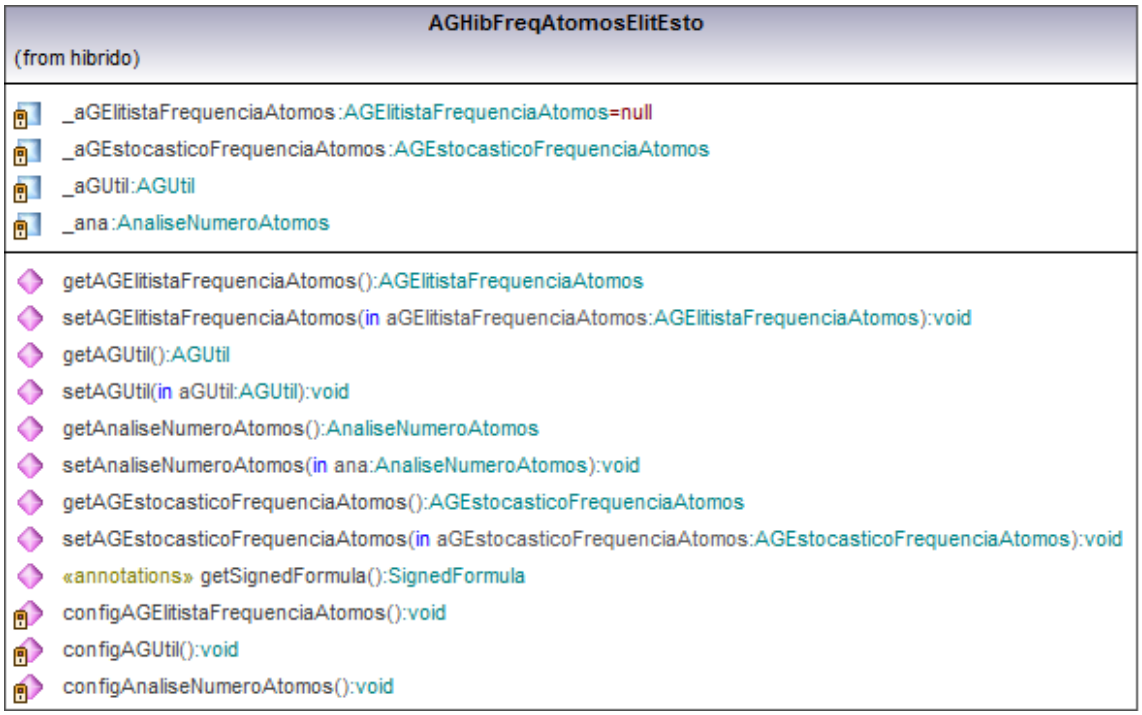


Figura 26 – Classe AGHibFreqAtomosElitEsto
Fonte: Autoria própria.

Classe:	AGHibFreqAtomosElitEsto	
Método	Descrição	Retorno
getSignedFormula	Retorna a Fórmula de acordo com a análise de frequência dos átomos. Usa uma abordagem híbrida elitista e estocástica.	SignedFormula

Quadro 16 - Dicionário da classe AGHibFreqAtomosElitEsto
Fonte: Autoria própria.

A classe AGHibrMaiorComplexidadeElitEsto (Figura 27) é semelhante a classe AGHibFreqAtomosElitEsto (Figura 26). A diferença esta na função de avaliação, que usa a valoração da complexidade das fórmulas PB candidatas. De forma idêntica, caso existam fórmulas com o mesmo valor de *fitness*, a seleção será estocástica entre as fórmulas PB candidatas de mesma valoração, caso contrário, a fórmula de maior valoração será a escolhida. O dicionário de informações está descrito no Quadro 17.

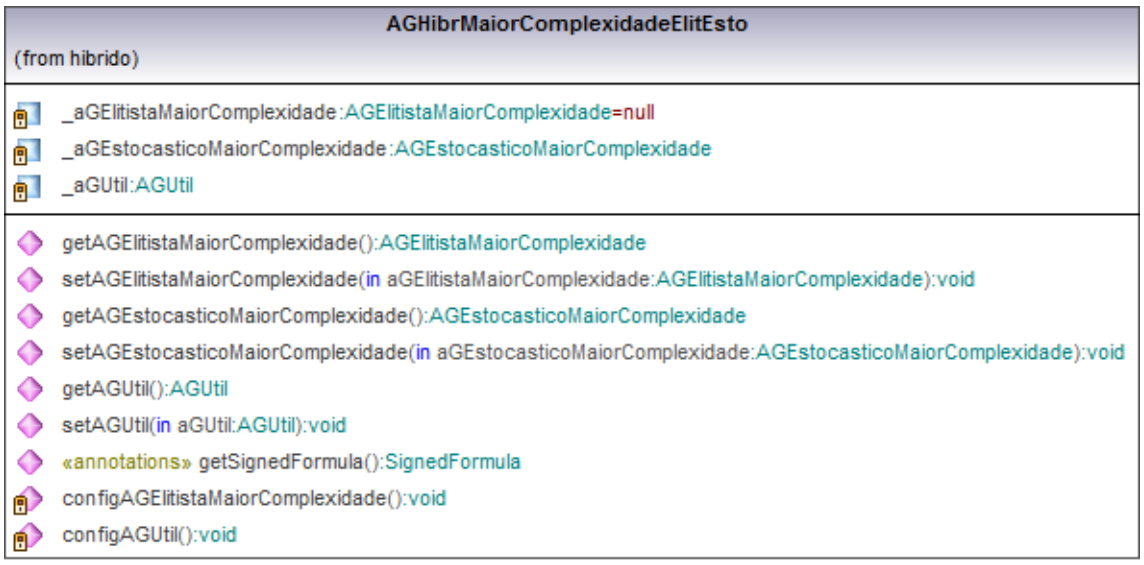


Figura 27 – Classe AGHibrMaiorComplexidadeElitEsto
Fonte: Autoria própria.

Classe:	AGHibrMaiorComplexidadeElitEsto	
Método	Descrição	Retorno
getSignedFormula	Retorna a Fórmula de acordo com a avaliação de complexidade das fórmulas. Usa uma abordagem híbrida elitista e estocástica.	SignedFormula

Quadro 17 - Dicionário da classe AGHibrMaiorComplexidadeElitEsto
Fonte: Autoria própria.

5. Diagrama de Objetos

O diagrama de objetos da classe AGElitistaFrequenciaAtomos (Figura 28) demonstra a interação entre os objetos das classe AGElitistaFrequenciaAtomos (Figura 21), SignedFormula (Figura 14), PBCandidateList (Figura 15) e AtomicFormula (Figura 12).

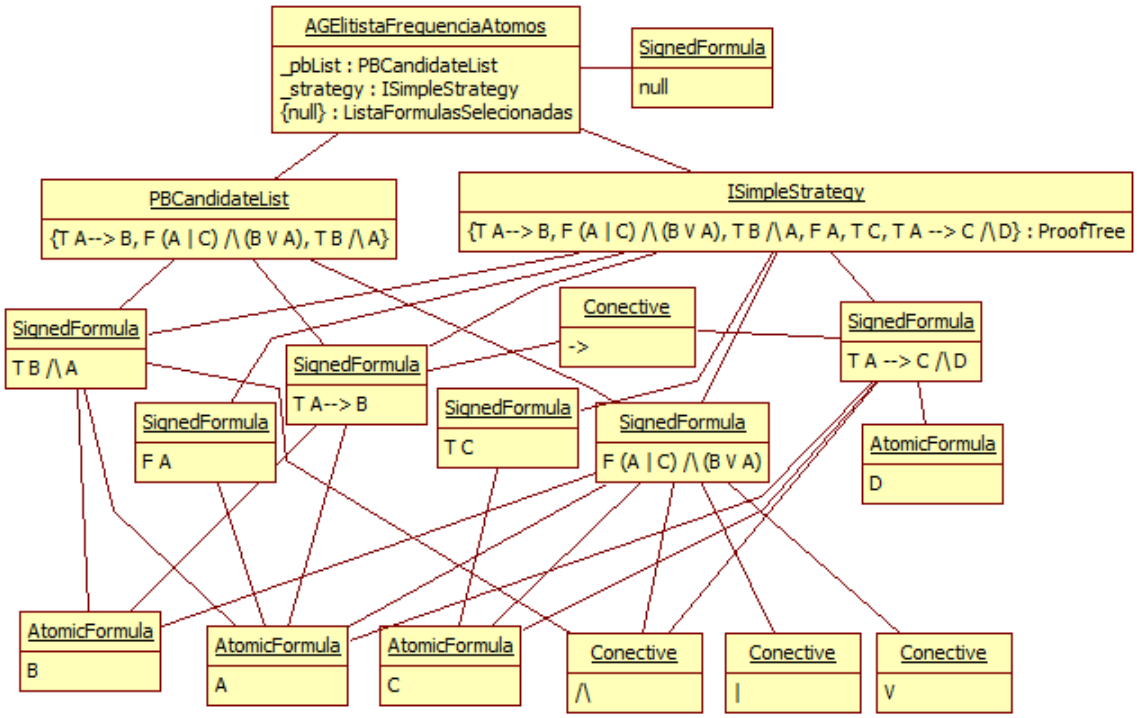


Figura 28 - Diagrama de Objetos AGElitistaFrequenciaAtomos
Fonte: Autoria própria.

O diagrama de objetos da classe AGEstocasticoFrequenciaAtomos (Figura 29) demonstra a interação entre os objetos das classe AGEstocasticoFrequenciaAtomos (Figura 22), SignedFormula (Figura 14), PBCandidateList (Figura 15) e AtomicFormula (Figura 12).

