EDGAR LEONARDO ROMERO TOBAR

Contribuições à Verificação Funcional Ajustada Por Cobertura Para Núcleos de Hardware de Comunicação e Multimídia

São Paulo
2010

EDGAR LEONARDO ROMERO TOBAR

Contribuições à Verificação Funcional Ajustada Por Cobertura Para Núcleos de Hardware de Comunicação e Multimídia

Tese apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do Título de Doutor em Engenharia Elétrica.

Área de concentração:
Microeletrônica

Orientador:
Prof. Dr. Wang Jiang Chau

São Paulo
2010

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, de julho de 2010.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Tobar, Edgar Leonardo Romero
Contribuições à verificação funcional ajustada por cobertura para núcleos de hardware de comunicação e multimídia / E.L.R. Tobar. -- ed.rev. -- São Paulo, 2010.
130 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1. Microeletrônica 2. Circuitos integrados 3. Qualidade do projeto I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II. t.

DEDICATÓRIA

A Dios fuente de
todas las bendiciones

A
Edgar, Elizabeth,
Diana e Paola.

Por acompañarme
en cada paso,
con determinación
paciencia y
mucho amor.

AGRADECIMENTOS

Agradeço ao Professor Wang Jiang Chau pela amizade, apoio, guia e dedicação, dados durante todas as etapas do desenvolvimento deste trabalho. Adicionalmente, agradeço a oportunidade que me foi dada de vir ao Brasil e obter uma experiência de vida muito frutífera em todos os aspectos pessoais.

Ao Professor Marius Strum pela amizade e o acompanhamento sempre brindados.

À Profa. Edna Barros, Prof. Elmar Melcher, Profa. Karina Rocha, Prof. Cristiano Araujo e Bruno Prado, assim como ao pessoal do Projeto Brazil-IP, na Universidade Federal de Campina Grande e na Universidade Federal de Pernambuco, pelos núcleos de hardware usados no presente trabalho.

Aos colegas e amigos do laboratório, Sebastian, Carlos, Jonas, Raul, Mario, Joel, Kleber e Alex. A sua participação e interação nesses anos foi de grande ajuda.

A Sebastian Eslava pela companhia, amizade e pelos gratos momentos compartilhados nesses anos de pesquisa.

A meus pais, Edgar e Elizabeth, por fornecerem recursos de todo tipo para que eu e minha irmã consigamos ser, cada dia, melhores e mais felizes. Devemos tudo a vocês. A Diana Elizabeth, pelo amor e preocupação, e por ser sempre um exemplo de tenacidade e disciplina.

A Paola, pelo amor, amizade, apoio e inspiração. Agradeço também por todas as lições que temos aprendido juntos.

Ao senhor Joaquim Galindo, à senhora Myriam Rozo, à Laura e ao Fernando, por torcerem e apoiarem o bom sucesso deste trabalho.

Às minhas tias, tios, primos que, da Colômbia e Estados Unidos, enviaram bênçãos e força necessárias para eu sempre manter a motivação. À Paulina, pelo apoio e pelas bênçãos.

Aos amigos Ana, Reinaldo, Renata, Gabriel, Erica, Marco, Liege, Luis e Diego.

Ao CNPq, por fornecer os recursos para o desenvolvimento deste trabalho, e à FAPESP, pelos equipamentos utilizados na realização dos experimentos.

RESUMO

Tornar a verificação funcional mais eficiente, em termos de gasto de recursos de computação e tempo, é necessário para a contínua evolução dos sistemas digitais. A verificação funcional com geração de casos de teste aleatória ajustada por cobertura é uma das alternativas identificadas nos últimos anos para acelerar a execução de *testbenches*. Várias abordagens têm sido testadas com sucesso na verificação funcional de núcleos de *hardware*, no domínio de aplicação dos processadores de propósito geral, porém, influenciada por características específicas do domínio, dos modelos de cobertura e do espaço possível de casos de teste. Por outro lado, pouca atenção tem sido dispensada à verificação ajustada por cobertura em outros domínios de aplicação como nos de sistemas de comunicação e de sistemas multimídia. Estes casos são tratados no presente estudo, com os fatores específicos que influenciam os resultados dos *testbenches* com geração ajustada. Entre os fatores relevantes para isto, foram identificados o tamanho do espaço de casos de teste e a distribuição da ocorrência dos eventos de cobertura, sendo necessária para o desenvolvimento do presente trabalho, a realização de várias alterações na construção de *testbenches* com ajuste.

A geração de casos de teste ajustada por cobertura é realizada a partir da realimentação da informação do estado da cobertura, para se determinar os casos de teste necessários para tornar o progresso da cobertura mais rápido. Esta realimentação depende da criação, por aprendizado automático, de modelos que relacionem os casos de teste com as ocorrências dos eventos de cobertura. Com núcleos de hardware realistas e de grande porte, neste trabalho, foram aplicadas as técnicas de aprendizado de redes Bayesianas e *data mining* com árvores de classificação, já utilizados em outras pesquisas mais específicas. Estas técnicas se caracterizam por requerer processos de maximização local para seu funcionamento. Neste trabalho, foi avaliada também a adoção da técnica de *Support Vector Machine* (SVM), por se basear em um processo de maximização global. Os resultados demonstram que as técnicas de geração de casos de teste ajustadas por cobertura precisam ser adaptadas às características do domínio de aplicação, para conseguir acelerar a execução dos *testbenches*.

ABSTRACT

Making functional verification more efficient in terms of computational and time resources is mandatory in order to maintain the evolution of digital systems. Coverage driven verification is one of the recently used alternatives for speeding up the execution of testbenches. Many approaches have been successfully applied to the functional verification of cores in the application domain of general purpose processors, however, being influenced by the specific coverage and testcase dimensionality characteristics of this domain. Furthermore, little attention has been given to the use of coverage driven verification in other domains, such as communication systems and multimedia systems. These domains have been considered in the present study, together with the specific factors that have influenced the coverage driven testbench results. Among these factors, one has identified the size of the testcase space and the distribution of the coverage events; making it necessary to the development of this work, several changes regarding the construction of the coverage driven testbenches.