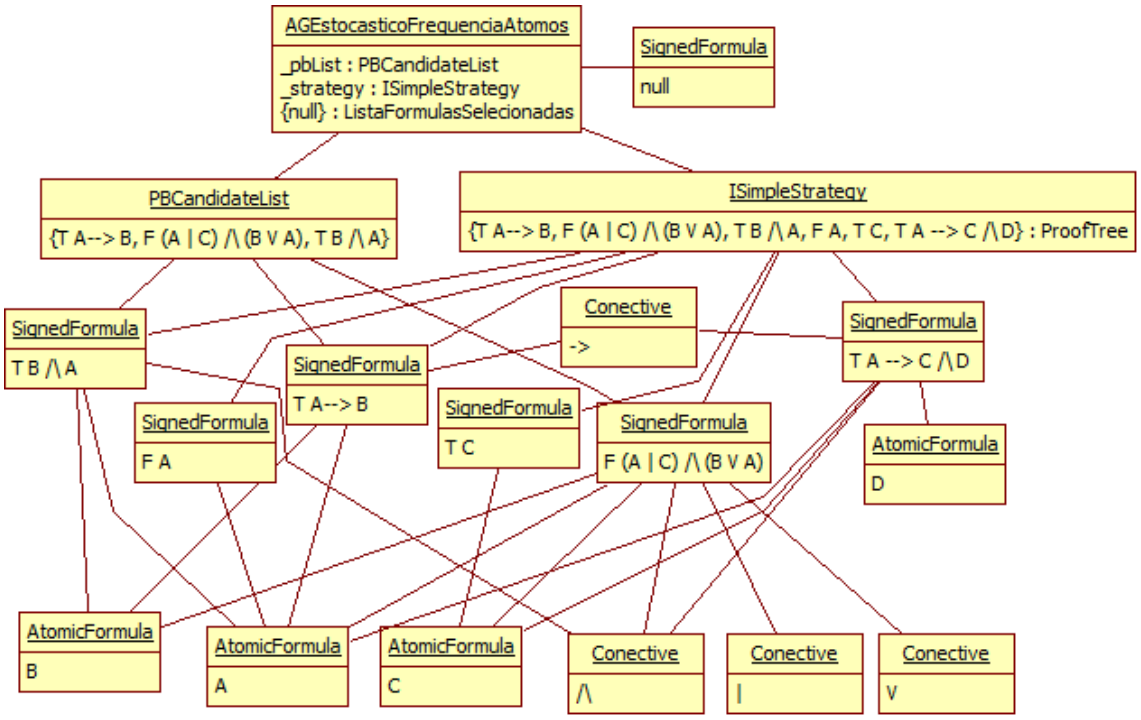


Figura 29 - Diagrama de Objetos AGEstocasticoFrequenciaAtomos
Fonte: Autoria própria.

6. Diagramas de Sequências

O diagrama de sequência da seleção elitista (Figura 30) representa a sequência de interação entre o *software* KEMS e as classes da abordagem AG elitista propostas. O *software* KEMS informa a estratégia (ISimpleStrategy), a lista de fórmulas PB candidatas (PBCandidateList) e a lista de fórmulas da árvore de prova (List<SignedFormula>). O primeiro método realiza a avaliação das fórmulas de acordo com a estratégia. Os métodos de avaliação de fórmulas adotados no presente trabalho estão descritos na seção 3.1.1. Posteriormente a seleção das fórmulas empregada é a elitista, descrita na subseção 3.1.5.

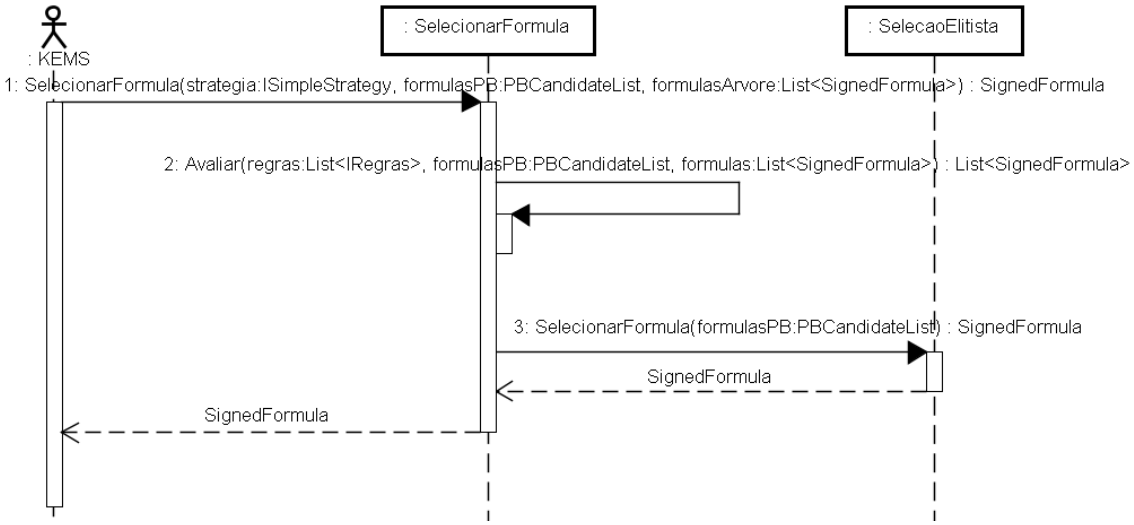


Figura 30 - Diagrama de Sequência da seleção elitista
Fonte: Autoria própria.

O diagrama de sequência da seleção estocástica (Figura 31) representa a sequência de interação entre o *software* KEMS e as classes da abordagem AG estocástica propostas. Os parâmetros informados pelo *software* KEMS são: a estratégia (ISimpleStrategy), a lista de fórmulas PB candidatas (PBCandidateList) e a lista de fórmulas da árvore de prova (List<SignedFormula>). O método Avaliar realiza a avaliação das fórmulas de acordo com a estratégia. Os métodos de avaliação de fórmulas adotados no presente trabalho estão descritos na seção 3.1.1. Posteriormente a

etapa de avaliação, a seleção das fórmulas é feita de forma estocástica, descrita na subseção 3.1.6.

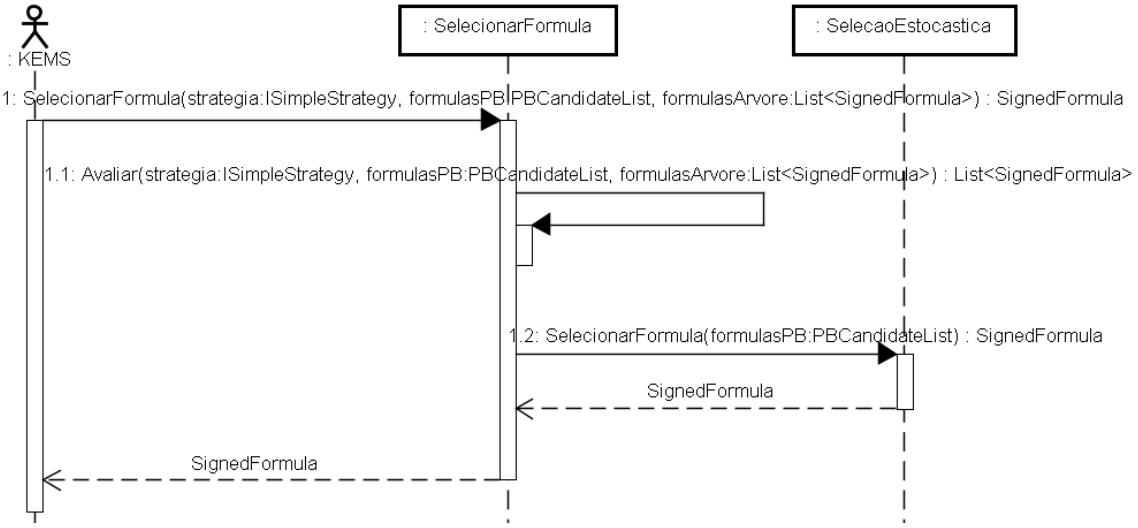


Figura 31 - Diagrama de Sequência da seleção estocástica
Fonte: Autoria própria.

A técnica de avaliação das fórmulas (seção 3.1.1) é definida pelo usuário através da interface gráfica do KEMS ou através da linha de comando. A estratégia de AG é informada deste parâmetro através do objeto de estratégia ISimpleStrategy.

7. WBS

A WBS (*Work Breakdown Structure*) do projeto está representada na Figura 32. A WBS é uma ferramenta de decomposição do trabalho do projeto em partes manejáveis. É de forma hierárquica (de mais geral para mais específica) orientada às entregas (*deliverables*) que precisam ser feitas para completar um projeto.

Fazem parte do presente projeto:

- Gerenciamento, composto pelo Planejamento, Acompanhamento e Entrega do Trabalho;
- Verificação dos Resultados;
- Avaliação de Fórmulas;
- Algoritmos, composto pela função de avaliação, abordagens elitista, estocástica e híbrida;
- Relatório de Testes do cálculo da complexidade de fórmulas, das abordagens estocástica, elitista e híbrida;

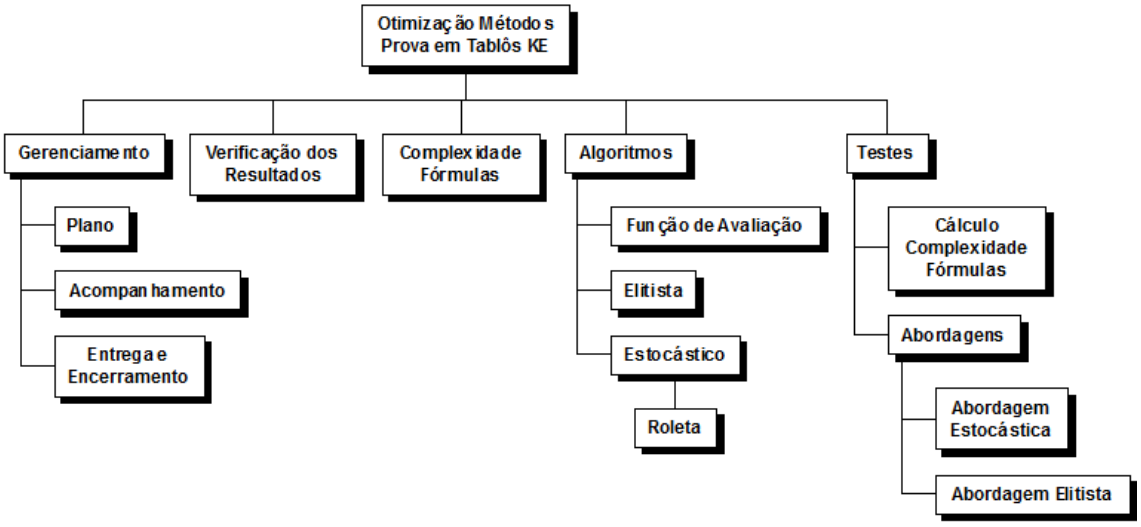


Figura 32 – WBS
Fonte: Autoria própria.

8. HOUSE OF QUALITY

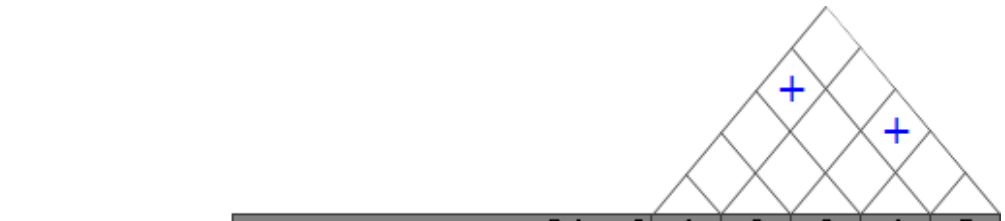
A Casa da Qualidade (HoQ – *House of Quality*) está ilustrada na Figura 33. Fazem parte dos requisitos os itens:

- Pacote de interface contendo o Algoritmo, com o objetivo de permitir a execução em diferentes provadores automatizados de teoremas;
- Portabilidade – funcionalidade sobre qualquer sistema operacional;
- Otimização do tempo de Prova;
- 10% Tempo de Prova;
- Facilidade de Alteração;
- Redução da curva de aprendizado por novos colaboradores;
- Redução do gasto dos recursos computacionais (memória, disco e CPU);

Os requisitos funcionais são:

- Encapsulamento: este item esta relacionado a interface do algoritmo;
- Desempenho;
- Documentação;
- Desenvolvimento em uma única linguagem. No caso a linguagem utilizada será o Java;
- *Design Patterns* (Padrões Projeto);

A relação entre os requisitos está descrito na Figura 33.



Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Quality Characteristics (a.k.a. "Functional Requirements" or "Hows") / Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	1 Encapsulamento	2 Desempenho	3 Documentação	4 Desenvolvimento em uma unica linguagem	5 Design Patterns (Padrões Projeto)
				Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)	▲	▲	▲	x	▲
1	9	12,5	5,0	Pacote de interface contendo o Algoritmo	Θ		▲	O	
2	9	10,0	4,0	Portabilidade	O			Θ	
3	9	7,5	3,0	Otimização do tempo de Prova		Θ			
4	9	10,0	4,0	Facilidade de Alteração			O	Θ	Θ
5	9	12,5	5,0	Redução da curva de aprendizado por novos colaboradores	▲		Θ	Θ	Θ
6	9	12,5	5,0	Redução do gasto dos recursos computacionais		Θ			
7	9	12,5	5,0	10% Tempo de Prova		Θ			O
				Target or Limit Value					
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)					
				Max Relationship Value in Column	9	9	9	9	9
				Weight / Importance	155,0	292,5	155,0	330,0	240,0
				Relative Weight	13,2	24,9	13,2	28,1	20,5

Figura 33 - House of Quality
Fonte: Autoria própria.

9. ANÁLISE DE RISCOS

O controle e avaliação de riscos são essenciais no desenvolvimento de qualquer projeto. É através do controle os riscos que a equipe de desenvolvimento consegue monitorar os possíveis pontos de maior probabilidade de fracasso do projeto. E, desta forma, elaborar uma forma de controlá-los de uma maneira aceitável, e se preparar para situações de riscos de falhas, falta de recursos e demais imprevistos que podem ocorrer durante a fase de desenvolvimento do projeto.

O levantamento de riscos do presente trabalho foi feito de acordo com Pressman (2006). Os riscos levantados, as respectivas probabilidades de ocorrência, o impacto e a estratégia adotada estão listados no Quadro 18. Estes riscos foram constantemente avaliados, principalmente os riscos com impactos catastróficos, pois estes em potencial podem inviabilizar o projeto todo.

Risco	Categoria	Probabilidade	Impacto	Estratégia	Tipo de estratégia
Consumo muito alto de memória em ambas as abordagens	Técnico	Alto	Crítico	Alteração das estruturas de dados que representam às fórmulas. Pré-seleção de regras mais aptas. Escolha de família mais simples.	Minimização
Falha das abordagens	Técnico	Médio	Catastrófico	Aplicação da Metodologia Proposta em Regras Beta Estudo e levantamento das causas do fracasso	Minimização
Estimativas de tempo irreais	Gestão	Baixa	Marginal	Foco na implementação de apenas uma abordagem	Contingência
Falha na implementação	Técnico	Baixa	Crítica	Estudo e aplicações teóricas	Minimização

Quadro 18 - Levantamento de Riscos
Fonte: Autoria própria.

9.1 Contenção de Riscos, Coordenação e Comunicação

As estratégias adotadas para controle dos riscos envolveram reuniões informais, devido ao tamanho da equipe, que é pequena. As tarefas e decisões do projeto foram compartilhadas via meio eletrônico, e-mail, e reuniões presenciais. O código desenvolvido foi compartilhado através do controle de versões do sistema Git.

APÊNDICE C – Diagrama de classes

O diagrama de classes completo da proposta de AG está representado na Figura 34.

Figura 34 – Diagrama de Classes AG completo
Fonte: Autoria própria.

APÊNDICE D – KEMS - Execução em lote

Para os testes em lote do *software* KEMS, o arquivo de configuração .seq deve ser informado ao .jar executável através do comando:

```
java -jar kems.jar filename.seq
```

O arquivo de configuração .seq está ilustrado na Figura 35. Os símbolos '#' representam comentários.

```
parser=sats5
saveOrigin=false
discardClosedBranches=false
saveDiscardedBranches=false
times=1
timeLimit=20
problems=
C:\...\PHP_02.prove
C:\...\PHP_03.prove
C:\...\PHP_04.prove

strategies=
ConfigurableSimpleStrategy
#MemorySaverStrategy
#SimpleStrategy
#BackjumpingSimpleStrategy
comparators=
#ReverseInsertionOrderComparator
InsertionOrderComparator
#OrComparator
#TrueComparator
modoag=
#entradas permitidas:
#ElitistaHibridoComplexidadeFreqAtomos,EstocasticoMaiorComplexidade,ElitistaFrequenciaAtomos,EstocasticoFrequenciaAtomos,ElitistaMaiorComplexidade,NotApplyAG - ver classe: main.newstrategy.simple.ag.util.AGConfiguration

ElitistaHibridoFreqAtomosComplexidade
#EstocasticoMaiorComplexidade
#ElitistaFrequenciaAtomos
#EstocasticoFrequenciaAtomos
#ElitistaMaiorComplexidade
#NotApplyAG
run
```

Figura 35 – Arquivo de entrada .seq
Fonte: Autoria própria.

O arquivo .jar executável do *software* KEMS de testes em lote deve possuir a classe CommandLineRunner do pacote proverinterface.commandline como classe *main*.

APÊNDICE E – RESULTADOS

1. Resultados família Gamma

Os resultados das abordagens propostas em AG para a família de problemas Gamma, instâncias de 1 a 10, estão representas nas tabelas: Tabela 1, Tabela 2, Tabela 3, Tabela 4, Tabela 5 e Tabela 6. O resultado da abordagem convencional está na Tabela 7.

Os critérios de avaliação de tempo, tamanho de prova, número de nós, número de bifurcações e uso de memória estão representados em forma gráfica nas figuras: Figura 1, Figura 2, Figura 3, Figura 4, Figura 5, respectivamente.

Tabela 1 – Gamma - AG Estocástico Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
gamma_01	16	78	26	11	0	0	128,4375
gamma_02	26	62	45	19	1	1	43,83072917
gamma_03	36	62	77	33	2	3	57,57291667
gamma_04	46	78	106	45	3	4	71,96614583
gamma_05	56	94	159	71	5	9	181,125
gamma_06	66	140	375	185	7	27	423,6197917
gamma_07	76	141	597	301	7	43	506,5442708
gamma_08	86	141	601	299	8	41	1456,578125
gamma_09	96	281	1871	981	13	141	4494,440104
gamma_10	106	234	1798	939	14	136	3759,747396

Fonte: Autoria própria.

Tabela 2 – Gamma - AG Estocástico Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
gamma_01	16	93	26	11	0	0	128,4166667
gamma_02	26	62	45	19	1	1	44,40625
gamma_03	36	78	69	29	3	3	53,453125
gamma_04	46	94	97	41	5	5	50,796875
gamma_05	56	94	134	57	6	6	105,09375
gamma_06	66	94	157	67	9	9	67,63802083
gamma_07	76	78	170	71	10	10	119,4010417
gamma_08	86	78	202	85	12	12	93,7109375
gamma_09	96	109	237	101	15	15	116,1510417
gamma_10	106	94	253	107	17	17	142,6640625

Fonte: Autoria própria.

Tabela 3 – Gamma - AG Elitista Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
gamma_01	16	78	26	11	0	0	127,9427083
gamma_02	26	62	45	19	1	1	44,05208333
gamma_03	36	63	77	33	2	3	65,20572917
gamma_04	46	94	131	59	3	7	129,1953125
gamma_05	56	109	229	109	4	15	276,8802083
gamma_06	66	124	415	207	5	31	582,0625
gamma_07	76	172	777	401	6	63	1908,723958
gamma_08	86	234	1491	787	7	127	4048,609375
gamma_09	96	375	2909	1557	8	255	8459,997396
gamma_10	106	670	5735	3095	9	511	17947,98177

Fonte: Autoria própria.

Tabela 4 – Gamma - AG Elitista Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
gamma_01	16	94	26	11	0	0	128,4348958
gamma_02	26	78	45	19	1	1	44,4296875
gamma_03	36	62	69	29	3	3	58,63541667
gamma_04	46	78	93	39	5	5	68,40885417
gamma_05	56	94	117	49	7	7	81,64322917
gamma_06	66	94	141	59	9	9	88,15104167
gamma_07	76	93	165	69	11	11	98,27604167
gamma_08	86	93	189	79	13	13	115,421875
gamma_09	96	94	213	89	15	15	115,4791667
gamma_10	106	109	237	99	17	17	124,5989583

Fonte: Autoria própria.

Tabela 5 – Gamma - AG Híbrido Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
gamma_01	16	90	26	11	0	0	128,4088542
gamma_02	26	60	45	19	1	1	44,67708333
gamma_03	36	80	77	33	2	3	53,98958333
gamma_04	46	120	94	39	4	4	139,0729167
gamma_05	56	120	206	97	6	14	189,8255208
gamma_06	66	100	273	129	6	17	212,0755208
gamma_07	76	130	399	195	7	29	724,15625
gamma_08	86	200	724	369	10	58	2134,744792
gamma_09	96	260	1437	753	11	119	5818,049479
gamma_10	106	600	4992	2677	16	418	9024,320313

Fonte: Autoria própria.

Tabela 6 – Gamma - AG Híbrido Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
gamma_01	16	90	26	11	0	0	128,4036458
gamma_02	26	80	45	19	1	1	43,48177083
gamma_03	36	80	69	29	3	3	61,1484375
gamma_04	46	90	93	39	5	5	68,3515625
gamma_05	56	90	117	49	7	7	81,58072917
gamma_06	66	100	141	59	9	9	87,96354167
gamma_07	76	100	165	69	11	11	99,31510417
gamma_08	86	90	189	79	13	13	115,375
gamma_09	96	100	213	89	15	15	115,421875
gamma_10	106	100	237	99	17	17	124,1015625

Fonte: Autoria própria.

Tabela 7 – Gamma - KEMS

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
gamma_01	16	63	26	11	0	0	128,4583333
gamma_02	26	63	45	19	1	1	43,13802083
gamma_03	36	78	77	33	2	3	64,8046875
gamma_04	46	78	131	59	3	7	128,8333333
gamma_05	56	94	229	109	4	15	277,0260417
gamma_06	66	94	415	207	5	31	581,4453125
gamma_07	76	125	777	401	6	63	1908,947917
gamma_08	86	188	1491	787	7	127	4047,434896
gamma_09	96	296	2909	1557	8	255	8462,789063
gamma_10	106	561	5735	3095	9	511	17943,55208

Fonte: Autoria própria.