Coverage driven testecase generation is performed by feedbacking the coverage status information and selecting those testcases that lead to the improvement of the coverage progression rate. This feedback depends on the construction of a model, by automatic learning, which relates testcases and the observations of coverage events. During this work, realistic large IP cores were verified with the following coverage driven techniques: Bayesian networks and classification tree data mining. These techniques, previously used in specific research works, adopt local optimization in their processing. In the present work, coverage driven verification with support vector machine learning, is tested due to the fact that this technique is based in a global optimization process. Results of this work have shown the need of adaptation of the coverage driven verification to the application domain characteristics, in order to obtain meaningful acceleration in testbench execution.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

B-VOP	*Bidirectional Video Object Plane*
CBP	Compositor de quadros
CDV	*Coverage driven verification*
CF	Cartão fraudado
C4.5	Algoritmo de aprendizado de árvores de classificação
CI	Circuito Integrado
COMET	*Coverage Measurement tool*
CRC	*Cyclic Redundancy Code*
DCT	*Discrete Cosine Transformation*
DCDCT	Decodificador dos coeficientes DCT
DVM	Decodificador de Vetores de Movimento
DUV	*Design Under Verification*
DSP	*Digital Signal Processing*
EM	*Expectation - Maximization*
FEC	*Forward Error Correction*
FHEC	*Forward Header Error Correction*
FN	Falso negativo
FP	Falso positivo
G	Gasolina
GSEIS	Grupo de Projeto de Sistemas Eletrônicos Integrados e Software Aplicado
GPP	*General Purpose Processors*
HEC	*Header Error Correction*
HDL	*Hardware Description Language*
I	Idade
IEC	*International Electrotechnical Commission*
ID3	*Iterative Dichotomiser 3*
IDCT	*Inverse Discrete Cosine Transformation*
IP	*Intellectual Property*
I-VOP	*Intra Video Object Plane*
ISO	*International Standardization Organization*
J	Joias
LME-USP	Laboratório de Microeletrônica da Universidade de São Paulo

MB	Macrobloco
MPEG	*Moving Picture Experts Group*
MPEG-4	Padrão de sistemas multimídia
P-VOP	*Predictive Video Object Plane*
QI	Quantizador Inverso
RGB	*Red Green and Blue*
RTL	*Register Transfer Level*
S	Sexo
SI	*Scan* Inverso
SMO	*Sequential Minimal Optimization*
SoC	*Systems-On-Chip*
SUM	Somador
STREAMPROC	*Stream Processor*
SVM	*Support Vector Machine*
UFCG	Universidade Federal de Campina Grande
VO	*Video Object*
VOL	*Video Object Layer*
VOP	*Video Object Plane*
WHITE	Módulo de redução no nível DC do STREAMPROC

1 INTRODUÇÃO

A grande capacidade de integração da atualidade tem permitido a construção de circuitos integrados contendo sistemas completos, conhecidos como sistemas-sobre-silício (do inglês *systems-on-chip* SoC). O desenvolvimento de sistemas-sobre-silício tem sido um grande desafio para os projetistas de sistemas de grande porte e complexidade, dada a dificuldade de se manter a consistência do sistema desenvolvido com a especificação. Uma das estratégias mais importantes utilizadas para lidar com tal complexidade tem sido o particionamento do sistema sendo projetado em sistemas menores, que possam ser reutilizados em futuros projetos(1). Esta estratégia é conhecida como projeto em plataforma, a qual pressupõe a criação de elementos de hardware e software que sejam facilmente reusáveis. Os elementos de hardware projetados para reuso denominam-se blocos de propriedade intelectual ou *cores* ou núcleos de hardware. A criação destes elementos requer que seja adotada uma forma padronizada de comunicação como, por exemplo, padrões de comunicação CORECONNECT (2), AMBA(3) e OPC-IP(4). Outro aspecto fundamental para o reuso é que os componentes garantam a conformidade da implementação com a especificação; para isto, é necessária a realização de etapas de verificação que permitam identificar e corrigir falhas introduzidas durante o projeto.

O elemento sobre o qual é realizada a verificação é denominado projeto sob verificação ou DUV, do inglês "*design under verification*"; tipicamente, é representado em linguagem de descrição de hardware (HDL) no nível de transferência entre registradores (RTL), ou seja, com uma precisão em ciclo de relógio. A verificação funcional tem como objetivo observar se o comportamento de um DUV tem conformidade com a sua especificação. A verificação tem como abordagens principais, a verificação formal e a verificação dinâmica por simulação. A verificação formal funciona ao redor de dois paradigmas principais: a checagem de modelos, e a prova de teoremas (5). Apesar de a verificação formal poder provar a existência (ou inexistência) de erros de projeto, esta prática apresenta limitações quanto ao tamanho do projeto sob verificação (5), enquanto a verificação dinâmica por simulação não as tem. Em grande parte da literatura técnica, o termo "verificação funcional" é usado de forma equivalente e alternada ao de "verificação dinâmica por

simulação", o que será adotado no restante desta tese. A verificação funcional é realizada num ambiente de simulação denominado *testbench*, onde casos de teste são gerados e aplicados ao projeto sob verificação e onde as respostas são avaliadas. A verificação por simulação é a estratégia padrão da indústria de CIs, sendo a sua importância evidenciada pela grande quantidade de ferramentas existentes para auxiliar na sua aplicação (Synopsys™ Vera(6), Cadence™ TestBuilder(7), Cadence™ Incisive (8), etc).

Para a verificação dinâmica, a quantidade de casos de teste possíveis cresce exponencialmente com o tamanho e a complexidade do sistema sob verificação, tornando inviável uma simulação exaustiva de todos eles. Para contornar esta dificuldade, os aspectos do DUV que mereçam ser revisados durante a simulação, são colocados, então, como os objetivos da verificação e devem ser determinados durante o planejamento desta. Estes objetivos são descritos num documento denominado plano de verificação, em forma de métricas, e correspondem quantitativamente a valores de cobertura; a cobertura é continuamente avaliada durante o processo de verificação para se determinar o estado corrente do processo e a proximidade ao ponto de finalização (9).

1.1 Motivação e justificativa

A forma mais óbvia de se realizar a verificação funcional sobre um núcleo de hardware é utilizar a geração direta de casos de teste, ou seja, o projetista prepara cada conjunto de estímulos com o objetivo de mapear alguma funcionalidade. Dada a complexidade dos circuitos atuais, a tarefa de obtenção dos casos de teste de interesse é penosa e demorada, apesar de que o número de casos simulados tende a ser limitado. Uma forma rápida de se alcançar de forma generalizada grande parte do espaço de entrada e de estados de um circuito é pela geração aleatória de estímulos. Nesta abordagem são criadas automaticamente grandes quantidades de estímulos e acompanha-se a verificação através de métricas de cobertura visando checar a observação das situações de interesse pelo engenheiro de verificação.