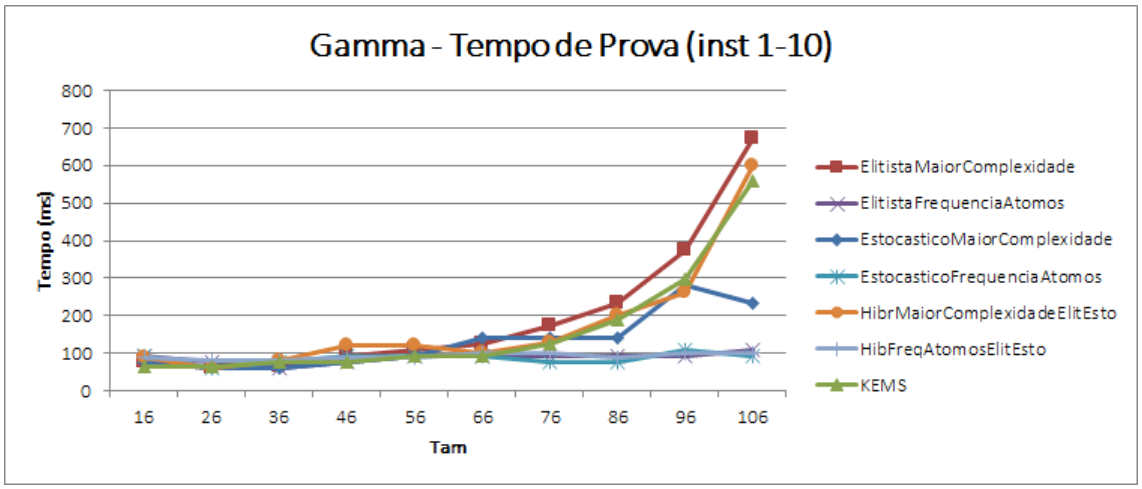


Figura 1 – Na comparação por tempo de prova para a família Gamma, as estratégias: análise de Frequência Átomos Estocástico, Elitista e Híbrida mostraram-se mais eficientes.
Fonte: Autoria própria.

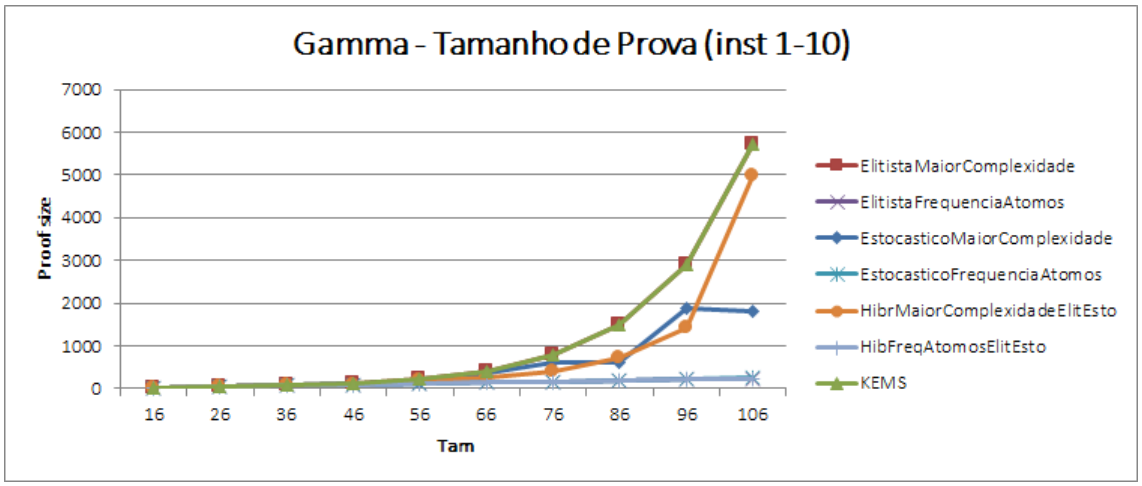


Figura 2 – Na comparação do tamanho da prova para a família Gamma, as estratégias: análise de Frequência Átomos Estocástico, Elitista e Híbrida mostraram-se mais eficientes.
Fonte: Autoria própria.

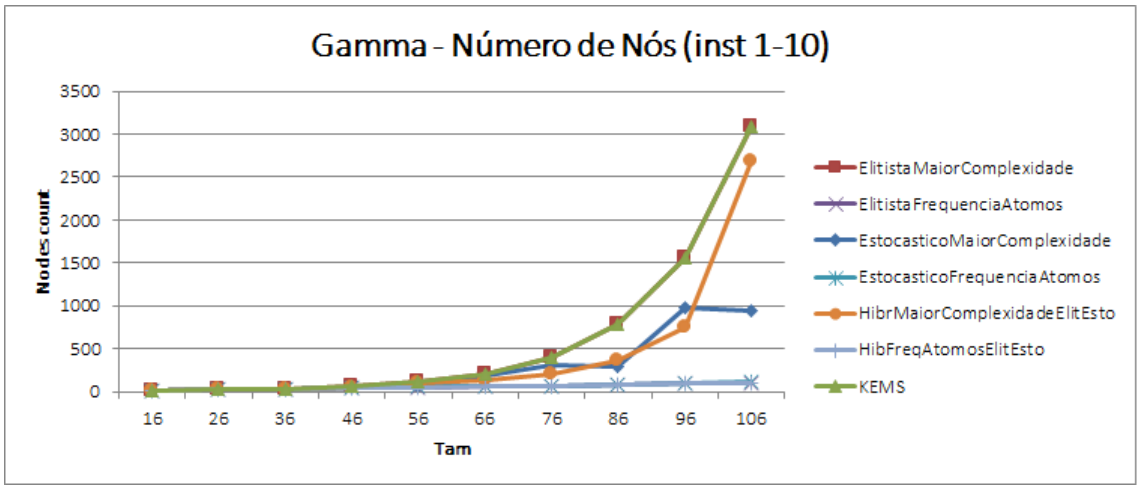


Figura 3 – Na comparação do número de nós para a família Gamma, as estratégias: análise de Frequência Átomos Estocástico, Elitista e Híbrida mostraram-se mais eficientes.
Fonte: Autoria própria.

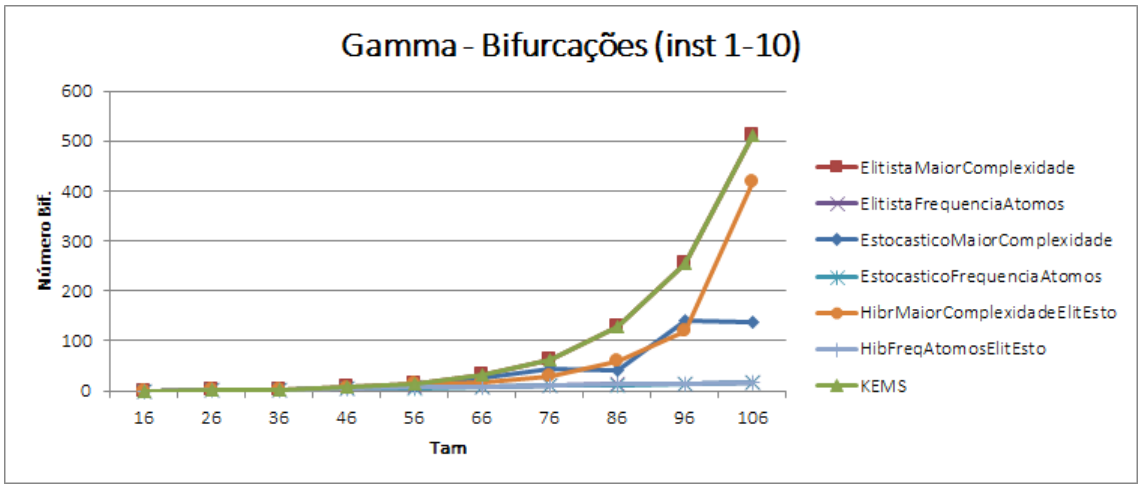


Figura 4 – Na comparação do número de bifurcações para a família Gamma, as estratégias: análise de Frequência Átomos Estocástico, Elitista e Híbrida mostraram-se mais eficientes.
Fonte: Autoria própria.

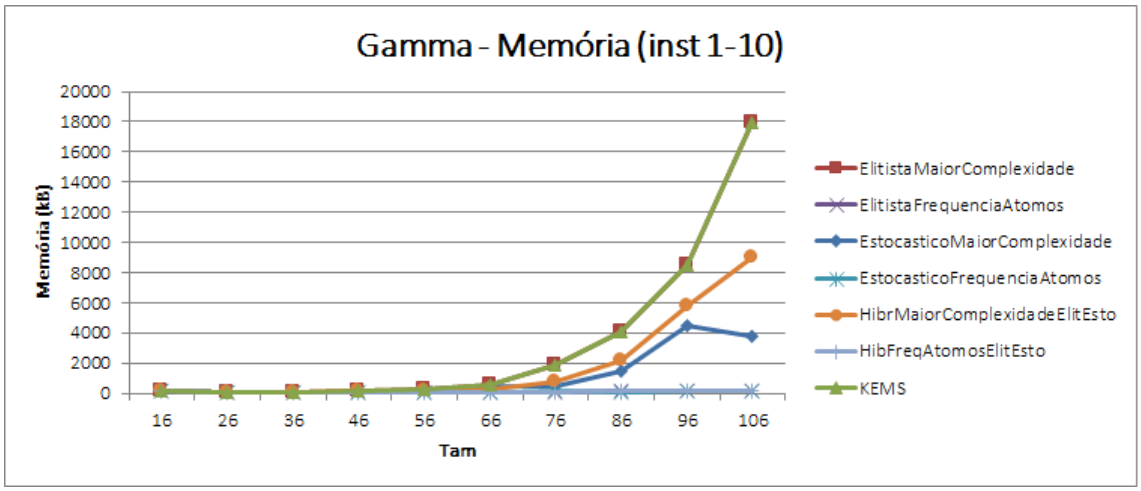


Figura 5 – Na comparação do uso de memória para a família Gamma, as estratégias: análise de Frequência Átomos Estocástico, Elitista e Híbrida mostraram-se mais eficientes.
Fonte: Autoria própria.

2. Resultados família Statman

Os resultados das abordagens propostas em AG para a família de problemas Statman, instâncias de 1 a 9, 13, 17, 21, 25 e 29, estão representas nas tabelas: Tabela 8, Tabela 9, Tabela 10, Tabela 11, Tabela 12 e Tabela 13. O resultado da abordagem convencional está na Tabela 14.

Os critérios de avaliação de tempo, tamanho de prova, número de nós, número de bifurcações e uso de memória estão representados em forma gráfica nas figuras: Figura 6, Figura 7, Figura 8, Figura 9 e Figura 10, respectivamente.

Tabela 8 – Statman - AG Estocástico Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
statman_1.prove	6	93	9	4	0	0	149,5781
statman_2.prove	17	62	40	12	1	1	48,54688
statman_3.prove	36	78	110	23	2	2	75,75521
statman_4.prove	63	93	500	103	5	12	167,7578
statman_5.prove	98	110	801	132	7	14	254,2005
statman_6.prove	141	109	916	144	9	17	1545,529
statman_7.prove	192	374	6570	778	11	85	1954,536
statman_8.prove	251	468	7759	820	13	90	4495,557
statman_9.prove	318	1123	19416	1785	15	204	12429,9
statman_13.prove	666	16911	145089	9716	23	1262	67587,53
statman_17.prove	1142	n/a	n/a	n/a	n/a	-	-
statman_21.prove	1746	n/a	n/a	n/a	n/a	-	-
statman_25.prove	2478	n/a	n/a	n/a	n/a	-	-
statman_29.prove	3338	n/a	n/a	n/a	n/a	-	-

Fonte: Autoria própria.

Tabela 9 – Statman - AG Estocástico Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
statman_1.prove	6	60	9	4	0	0	149,5495
statman_2.prove	17	60	40	12	1	1	49,19531
statman_3.prove	36	70	132	31	3	4	62,69271
statman_4.prove	63	90	312	58	5	9	231,0469
statman_5.prove	98	140	847	142	7	20	531,3802
statman_6.prove	141	180	2020	321	9	40	797,6484
statman_7.prove	192	190	1980	270	11	35	3609,716
statman_8.prove	251	350	4315	533	13	63	7911,943
statman_9.prove	318	690	8771	945	14	113	11064,98
statman_13.prove	666	10274	85731	6211	22	929	76916,39
statman_17.prove	1142	n/a	n/a	n/a	n/a	-	-
statman_21.prove	1746	n/a	n/a	n/a	n/a	-	-
statman_25.prove	2478	n/a	n/a	n/a	n/a	-	-
statman_29.prove	3338	n/a	n/a	n/a	n/a	-	-

Fonte: Autoria própria.

Tabela 10 – Statman - AG Elitista Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
statman_1.prove	6	47	9	4	0	0	149,0755
statman_2.prove	17	63	40	12	1	1	46,11458
statman_3.prove	36	62	110	23	2	2	62,67188
statman_4.prove	63	78	264	52	4	5	133,1146
statman_5.prove	98	78	449	73	6	7	137,5729
statman_6.prove	141	94	866	134	8	13	987,0182
statman_7.prove	192	110	1264	173	10	16	1577,615
statman_8.prove	251	171	2297	296	12	28	3444,964
statman_9.prove	318	219	3142	369	14	34	5837,747
statman_13.prove	666	1747	16862	1469	22	142	13756,52
statman_17.prove	1142	16474	85426	5641	30	574	54006,27
statman_21.prove	1746	128626	422022	21957	38	2302	264091,6
statman_25.prove	2478						
statman_29.prove	3338						

Fonte: Autoria própria.

Tabela 11 – Statman - AG Elitista Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
statman_1.prove	6	78	9	4	0	0	148,5625
statman_2.prove	17	62	40	12	1	1	49,15625
statman_3.prove	36	78	110	23	2	2	62,71354
statman_4.prove	63	94	235	42	4	4	101,4792
statman_5.prove	98	94	416	65	6	6	131,3776
statman_6.prove	141	109	661	92	8	8	814,5755
statman_7.prove	192	140	1027	131	10	11	1551,383
statman_8.prove	251	156	1432	166	12	13	231,8307
statman_9.prove	318	187	1925	205	14	15	2884,797
statman_13.prove	666	593	5026	409	22	24	7828,565
statman_17.prove	1142	1809	10819	711	30	37	4819,435
statman_21.prove	1746	4727	19371	1069	38	49	3856,703
statman_25.prove	2478	11840	32960	1553	46	68	4420,711
statman_29.prove	3338	24117	49024	2039	54	80	5450,917

Fonte: Autoria própria.

Tabela 12 – Statman - AG Híbrido Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
statman_1.prove	6	90	9	4	0	0	149,0677
statman_2.prove	17	70	40	12	1	1	49,46615
statman_3.prove	36	70	110	23	2	2	62,67188
statman_4.prove	63	90	264	52	4	5	130,4089
statman_5.prove	98	100	449	73	6	7	137,9271
statman_6.prove	141	120	866	134	8	13	986,7708
statman_7.prove	192	160	1264	173	10	16	1577,654
statman_8.prove	251	220	2297	296	12	28	3444,109
statman_9.prove	318	310	3142	369	14	34	5838,24
statman_13.prove	666	2020	16862	1469	22	142	13761,69
statman_17.prove	1142	16536	85426	5641	30	574	54014,34
statman_21.prove	1746	130854	422022	21957	38	2302	263856,9
statman_25.prove	2478	n/a	n/a	n/a	n/a	-	-
statman_29.prove	3338	n/a	n/a	n/a	n/a	-	-

Fonte: Autoria própria.

Tabela 13 – Statman - AG Híbrido Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
statman_1.prove	6	90	9	4	0	0	148,5833
statman_2.prove	17	70	40	12	1	1	50,31771
statman_3.prove	36	90	110	23	2	2	62,71354
statman_4.prove	63	122	235	42	4	4	101,4583
statman_5.prove	98	100	416	65	6	6	132,1953
statman_6.prove	141	120	661	92	8	8	814,2552
statman_7.prove	192	140	978	123	10	10	1518,201
statman_8.prove	251	180	1432	166	12	13	2896,674
statman_9.prove	318	220	1925	205	14	15	253,112
statman_13.prove	666	750	5107	417	22	25	10112,43
statman_17.prove	1142	2240	10819	711	30	37	2546,005
statman_21.prove	1746	5960	19371	1069	38	49	3436,63
statman_25.prove	2478	14823	32815	1545	46	67	2650,323
statman_29.prove	3338	29420	48253	2013	54	77	6499,052

Fonte: Autoria própria.

Tabela 14 – Statman - KEMS

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
statman_1.prove	6	62	9	4	0	0	149,8906
statman_2.prove	17	78	62	20	2	3	78,92188
statman_3.prove	36	78	285	61	3	7	159,0651
statman_4.prove	63	110	928	154	5	19	464,349
statman_5.prove	98	187	2547	339	7	43	1801,25
statman_6.prove	141	375	6462	708	9	91	6626,682
statman_7.prove	192	858	15665	1445	11	187	14933,57
statman_8.prove	251	1997	36844	2918	13	379	14200,36
statman_9.prove	318	5553	84783	5863	15	763	32592,94
statman_13.prove	666	n/a	n/a	n/a	n/a	-	-
statman_17.prove	1142	n/a	n/a	n/a	n/a	-	-
statman_21.prove	1746	n/a	n/a	n/a	n/a	-	-
statman_25.prove	2478	n/a	n/a	n/a	n/a	-	-
statman_29.prove	3338	n/a	n/a	n/a	n/a	-	-

Fonte: Autoria própria.

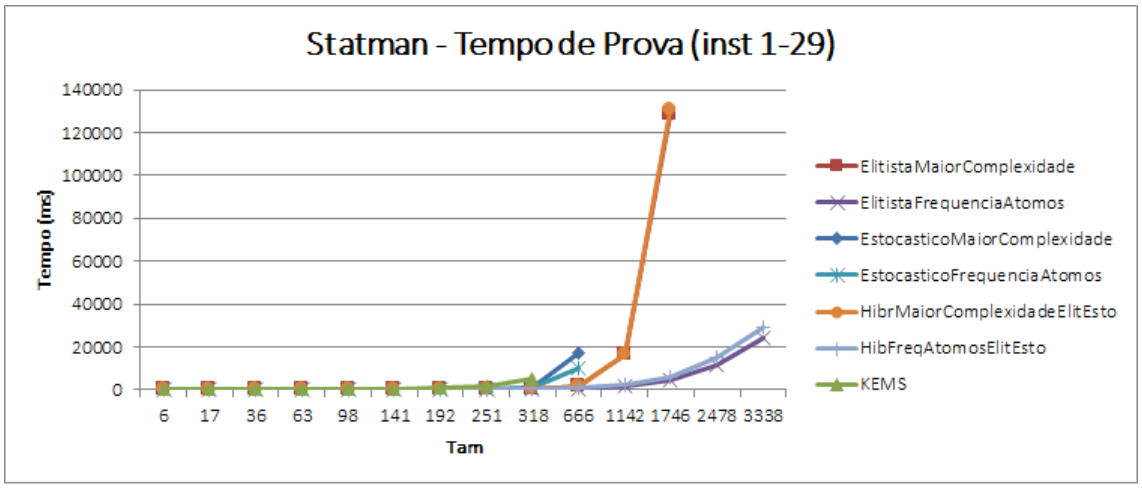


Figura 6 – Na comparação do tempo da prova para a família Statman, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

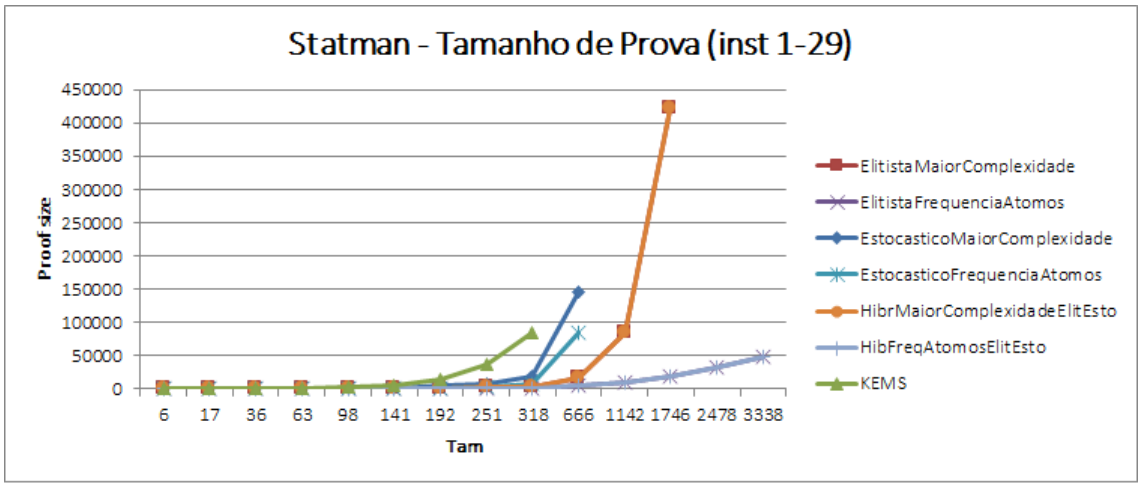


Figura 7 – Na comparação do tamanho da prova para a família Statman, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