A Figura 1 apresenta a relação entre o estado da cobertura funcional e o tempo gasto no processo de verificação, para três abordagens de geração de casos de teste: a primeira é a geração dirigida onde a equipe de verificação cria um a um os

casos de teste para cada um dos eventos de cobertura identificados; a segunda é a geração aleatória onde é construído um gerador automático de casos de teste e a cobertura é conferida mediante o uso elementos monitores; e a terceira é a geração ajustada por cobertura, onde a informação do estado da cobertura é realimentada ao gerador aleatório. Análises realizadas sobre resultados dos *testbenches* têm mostrado, como indicado na Figura 1, que com respeito ao tempo de desenvolvimento e execução dos testbenches, a progressão da cobertura funcional é superior (mais rápida) para o caso da geração aleatória de casos de teste comparada à geração dirigida feita de forma manual (10). Uma forma de contornar o fenômeno de saturação da geração aleatória, mostrada na Figura 1, é realizar uma geração aleatória adaptada de forma incremental ao andamento da cobertura funcional. Isto pode ser realizado por meio de automatização da checagem da cobertura e da geração de casos de teste, no esquema de ajuste automático denominado de geração ajustada por cobertura, ou CDV do inglês "*coverage driven verification*" (5). A Figura 2 ilustra o princípio por detrás da CDV, pelo qual, deve-se capturar o relacionamento entre eventos de cobertura e estímulos para a simulação dos *testbenches*, para serem inseridos no módulo de decisão automática e se escolher, em determinado estado da cobertura, os casos de teste que podem fornecem um maior progresso na execução do *testbench* (almejando eventos de cobertura específicos). Desta forma, a CDV tem se mostrado capaz de proporcionar uma progressão da verificação funcional mais acentuada que a geração aleatória simples, como mostra a Figura 1.

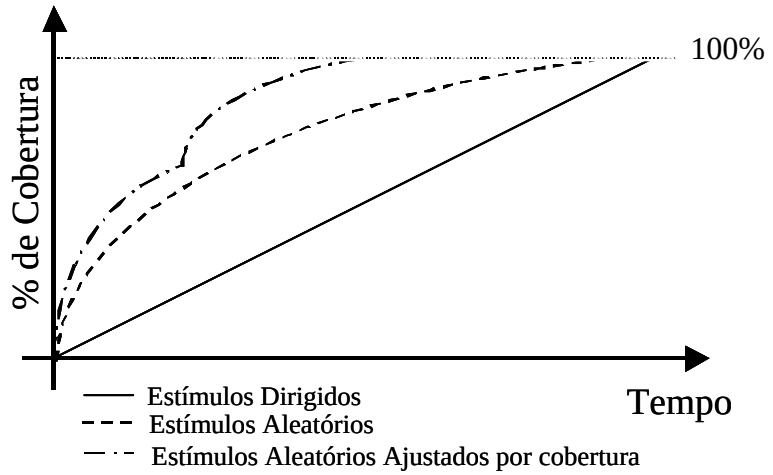

Figura 1 - Progressão da verificação

A literatura técnica apresenta exemplos de sucesso na aplicação de geração aleatória de casos de teste ajustada por cobertura funcional na verificação de processadores de uso geral. Duas abordagens têm sido adotadas para os casos de teste: por trechos de código e por operações unitárias. A primeira estratégia é basicamente utilizada em implementações completas de processadores de uso geral, aproveitando-se das características do código do programa para ajustar a geração aleatória de casos de teste. As técnicas utilizadas nesta abordagem têm sido: algoritmos genéticos (11), programação genética (12), modelos Markovianos (13) e programação por indução lógica (14). A segunda abordagem visa a geração de casos de teste para núcleo de hardware, nos quais os casos de teste correspondem a operações do hardware. Para esta abordagem, têm sido utilizadas as técnicas redes Bayesiana (15), *data mining* com árvores de decisão (16), e filtragem de estímulos (17). Dentre elas, as técnicas de redes Bayesianas e *data mining* se destacam, porque permitem criar modelos da relação entre os casos de teste e a cobertura resultante.

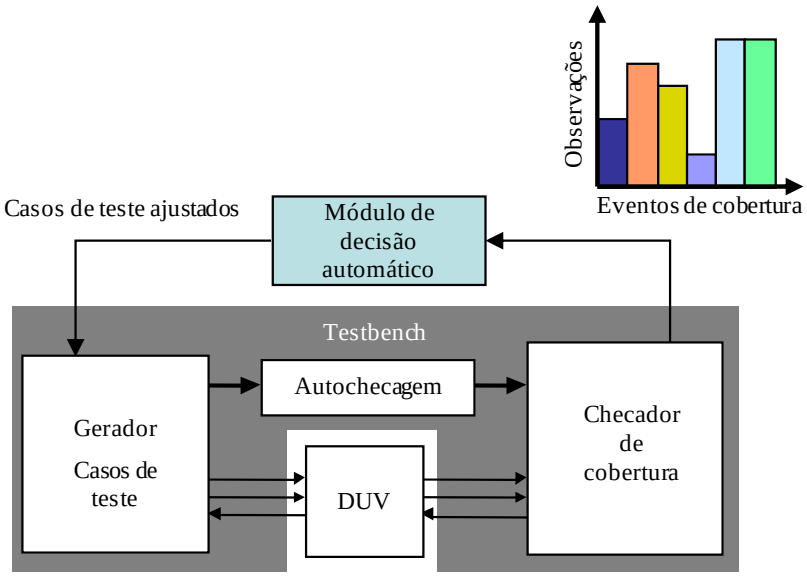

Figura 2 - Modelo simplificado do *testbench* reativo.

Embora os estudos sobre as técnicas de CDV tenham se mostrado bem sucedidos, eles não chegam a ser representativos porque consideraram principalmente o domínio de aplicação dos processadores de propósito geral, ou GPPs do inglês "*general-purpose processors*". No domínio de aplicação de processadores, a verificação é delimitada por características especificas, por exemplo, a necessidade

de representar uma grande diversidade de processos em função de operações básicas (18), a completa descrição da funcionalidade incluindo-se as suas interfaces (conceito de arquitetura), dadas pelo conjunto de instruções (19), a tendência ao uso de instruções curtas e rápidas em temos de execução (19). Tais características fazem com que os modelos de cobertura funcional usados sejam caracterizados pela baixa granularidade das transações, pelo tamanho específico do espaço de casos de teste e pela descrição no nível de caixa preta. Os modelos de cobertura cruzada e por itens que são amplamente usados na verificação de GPPs, não resultam igualmente úteis no domínio de sistemas de comunicação (20)(21). Isto se deve principalmente ao fato de que a funcionalidade nos sistemas de comunicação relaciona-se a um processamento mais orientado a dados do que a controle e muitos vezes as saídas contêm praticamente a mesma informação das entradas, mudando-se basicamente o formato e acrescentando-se redundâncias necessárias para contornar os defeitos do canal de comunicação; tal característica faz com que a funcionalidade deste tipo de sistemas não seja completamente expressa do ponto de vista das entradas e saídas.