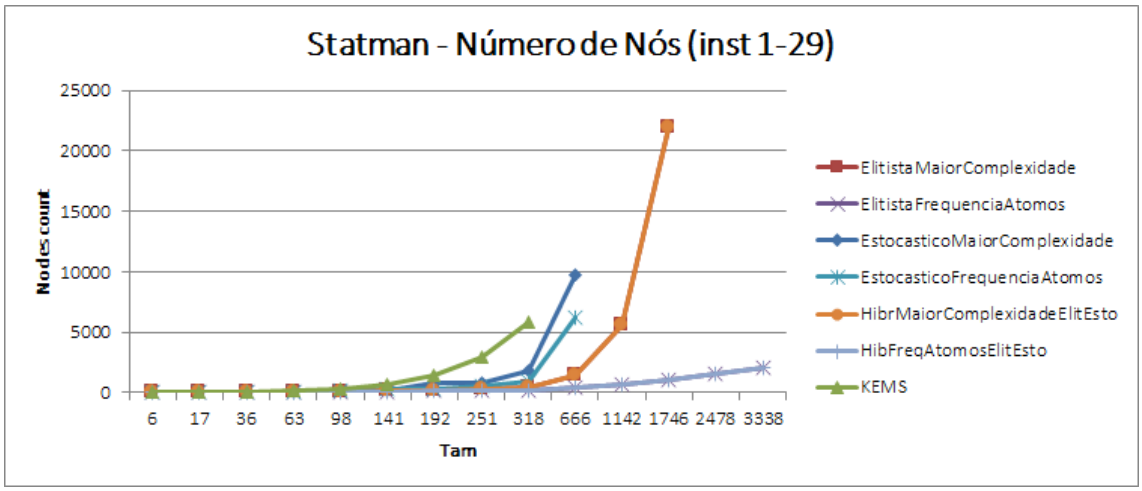


Figura 8 – Na comparação do número de nós para a família Statman, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

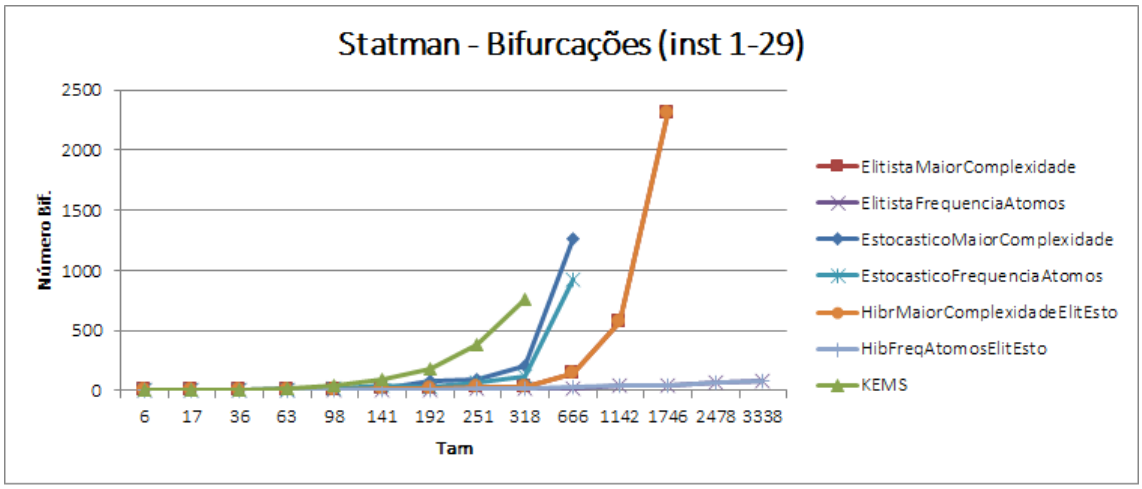


Figura 9 – Na comparação do número de bifurcações para a família Statman, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

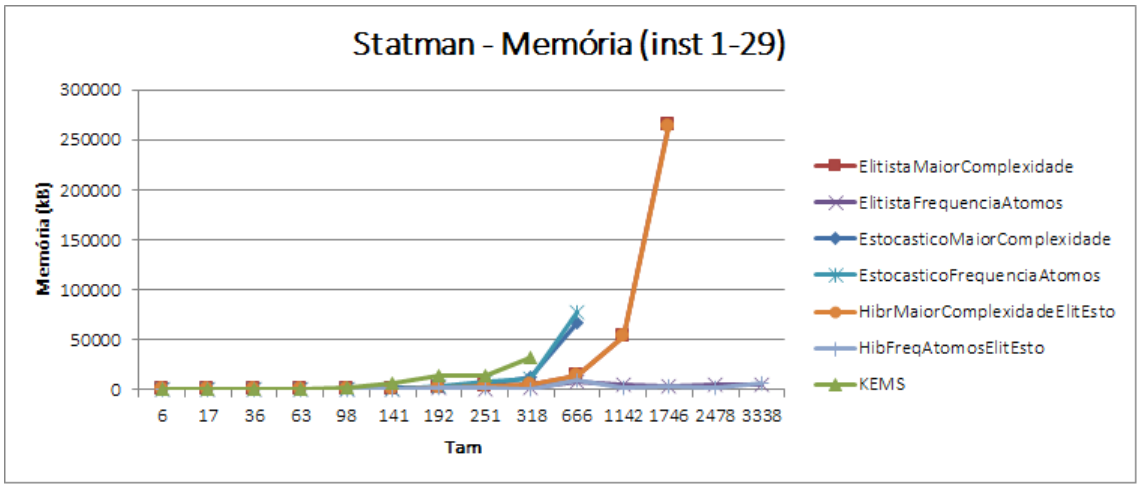


Figura 10 – Na comparação do uso de memória para a família Statman, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

3. Resultados família H

Os resultados das abordagens propostas em AG para a família de problemas H, instâncias de 1 a 7, estão representas nas tabelas: Tabela 15, Tabela 16, Tabela 17, Tabela 18, Tabela 19 e Tabela 20. O resultado da abordagem convencional está na Tabela 21.

Os critérios de avaliação de tempo, tamanho de prova, número de nós, número de bifurcações e uso de memória estão representados em forma gráfica nas figuras: Figura 11, Figura 12, Figura 13, Figura 14 e Figura 15.

Tabela 15 – H - AG Estocástico Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
h_1.prove	4	78	11	6	0	0	145,7865
h_2.prove	19	78	78	19	1	1	67,61198
h_3.prove	59	94	409	59	5	5	235,2188
h_4.prove	159	297	1930	181	14	15	4186,125
h_5.prove	399	2886	9438	621	30	63	8170,849
h_6.prove	959	44447	40951	1584	62	105	21313,57
h_7.prove	2239	348115	184924	4947	125	324	152147,7

Fonte: Autoria própria.

Tabela 16 – H - AG Estocástico Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
h_1.prove	4	93	11	6	0	0	139,3151
h_2.prove	19	63	78	19	1	1	57,1224
h_3.prove	59	109	424	66	6	6	239,7135
h_4.prove	159	390	1936	178	13	13	4264,372
h_5.prove	399	3120	8752	489	29	37	9422,461
h_6.prove	959	49077	39789	1460	61	91	18969,54
h_7.prove	2239	421786	186843	5306	126	372	24414,41

Fonte: Autoria própria.

Tabela 17 – H - AG Elitista Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
h_1.prove	4	47	11	6	0	0	142,5911
h_2.prove	19	62	78	19	1	1	55,47656
h_3.prove	59	94	431	68	5	5	225,9635
h_4.prove	159	265	2048	203	13	13	4140,042
h_5.prove	399	2434	9361	579	29	29	6489,724
h_6.prove	959	40155	42328	1666	61	61	12591,16
h_7.prove	2239	332545	190386	4996	125	125	96799,33

Fonte: Autoria própria.

Tabela 18 – H - AG Elitista Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
h_1.prove	4	78	11	6	0	0	139,2474
h_2.prove	19	63	80	20	2	2	65,91146
h_3.prove	59	125	422	65	6	6	232,5859
h_4.prove	159	344	1971	188	14	14	3660,518
h_5.prove	399	3697	8761	500	30	30	4615,513
h_6.prove	959	65676	41035	1522	62	62	14770,62
h_7.prove	2239	972116	182667	4374	125	125	27907,1

Fonte: Autoria própria.

Tabela 19 – H - AG Híbrido Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
h_1.prove	4	94	11	6	0	0	139,1797
h_2.prove	19	78	78	19	1	1	53,03125
h_3.prove	59	94	431	68	5	5	227,0391
h_4.prove	159	297	2083	207	13	13	4141,612
h_5.prove	399	2559	9589	607	29	29	5002,083
h_6.prove	959	41776	43431	1771	61	61	13953,35
h_7.prove	2239	340877	195628	5412	125	125	72176,81

Fonte: Autoria própria.

Tabela 20 – H - AG Híbrido Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
h_1.prove	4	78	11	6	0	0	139,2266
h_2.prove	19	78	80	19	1	1	58,71615
h_3.prove	59	156	414	62	5	5	442,6432
h_4.prove	159	577	1921	186	13	13	5594,724
h_5.prove	399	6661	6661	501	29	29	2801,115
h_6.prove	959	134394	39737	1409	61	61	14144,77
h_7.prove	2239	n/a	n/a	n/a	n/a	-	-

Fonte: Autoria própria.

Tabela 21 – H - KEMS

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
h_1.prove	4	63	11	6	0	0	139,2083
h_2.prove	19	62	80	20	2	2	64,78646
h_3.prove	59	94	427	64	6	8	275,9479
h_4.prove	159	296	2115	212	14	30	4853,26
h_5.prove	399	3042	10247	748	30	116	14633,18
h_6.prove	959	57783	48811	2784	62	458	65370,75
h_7.prove	2239	n/a	n/a	n/a	n/a	-	-

Fonte: Autoria própria.

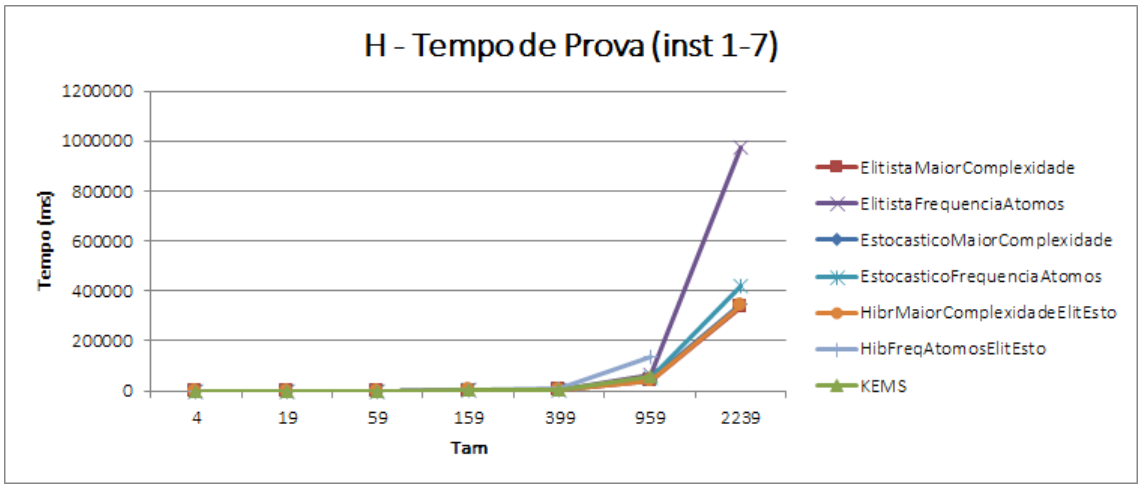


Figura 11 – Na comparação do tempo de prova para a família H, todas as estratégias (exceto a abordagem híbrida com função de avaliação análise de frequência de átomos) mostraram-se mais eficientes.

Fonte: Autoria própria.

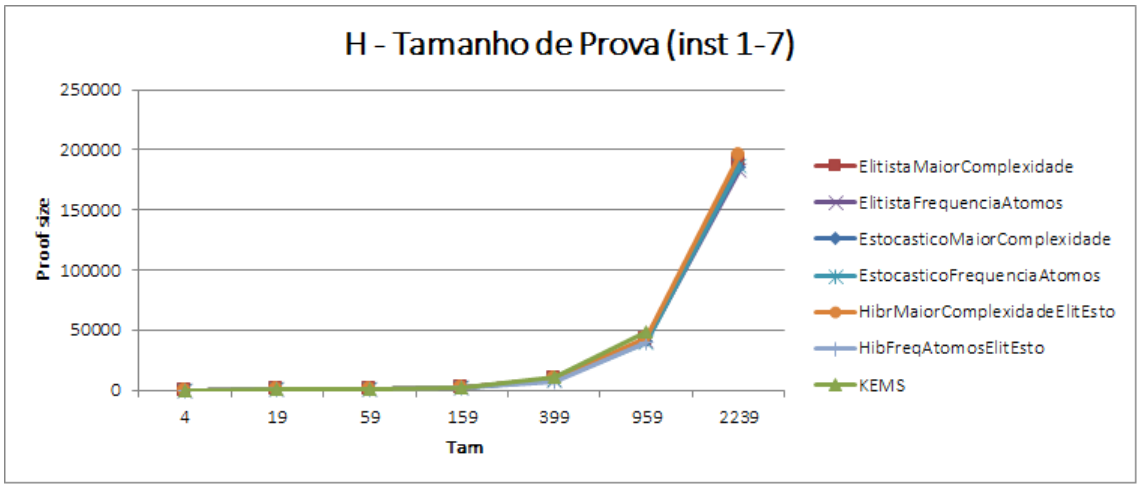


Figura 12 – Na comparação do tamanho da prova para a família H, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

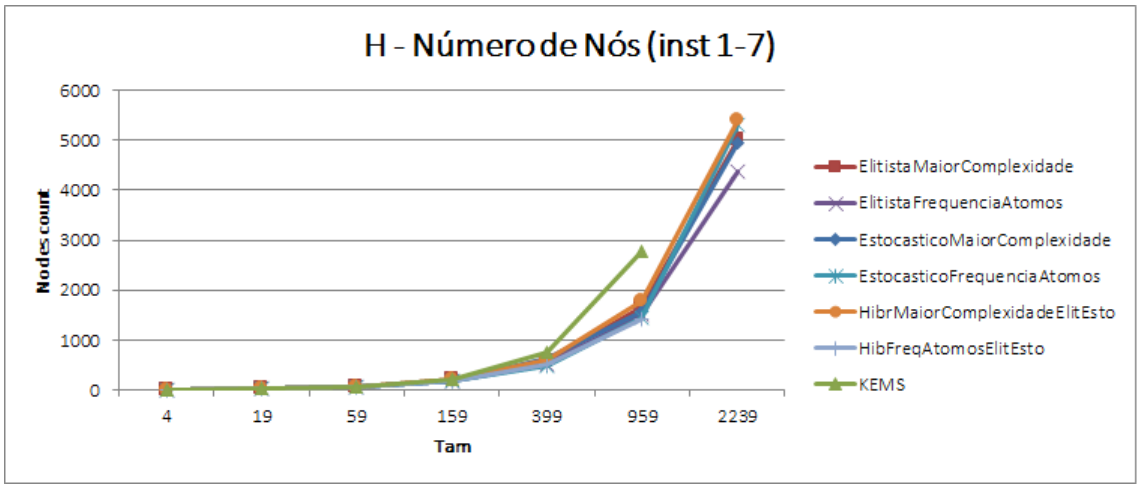


Figura 13 – Na comparação do número de nós para a família H, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

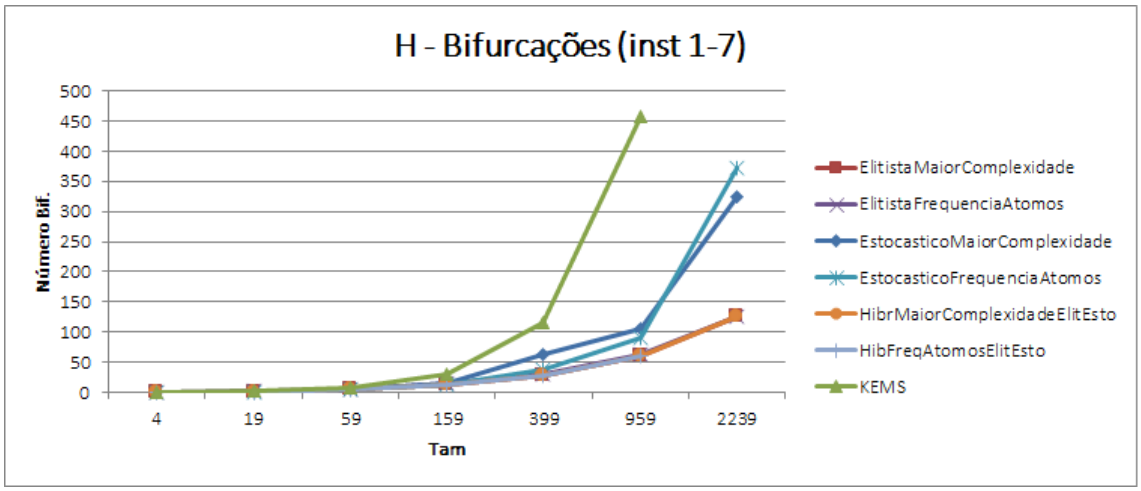


Figura 14 – Na comparação do número de bifurcações para a família H, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

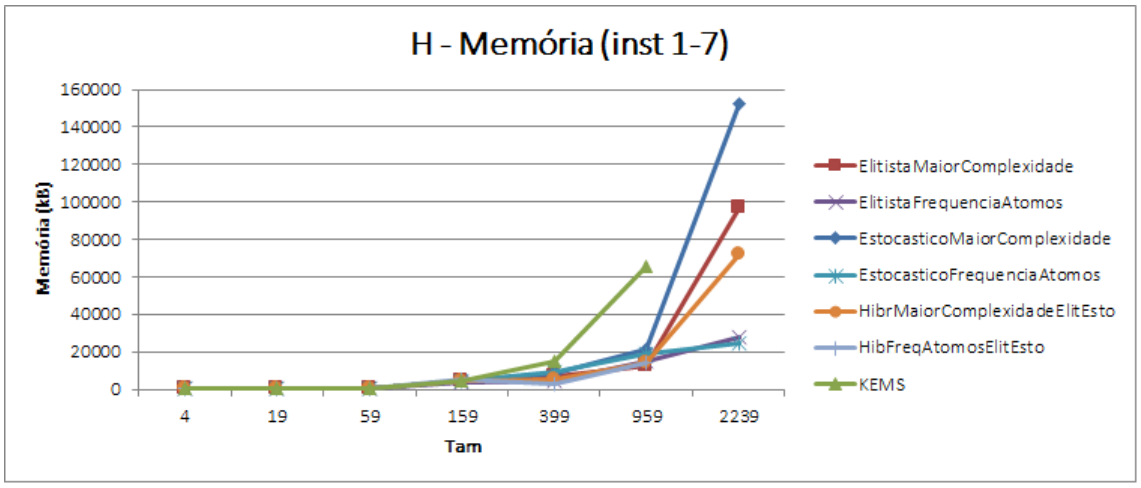


Figura 15 – Na comparação do uso de memória para a família H, todas as estratégias (exceto a abordagem híbrida com função de avaliação análise de frequência de átomos) mostraram-se mais eficientes.
Fonte: Autoria própria.

4. Resultados família PHP

Os resultados das abordagens propostas em AG para a família de problemas PHP, instâncias de 1 a 6, estão representas nas tabelas: Tabela 22, Tabela 23, Tabela 24, Tabela 25, Tabela 26 e Tabela 27. O resultado da abordagem convencional está na Tabela 28.

Os critérios de avaliação de tempo, tamanho de prova, número de nós, número de bifurcações e uso de memória estão representados em forma gráfica nas figuras: Figura 16, Figura 17, Figura 18, Figura 19 e Figura 20.

Tabela 22 – PHP - AG Estocástico Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
PHP_01.prove	5	94	9	6	0	0	142,3281
PHP_02.prove	27	62	45	25	1	1	65,14583
PHP_03.prove	74	156	283	154	6	12	431,7813
PHP_04.prove	155	468	2230	1037	13	100	7644,19
PHP_05.prove	279	4587	22296	8984	21	763	94279,44
PHP_06.prove	455	58329	239977	78408	34	5856	1164386

Fonte: Autoria própria.

Tabela 23 – PHP - AG Estocástico Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
PHP_01.prove	5	62	9	6	0	0	131,3828
PHP_02.prove	27	46	45	25	1	1	72,42188
PHP_03.prove	74	140	263	138	5	10	458,5729
PHP_04.prove	155	499	2265	1054	13	94	8101,479
PHP_05.prove	279	2621	15954	7184	23	611	93092,41
PHP_06.prove	455	67923	225427	78448	31	6040	1195758

Fonte: Autoria própria.

Tabela 24 – PHP - AG Elitista Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
PHP_01.prove	5	94	9	6	0	0	149,0521
PHP_02.prove	27	62	45	25	1	1	71,78646
PHP_03.prove	74	156	224	134	5	9	435,7292
PHP_04.prove	155	374	1274	832	11	61	4912,339
PHP_05.prove	279	2215	9066	6178	19	457	62606,13
PHP_06.prove	455	41792	78221	54237	29	3921	704967,3

Fonte: Autoria própria.

Tabela 25 – PHP - AG Elitista Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
PHP_01.prove	5	63	9	6	0	0	122,4036
PHP_02.prove	27	62	45	25	1	1	72,41406
PHP_03.prove	74	156	241	129	5	9	439,2396
PHP_04.prove	155	484	1525	725	10	53	4700,906
PHP_05.prove	279	2886	9702	4823	16	354	58153,26
PHP_06.prove	455	83569	88014	43356	27	3091	543515,7

Fonte: Autoria própria.

Tabela 26 – PHP - AG Híbrido Maior Complexidade

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
PHP_01.prove	5	78	9	6	0	0	122,5365
PHP_02.prove	27	93	45	25	1	1	73,14583
PHP_03.prove	74	140	226	136	5	9	436,625
PHP_04.prove	155	468	1356	908	11	67	5997,659
PHP_05.prove	279	2122	10766	7758	22	577	85442,88
PHP_06.prove	455	55365	106893	80923	34	5907	1046019

Fonte: Autoria própria.