Uma das razões que justificam um maior desenvolvimento de metodologias de verificação para GPPs é o fato de eles permitirem uma verificação mais estruturada devido à presença de arquiteturas computacionais comuns ou, pelo menos, correspondentes, o que se reflete também na microarquitetura. Apesar de famílias de GPPs apresentarem arquiteturas próprias, elas se caracterizam por traços comuns, como uso de instruções, esquemas de *pipelining*, registradores de uso geral, etc., o que facilita o reuso das técnicas desenvolvidas. Com respeito à granularidade das transações, estas costumam estar associadas às instruções do processador, no domínio dos GPPs (11) (12), caracterizando uma correspondência de entradas com poucos bytes para saídas, também, com poucos bytes. Em outros domínios como os de sistemas de comunicação ou sistemas multimídia, tem-se sistemas com maior processamento de dados que de controle, sendo comum que as operações sejam feitas em grandes quantidades de bytes tanto nas entradas quanto nas saídas (21) (22), tornando os espaços de entrada para cada operação comparativamente maiores que nos sistemas do domínio de GPPs.

As diferenças entre as características da verificação nos diferentes domínios fazem com que as características da geração de casos de teste ajustada por cobertura sejam diferentes também. Infelizmente, pouco tem sido produzido sobre CDV para núcleos de hardware no domínio de sistemas de comunicação e multimídia; inclui-se no primeiro grupo, algumas publicações realizadas no âmbito deste estudo (17) (20) e outras realizadas posteriormente para sistemas de rede (21) (23), enquanto, pelo nosso conhecimento, não há nenhum trabalho de CDV com sistemas de multimídia. Tampouco se fez até o momento comparação mais sistemática, com exemplos realistas e de grande porte, entre os esquemas de CDV baseados em diferentes técnicas de aprendizado; única comparação existente, porém específica para o domínio dos GPPs encontra-se em (16).

Por este motivo o presente trabalho aborda a aplicação de diferentes estratégias de CDV nos domínios de sistemas de comunicação e de sistemas multimídia. A análise da geração de casos de teste foi realizada aplicando-se as técnicas de ajuste com redes Bayesianas e *data mining* com árvores de classificação, uma vez que são aquelas mencionadas na literatura para ajuste por cobertura por operações (15) (16).

Nesta última década, Máquinas de Vetor de Suporte, ou SVM do inglês *Support Vector Machines* (24), têm ganhado destaque na literatura como uma técnica de classificação eficiente para atividades de aprendizado automático na área de reconhecimento de padrões (25), dada a sua alta capacidade de generalização. A aplicação de SVM requer uma fase de aprendizado por treinamento não supervisionado; a SVM tem apresentado melhores resultados quando comparado a outras técnicas de classificação semelhantes, segundo a literatura técnica (25), devido ao fato de o aprendizado ser resolvido como um problema de otimização matemática. Desta forma, a inclusão da técnica de SVM neste trabalho, como mais uma técnica de ajuste de cobertura, foi considerada como promissora, tendo sido então adicionada para o estudo comparativo desta tese.

1.2 Declaração do problema

A geração de casos de teste ajustada por cobertura é uma ferramenta necessária para se fazer a verificação funcional mais eficiente, porém a literatura mostra que esta abordagem tem sido aplicada primordialmente dentro do domínio de aplicação

de processadores de propósito geral. Por outro lado, a aplicação de CDV em sistemas de comunicação e multimídia é assunto pouco explorado, porém importante, pois trata-se de domínios de aplicação fundamentais dos sistemas digitais atuais, ademais, possuindo características de tamanho dos espaços de entrada e tipos de cobertura funcional diferentes daqueles usados no domínio de processadores. Adicionalmente, abordagens de CDV com técnicas de aprendizado diferentes podem gerar resultados diferentes nos *testbenches*, dependendo da sua eficiência e capacidade de generalização, sendo portanto, interessante um estudo aprofundado e comparativo entre elas no contexto de casos reais e de grande porte.

1.3 Objetivo geral

É objetivo geral deste trabalho, implementar e avaliar técnicas de geração aleatória de casos de teste ajustada por cobertura com redes Bayesianas, *data mining* baseada em árvores e *Support Vector Machine* (SVM) no contexto de núcleos de hardware nos domínios de sistemas de comunicação e sistemas multimídia.

1.4 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Criar uma proposta de metodologia para o uso de SVM na geração de casos de teste ajustados por cobertura, incluindo-se a definição do algoritmo de treinamento.

- Estabelecer as metodologias para o uso de *data mining* baseada em árvores de decisão e redes Bayesianas, na geração de casos de teste ajustados por cobertura, incluindo-se a definição dos algoritmos de treinamento.

- Aplicar as técnicas de SVM, redes Bayesianas e *data mining* ao caso de estudo do processador de cadeias (STREAMPROC) do protocolo *Bluetooth*.

- Aplicar as técnicas de SVM, redes Bayesianas e *data mining* ao caso de estudo do compositor de quadros (CBP) do decodificador para o padrão MPEG-4.

- Aplicar as técnicas de SVM, redes Bayesianas e *data mining* ao caso de estudo, preditor de blocos AC e DC (PIACDC) do decodificador para o padrão MPEG-4.

- Comparar criticamente os resultados das técnicas de geração ajustada por cobertura segundo as diferentes técnicas e diversas condições de aprendizado.

1.5 Contribuição do trabalho

As contribuições originais deste trabalho são:

- A abordagem da geração de casos de teste ajustada por cobertura foi estendida para os domínios de aplicação de sistemas de comunicação e sistemas multimídia.

- A criação de uma metodologia baseada na técnica de *Support Vector Machine* para a geração de casos de teste ajustada por cobertura, para avaliar a influência da otimização matemática no processo de aprendizado.

- A análise comparativa das técnicas de geração de casos de teste ajustadas por cobertura para blocos de hardware, através do cruzamento dos resultados do cálculo do erro obtido no aprendizado automático e os resultados obtidos com os *testbenches*.

- A análise comparativa das técnicas de geração de casos de teste ajustada por cobertura para blocos de hardware, a partir de exemplos realistas e de grande porte.

- O ajuste dos conjuntos de treinamento mediante a técnica denominada balanceamento para melhorar a eficiência da geração ajustada por cobertura

1.6 Organização do documento

A seguir, a forma como este documento está divido.

O segundo capítulo apresenta os conceitos associados à verificação funcional de sistemas em *hardware*, mostrando considerações importantes para a sua aplicação. Também são apresentados os algoritmos de aprendizado das técnicas de redes Bayesianas, *data mining* e SVM, em conjunto aos métodos tradicionais de avaliação de seus resultados.

O terceiro capítulo apresenta os métodos seguidos para a aplicação das redes Bayesianas e de *data mining* na geração ajustada por cobertura, e, em particular, o método desenvolvido neste trabalho para o uso de SVM.

O quarto capítulo apresenta os casos de estudo aplicados durante o presente trabalho, mostrando detalhes da sua funcionalidade e da definição de parâmetros para a sua verificação funcional.