Tabela 27 – PHP - AG Híbrido Freq. átomos

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
PHP_01.prove	5	78	9	6	0	0	122,5417
PHP_02.prove	27	78	45	25	1	1	73,98698
PHP_03.prove	74	156	200	106	4	7	355,0365
PHP_04.prove	155	452	1165	568	8	40	5514,115
PHP_05.prove	279	3182	8724	4038	15	289	53698,46
PHP_06.prove	455	171422	99027	49704	29	3548	644806,6

Fonte: Autoria própria.

Tabela 28 – PHP - KEMS

Problem	Size	Time (ms)	Proof size	Nodes count	Proof tree height	Number Bif.	Memory (kB)
PHP_01.prove	5	94	9	6	0	0	149,0313
PHP_02.prove	27	78	45	25	1	1	71,16406
PHP_03.prove	74	141	206	108	4	7	356,2943
PHP_04.prove	155	453	1728	900	9	93	8521,695
PHP_05.prove	279	7347	27602	11989	16	1160	142046,8
PHP_06.prove	455	n/a	n/a	n/a	n/a	-	-

Fonte: Autoria própria.

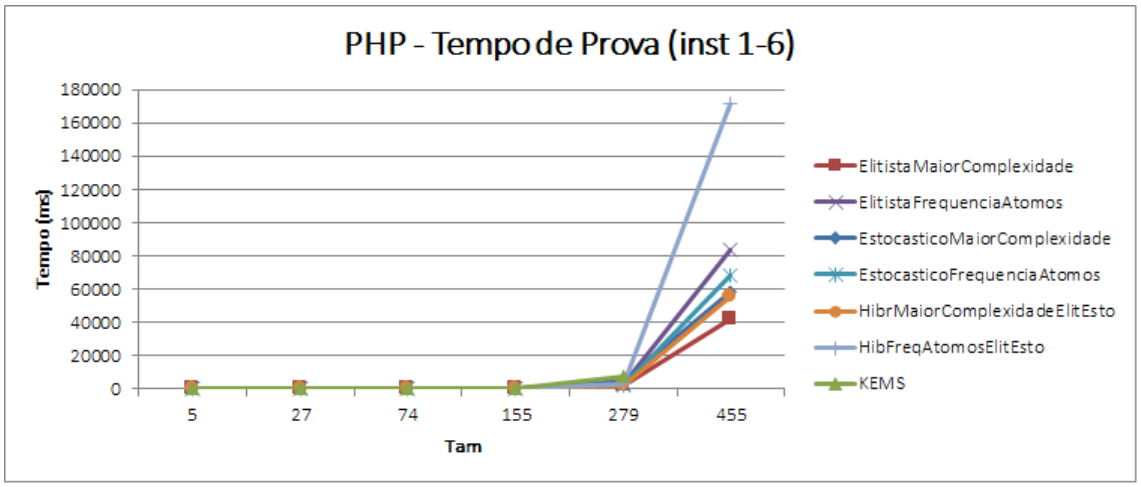


Figura 16 – Na comparação do tempo de prova para a família PHP, todas as estratégias (exceto a abordagem híbrida com função de avaliação análise de frequência de átomos) mostraram-se mais eficientes.

Fonte: Autoria própria.

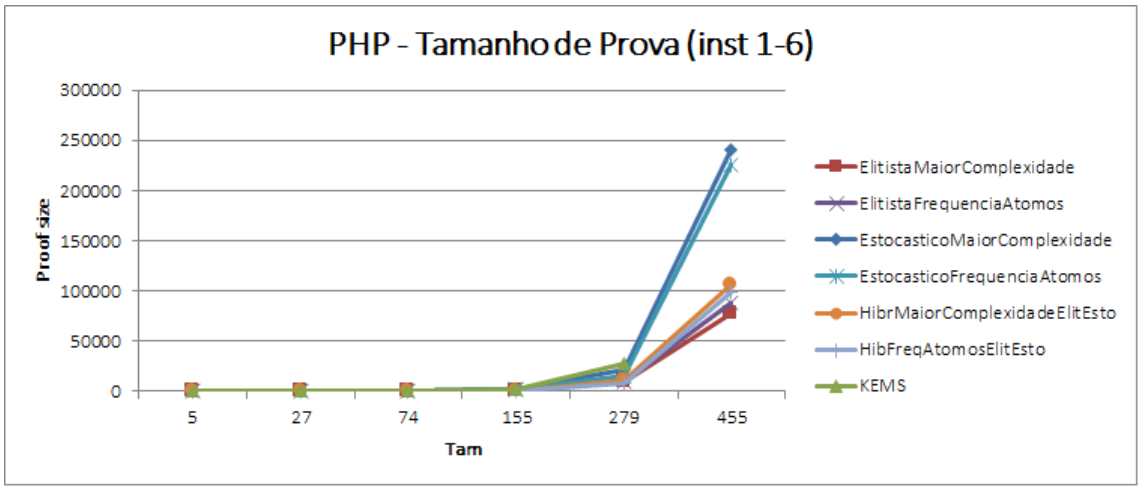


Figura 17 – Na comparação do tamanho da prova para a família PHP, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

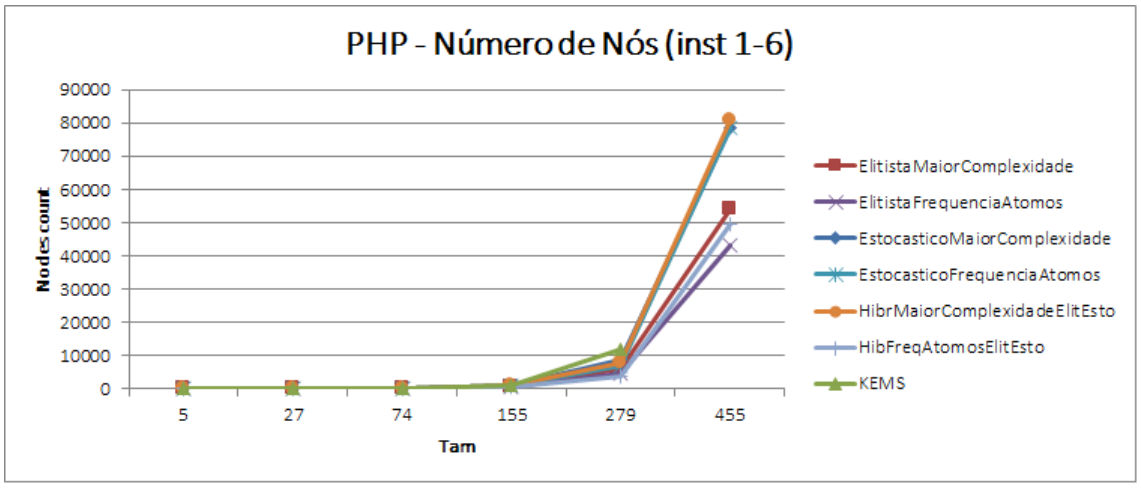


Figura 18 – Na comparação do número de nós para a família PHP, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

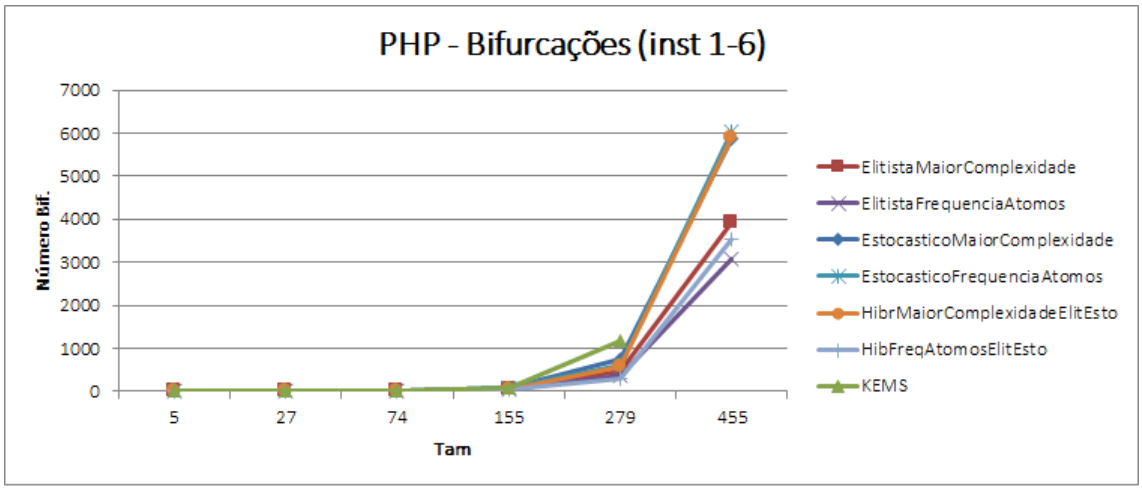


Figura 19 – Na comparação do número de bifurcações para a família PHP, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.

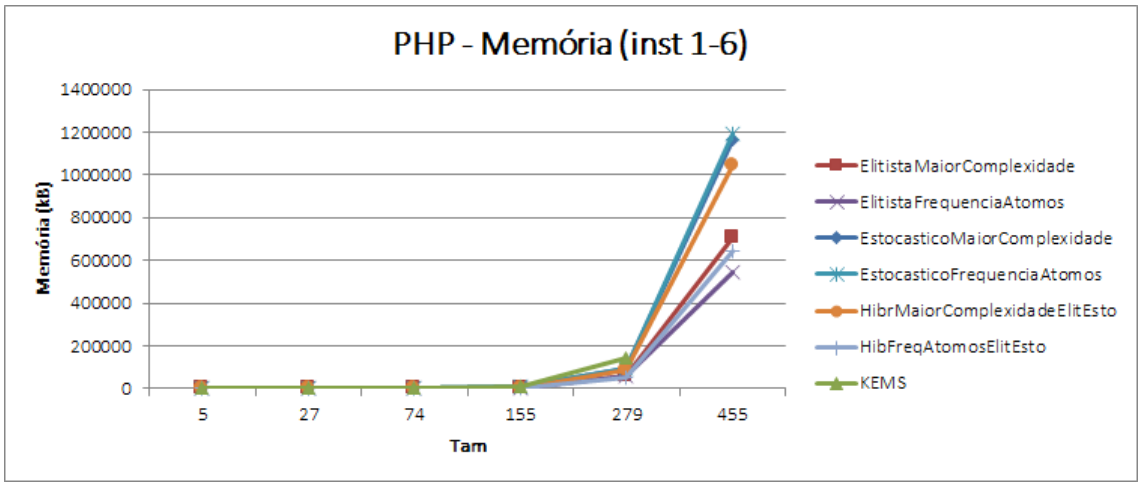


Figura 20 – Na comparação do uso de memória para a família PHP, todas as estratégias mostraram-se mais eficientes.
Fonte: Autoria própria.