O quinto capítulo mostra os resultados e análise da aplicação das técnicas de redes Bayesianas, *data mining* e SVM para a geração de casos de teste ajustada por cobertura nos casos de estudo dos módulos STREAMPROC, CBP e PIACDC.

Finalmente, o sexto capítulo apresenta as considerações finais e são indicadas as propostas para trabalhos futuros de pesquisa nesta área.

2 VERIFICAÇÃO FUNCIONAL AJUSTADA POR COBERTURA

Apresentamos neste capítulo conceitos associados à verificação funcional de sistemas em *hardware*, como *testbenches*, cobertura e geração de estímulos, mostrando, também, considerações importantes para a sua aplicação. Também são apresentados os algoritmos de aprendizado das técnicas de redes Bayesianas, *data mining* e SVM, além dos métodos tradicionais de avaliação de seus resultados.

2.1 Conceitos associados

A verificação funcional por simulação compartilha conceitos de outras áreas da eletrônica como a da confiabilidade de sistemas. Nela, um determinado sistema é analisado para se ter um comportamento coerente com a especificação durante determinado tempo. Em sistemas de software as principais divergências de comportamento são devidas a defeitos no projeto, já que os componentes destes sistemas não apresentam desgaste, fato que torna o seu desenvolvimento semelhante ao dos sistemas projetados com linguagens de descrição de hardware (HDLs, *hardware description languages*). Na teoria de confiabilidade de sistemas existem definições importantes sobre falha, avaria e erro[1,2] (5), as quais são apresentadas a seguir:

- Falha (*fault*): A falha é o defeito no sistema que pode produzir um comportamento divergente.
- Avaria (*failure*): A avaria é um comportamento divergente, que é detectável.
- Erro (*error*): O erro é a manifestação da falha nas saídas do sistema (ou então, a avaria quando observada à saída).

Seguindo esta lógica, no contexto da verificação funcional, a falha corresponde a um defeito na implementação do modelo no nível de transferência entre

[1] Esta terminologia é conhecida como a de grupo de Coimbra. Existe outra possibilidade, também, bastante difundida na literature técnica,referenciada como a de grupo de Lisboa, onde os termos adotados para *fault*, *failure* e *error*, são falta, falha e erro, respectivamente.

[2] Usamos esta terminologia, do grupo de Coimbra, para manter a consistência em relação a termos já consagrados como, por exemplo, tolerância-a-falhas, além de compatibilizar com o conceito de falhas utilizado na área de teste de manufatura.

registradores, RTL, do sistema, a avaria corresponde à ativação daquele defeito durante a simulação do sistema, o qual apresenta um comportamento distorcido, e o erro corresponde à manifestação da falha à saída do sistema ou nos elementos de monitoramento. Junto a tais conceitos vêm associados outros já utilizados na área de testabilidade, que são os conceitos da controlabilidade e da observabilidade. A controlabilidade está associada à capacidade de gerar estímulos nas entradas do sistema que excitem as falhas no código RTL, enquanto a observabilidade está associada com a capacidade dos detectores de erros poderem observar as ocorrências destas falhas (5).

2.2 Conceitos básicos

A discussão sobre verificação funcional por simulação requer que sejam apresentados conceitos importantes sobre *testbenches*, geração de casos de teste, autochecagem e cobertura.

2.2.1 Testbenches

O *testbench* é o ambiente de simulação utilizado para conferir o apropriado comportamento do DUV. Um modelo de *testbench*, adotado neste trabalho de pesquisa, é mostrado na Figura 3. Nesse ambiente, são realizadas conjuntamente as tarefas de geração de casos de teste, autochecagem e monitoramento da cobertura, como será mostrado nas sub-seções seguintes. No *testbench* estão presentes os conceitos de controlabilidade e observabilidade, que, segundo Wile, Gross e Roesner (5), devem ser bem considerados ao se realizar uma verificação: o primeiro, na controlabilidade nos geradores de casos de teste, e o segundo, na autochecagem. Adicionalmente a observabilidade é considerada também na cobertura para determinar o estado do processo.

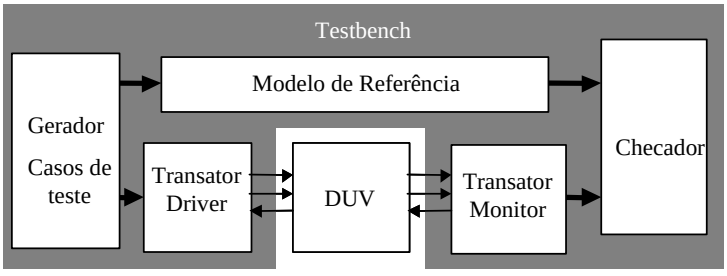

Figura 3 - Modelo de *testbench*

2.2.2 Geração de casos de teste

A geração de casos de teste consiste na criação de estímulos válidos segundo a especificação. Estes estímulos são geralmente abstraídos como casos de teste e depois são refinados a vetores de bits para serem aplicados diretamente nos pinos do modelo RTL (26). No presente documento é entendido um caso de teste como o conjunto de valores que são necessários nas variáveis de entrada do DUV para realizar uma operação completa. Os resultados da operação no DUV observados nos pinos de saída são abstraídos também em forma de variáveis e valores de saída. Finalmente, a informação de cada operação do DUV é representada como uma transação, que é a junção do caso de teste e os valores das variáveis de saída.

Como indicado na Figura 3, isto é realizado através de um transator driver. Existem duas formas principais para gerar casos de teste: a primeira, a partir de fontes dirigidas, e a segunda, a partir de fontes aleatórias. A geração por fontes dirigidas é utilizada principalmente quando o engenheiro de verificação identifica casos individuais de teste a serem aplicados de acordo com a interpretação que tem da especificação e a sua experiência. A geração com fontes aleatórias visa a criação automática de casos de teste, especificando-se distribuições de probabilidade para as suas variáveis de entrada. Este tipo de descrição foca aspectos que o engenheiro de verificação considera relevantes a serem checados durante a verificação, mas impondo algumas restrições para evitar situações que não sejam de interesse.

Na geração de casos de teste é importante ressaltar que eles são constituídos por dois tipos de variáveis: aquelas que definem o comportamento do DUV, denominadas parâmetros (27) e as que acionam operadores, denominadas variáveis de dados. Os parâmetros estão diretamente relacionados com as funcionalidades do sistema, já que eles determinam o modo de operação do DUV; exemplos de parâmetros seriam os códigos de operação de um processador de propósito geral; já as variáveis de operadores são mais genéricas, como, por exemplo, os valores nos registradores do processador.

2.2.3 Autochecagem

A estratégia de autochecagem propicia a detecção de erros no comportamento no DUV. Duas abordagens principais têm sido utilizadas: a primeira, usando modelos

da funcionalidade do DUV, conhecidos como modelos de referência (28), e a segunda através da checagem de propriedades usando asserções (28). No caso da Figura 3, a detecção é realizada através da inserção do modelo de referência (*golden model*), sabidamente correto, e do bloco Checador, que recebe as respostas do sistema aos casos de teste convertido pelo transator Monitor a partir de sequências de sinais RTL. Entretanto asserções são monitores inseridos no código do DUV que alertam no caso do não cumprimento de determinadas propriedades consideradas importantes pelos projetistas, desta forma indica-se o DUV se comporta dentro dos parâmetros estabelecidos.

2.2.4 Cobertura

As falhas que ocorrem em projetos com modelagem RTL têm múltiplas origens que vão de uma interpretação errada da especificação até enganos de diferentes tipos no momento de se escrever o código. Não existe na verificação funcional uma abstração de modelos de falhas como os existentes na área de testabilidade e, por este motivo, a cobertura não tem como objetivo quantificar falhas do DUV, mas procura-se quantificar a abrangência da simulação realizada. Assim a forma mais básica e trivial para conferir as funcionalidades do DUV é o teste exaustivo, que, entretanto, não é possível para a maioria dos DUVs, dada a quantidade imensa de casos de teste possíveis. Desta forma, para se avaliar o nível de abrangência da verificação funcional, tem sido necessária a criação de métricas conhecidas como cobertura. Segundo Piziali (9), a cobertura é uma representação dos objetivos da verificação do sistema, subjetivamente descritos pela equipe de verificação segundo a sua interpretação da funcionalidade do DUV.

Enquanto, no tipo de caixa preta, só é possível a observação do comportamento pelas interfaces externas do DUV, o modelo de caixa branca, por sua vez, já considera a observação de aspectos funcionais internos da descrição funcional ou do DUV como, por exemplo, dos estados das máquinas de estado como evento de cobertura. No modelo de caixa cinza, só alguns detalhes específicos internos são considerados, em conjunto a eventos de caixa preta.

Os diferentes níveis de observabilidade para a cobertura não correspondem a estratégias exclusivas entre si, mas complementares. Cada forma de cobertura tem a sua razão de ser e existir. Ainda que seja possível obter maior detecção de falhas

no tipo de caixa branca, o esforço para manter este nível de observação é muito maior, uma vez que é preciso acompanhar mudanças na implementação do DUV. Adicionalmente, este acompanhamento costuma ser mais difícil ainda após da síntese lógica do DUV ou após a realização de mudanças no projeto. A cobertura no tipo caixa preta, por sua vez, apresenta uma menor capacidade de observação, mas o seu uso tem vantagem nas etapas iniciais de projeto: como apenas as entradas e saídas precisam ser conhecidas, a construção do *testbench* pode ser iniciada quando o DUV está ainda em desenvolvimento.

2.2.4.1 Tipos de cobertura

A literatura descreve dois tipos de cobertura, a estrutural e a funcional. A cobertura estrutural visa abarcar de forma abrangente os elementos ou componentes que compõem o modelo do DUV. A cobertura funcional visa abarcar de forma abrangente as características funcionais do DUV. Denomina-se modelo de cobertura ao conjunto de diretrizes utilizadas para determinar a abrangência da verificação; com tais diretrizes, decide-se o evento a ser observado. Para o esquema de cobertura estrutural, que é tipicamente de caixa branca ou cinza, a literatura identifica as seguintes formas (29):

- Cobertura de linhas: esta cobertura mede a abrangência segundo a quantidade de linhas do código do DUV executadas durante a simulação. Este modelo de cobertura é utilizado para descobrir quais áreas do código RTL e do *testbench* que não foram estimuladas durante a simulação.

- Cobertura de ramos: esta cobertura mede nas estruturas condicionais quais dos ramos são utilizados. Com este modelo é possível identificar se, na execução do *testbench*, todos os ramos de execução possíveis junto às estruturas condicionais do código são contemplados. Este modelo de cobertura serve para indicar se condições fora do caminho normal, como exceções, são observadas na verificação

- Cobertura de condições: tal cobertura permite a medição nas estruturas condicionais aninhadas, testando todas as possibilidades de execução. Este tipo de cobertura é uma extensão da cobertura de ramos e considera as

possibilidades adicionais existentes quando há condicionais com dependência em mais de uma variável.

- Cobertura de caminhos: este modelo mede as combinações possíveis quando há condicionais descritos em sequência dentro do código do DUV. É também uma extensão da cobertura de ramos na qual são detalhados todos os possíveis comportamentos da lógica descrita no DUV. Este modelo tem uma tendência de crescimento exponencial no número de combinações, sendo difícil atingir 100% de cobertura.

- Cobertura de mudanças: este modelo de cobertura visa a checagem se mudanças em determinados sinais ou registradores do DUV acontecem. Com ele, busca-se identificar problemas de controlabilidade em determinadas variáveis do sistema.

- Cobertura de ativação: com a cobertura de ativação, objetiva-se mostrar se os sinais das listas de sensibilidade que ativam os processos do código HDL do DUV são utilizadas. Com este modelo de cobertura busca-se identificar problemas na lógica da implementação, já que esta particularidade não é realmente avaliada pelos modelos de cobertura anteriores.

A cobertura estrutural é muito utilizada nos meios industriais, porém, é sabido que não é efetivo porque existem falhas que necessitam de condições muito especificas nas entradas para se tornarem avarias e erros; essas condições são dificilmente representadas pelos modelos de cobertura estruturais. Isto faz com que seja necessário utilizar informações sobre a funcionalidade do DUV para descobrir inconsistências com a especificação, criando-se assim modelos de cobertura funcional (9). De forma equivalente à cobertura estrutural, a cobertura funcional também define modelos para a avaliação da abrangência da simulação. A cobertura funcional se baseia na checagem das funcionalidades do DUV, correspondendo a uma representação da especificação em listas, que são criadas durante a etapa de planejamento da verificação. A cobertura funcional não prova que cada funcionalidade está correta; simplesmente, mostra que ela foi executada e o quanto foi executada (30). Nesta, tese maior ênfase é dada à cobertura funcional por que descreve características associadas à semântica da descrição do DUV, a qual

resulta mais representativa que as características sintáticas fornecidas pelas coberturas estruturais.

Os modelos de cobertura funcional estão compostos de quatro componentes: os elementos de cobertura, os grupos de cobertura, o esquema de amostragem e os alvos de cobertura. Os elementos de cobertura representam eventos do DUV, expressos em forma de condicionais a serem observados durante a verificação. Os grupos de cobertura representam agrupações de elementos de cobertura segundo a semelhança na funcionalidade. O esquema de amostragem indica o momento da execução do DUV no qual os elementos de cobertura são avaliados na simulação. Finalmente o alvo da cobertura corresponde ao número de amostras dos elementos ou grupos de cobertura (9). O modelo por itens pode ser usado para ilustrar estes conceitos; por ele são estabelecidos como elementos, as observações dos valores das variáveis de entrada e saída do DUV, como por exemplo, valores individuais possíveis para tamanho de pacote; alternativamente, um tamanho possível de pacote poderia ser de (1-100 bits) ao invés de valores individuais. O momento de amostragem é ao final da cada transação quando, por exemplo, o pacote é processado e enviado. Pelo modelo, é costume que o número de amostras por grupo seja determinado proporcionalmente ao tamanho do grupo e segundo o critério de engenheiro de verificação; por exemplo, pode-se ter a determinação de se observar 10.000 amostras dentro da faixa (1-100 bits).

Os modelos de cobertura funcional do tipo caixa preta são apresentados principalmente na literatura pelos modelos por itens e cruzado. O modelo por itens tem como objetivo conferir que os valores das variáveis de entrada e saída sejam observadas durante a simulação do DUV, como já citado. O modelo cruzado visa também que as entradas e saídas sejam observadas em todos os seus valores, mas fazendo com que as combinações entre estas sejam observadas durante a verificação, com objetivo de dar um maior nível de detalhamento.

Modelos de cobertura branca e cinza são altamente dependentes da implementação; por exemplo, para sistemas hierárquicos, tem sido proposto o modelo de cobertura modular (18). Neste modelo, visa-se a checagem uniforme das unidades funcionais que compõem os DUV, que são blocos de hardware previamente verificados. Os eventos de cobertura neste modelo de cobertura são os ciclos de relógio nos quais

as unidades funcionais apresentam atividade, sendo observada, então, a uniformidade no teste das diferentes unidades funcionais. Este modelo de cobertura caracteriza-se por mostrar que os blocos internos são suficientemente utilizados quando operam conjuntamente no DUV. Este modelo de cobertura mostra-se simples e pouco intrusivo ao DUV, sendo desta forma fácil de adotar na verificação de diversos DUVs hierarquicamente implementados.

2.2.4.2 Cobertura segundo o domínio de aplicação

A cobertura funcional, como o seu nome o indica, possui uma forte correlação com a funcionalidade do DUV. Como apresentado anteriormente, a cobertura funcional visa a checagem de aspectos da funcionalidade identificados como relevantes pela equipe de verificação. Os sistemas digitais são agrupados dentro de domínios de aplicação de acordo com a semelhança nas suas funcionalidades (31), sendo claramente identificáveis, o domínio de sistemas de comunicação, o de processamento de sinais (DSP do inglês *digital signal processing*), o de multimídia, etc. Da mesma forma que os domínios de aplicação requerem diferentes modelos de computação para uma representação eficiente no nível de sistemas de suas funcionalidades, na verificação funcional diferentes modelos de cobertura são utilizados para a checagem de suas funcionalidades.

Para sistemas de comunicação, observa-se que a funcionalidade principal é a preparação dos dados para serem enviados por um canal de comunicação, onde normalmente estão expostos ao ruído. Estes sistemas arranjam os dados em formatos compatíveis ao protocolo e acrescentam determinada redundância para aumentar as chances de uma correta interpretação no outro extremo de canal. Para a checagem destas funcionalidades, os modelos de cobertura visam principalmente a checagem de eventos temporais, como em (20) os eventos correspondem ciclos de relógio e em (23) onde os eventos são a quantidade de pacotes transmitidos nos diferentes modos de operação do DUV. Já se tratando de sistemas de redes de comunicação, os eventos dos modelos de cobertura representam as condições de operação, como condições de tráfego, de utilização e de arbitragem (21). No domínio de sistemas de comunicação, observa-se que os modelos de cobertura têm níveis de visibilidade de caixa branca ou cinza, pois os dados observados junto às

saídas são cópias formatadas dos dados nas entradas, já que há pouca computação na transmissão.

Em sistemas multimídia, a funcionalidade é associada ao processamento de grandes quantidades de dados além de grandes transformações nos dados de saída com respeito aos de entrada; por exemplo, o decodificador MPEG-4 (22) apresenta eventos de cobertura que utilizam o modelo de cobertura por itens dos valores de saída (32) da mesma forma que decodificador do decodificador MPEG layer 3 apresentado em (32).

Para o domínio de processadores de propósito geral, os modelos de cobertura cruzado (34) e por itens são amplamente utilizados, com eventos de cobertura de nível de visualização de caixa preta, no relacionamento dos eventos de entrada com os eventos de saída (15)(16)(35)(36). Dado que as operações realizadas pelos processadores são principalmente lógicas e matemáticas as quais podem ser expressas facilmente em termos de relações de entrada/saída, portanto mediante a observação destas variáveis é possível inferir quanto da funcionalidade do DUV é testada.

2.3 Ajuste por cobertura funcional

2.3.1 Análise da cobertura

A geração de estímulos ajustada por cobertura é uma forma automática de otimização da verificação, composta de duas tarefas: a identificação dos desequilíbrios e insuficiências no estado da cobertura e o ajuste dos geradores de casos de teste para atingir a cobertura faltante. Uma das primeiras abordagens para a análise de cobertura é apresentada no trabalho de Grinwald et al. (33),onde é desenvolvida uma ferramenta denominada COMET (do inglês *Coverage Measurement Tool*). Neste trabalho a principal contribuição foi a criação de uma ferramenta genérica para a captura e processamento de qualquer tipo de modelo de cobertura funcional, facilitando a análise da cobertura, a qual era realizada de forma totalmente manual

Uma segunda abordagem para a análise de cobertura foi apresentada por Lanish (34), que descreve um método de análise de cobertura para o modelo de cobertura

funcional cruzada. Este método, denominado análise de lacunas, identifica como lacunas os elementos do modelo de cobertura não observados durante a simulação. Nesta análise as lacunas são agrupadas com o objetivo de identificar denominadores comuns nos valores dos eventos de cobertura que os distinguem. O agrupamento.

O agrupamento das lacunas de cobertura pode ser de diferentes tipos:

- Lacunas agregadas: este agrupamento inclui lacunas com semelhanças quantitativas, por exemplo, lacunas que, na representação de plano cartesiano do modelo de cobertura cruzada, apresentam a mesma distância de Hamming entre si.

- Lacunas particionadas: este agrupamento inclui as lacunas conceitualmente semelhantes, ou seja, lacunas que estejam associadas à mesma funcionalidade, ou no caso mais geral a funcionalidades semelhantes.

- Lacunas projetadas: este agrupamento considera a união de lacunas particionadas e agregadas, através de projeções sobre os planos cartesianos da representação do modelo de cobertura. Com este agrupamento busca-se aumentar a dimensão da lacuna com o objetivo de facilitar a identificação do motivo do desequilíbrio na cobertura.

Nesta abordagem a identificação dos problemas de cobertura é feita automaticamente, enquanto o ajuste dos geradores de casos de teste é feito manualmente.

Outra técnica para a análise da cobertura foi proposta por Asaf; Marcus e Ziv (35), que apresentam uma extensão da análise de lacunas. Por esta técnica, define-se um conjunto de vistas que ajudam a lidar com a grande dimensionalidade dos modelos de cobertura cruzada. Três operações básicas são apresentadas para a definição das vistas: projeção, seleção e agrupamento. A operação de projeção consiste em projetar os dados colhidos sobre um sub-conjunto de combinações do modelo de cobertura cruzada. Com esta operação busca-se identificar o progresso da cobertura sob determinadas áreas do modelo cruzado. A operação de seleção visa filtrar informação não interessante para a análise de cobertura. A operação de

agrupamento permite a avaliação do estado da cobertura de acordo com as funcionalidades. Similar à análise de cobertura por lacunas, a análise por vistas facilita a identificação de desequilíbrios na cobertura, mas o processamento é realizado manualmente.

As propostas para inclusão na verificação de estratégias, que ajustem automaticamente a geração de estímulos segundo o estado da cobertura, são conhecidas como *testbenches* reativos. O modelo geral de *testbench* reativo é apresentado na Figura 4, destacando-se nela o módulo de decisão, o qual determina como a geração de estímulos segundo o estado da cobertura deve ser alterada.

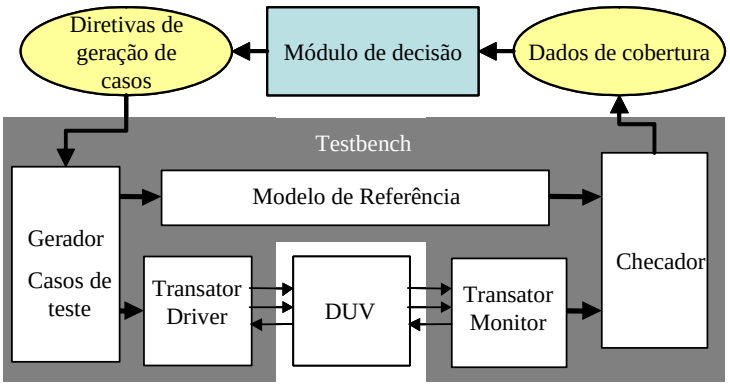

Figura 4 - Modelo de *testbench* reativo

Os *testbenches* reativos podem ser divididos em dois grupos, segundo o tipo de geração de estímulos utilizado nos *testbenches*: a de programas de teste e a de casos de teste. Os *testbenches* focados para a geração de programas são utilizados para a verificação de processadores e aproveitam características do código para o aprimoramento da cobertura. Por outro lado, a geração de casos de teste visa o teste de núcleos de hardware ou blocos de propriedade intelectual, IPs, através de sequências de transações. As duas abordagens têm sido implementadas com técnicas diferentes como descritas nos seguintes trabalhos:

– Programação Genética: no trabalho realizado por Corno et al. (12) , a técnica de programação genética é utilizada para a criação de programas de teste, focados na manutenção de uma alta taxa de progresso de cobertura. A programação genética é uma técnica que imita o processo de evolução das espécies e, na técnica abordada, durante a execução do *testbench*, são selecionados dinamicamente aqueles programas de teste que oferecem o melhor progresso na

cobertura. Um problema associado a esta técnica é o alto processamento adicional requerido, dado que a otimização baseia-se na avaliação de muitos conjuntos de programas de teste, denominados populações, dos quais somente uma fração é utilizada para a progressão da cobertura.

- Algoritmos Genéticos: no trabalho de Smith; Bartley e Fograty (11), algoritmos genéticos são usados para a criação de programas de teste em forma de sequências de linguagem *assembly*, o que torna os resultados bastante específicos. Com respeito à programação genética os algoritmos genéticos diferem no objeto ajustado, enquanto a programação gera diretamente o código do programa, o algoritmo genético configura um gerador de código parametrizável. O objetivo deste trabalho é a criação de programas de teste que consigam obter uma alta cobertura, sendo para isto necessário, como na programação genética, teste de múltiplos programas, além dos propriamente necessários.

- Modelos *Markovianos*: esta técnica, apresentada por Wagner, Bertaco e Austin (13), baseia-se em modelos *Markovianos* dos conjuntos de instruções do processador de uso geral. A modelagem é ajustada de acordo com a cobertura observada, reforçando a geração de determinadas sequências de instruções interessantes para a verificação. Esta técnica é altamente útil para a geração de programas de teste ajustados para estimular os casos extremos de cobertura.

- Programação por Indução Lógica: o trabalho feito por Hsueh e Eder (14), baseia-se no aprendizado automático através de uma tradução dos códigos dos programas de teste em forma de predicados, associando-se a eles a cobertura obtida. Com este modelo de predicados a geração de programas de teste é guiada utilizando-se a inferência de predicados. O método é aplicado em processadores de uso geral e os resultados indicam que para casos de cobertura com poucas amostras, o aprendizado é pobre.

- Redes *Bayesianas*: o trabalho de Fine e Ziv (15) descreve o uso de redes *Bayesianas* como uma plataforma, baseada em inferência estatística, para a modelagem e o aprendizado da relação existente entre os eventos da cobertura e os casos de teste. Neste trabalho, o aprendizado é feito a partir de um conjunto

de treinamento, formado por amostras dos casos de teste e a sua respectiva cobertura. O trabalho com redes *Bayesianas* caracteriza-se por ser uma abordagem quase automática, uma vez que é possível ajustar os pesos da rede segundo o conjunto de treinamento; por outro lado, a obtenção de uma estrutura da rede adequada é altamente dependente da existência de conhecimento especialista.

- *Data Mining por Árvores de Classificação*: no trabalho apresentado por Braun; Rosenstiel e Shubert (16), usa-se a técnica de mineração de dados (*data mining*) com árvores de classificação sendo que os resultados são comparados com os obtidos com redes Bayesianas usando-se um modelo de cobertura cruzado. A conclusão é que o *data mining* aumenta o nível de automatização do processo, já que esta técnica não requer o uso de conhecimento especialista na etapa de aprendizado. Também foi observada a dificuldade para a técnica *data mining* incorporar na sua modelagem os eventos de cobertura menos frequêntes do conjunto de treinamento.

- Filtragem de Estímulos: esta técnica desenvolvida por Romero; Strum e Wang (17) propõe o uso do modelo de referência para avaliar antecipadamente a cobertura funcional dos casos de teste. Esta avaliação tem como objetivo de selecionar os casos de teste que apresentem o melhor progresso na cobertura para serem aplicados ao modelo RTL. Embora este esquema de ajuste é de simples implementação, a sua aplicação traz limitações quanto aos modelos de cobertura que podem ser avaliados junto ao modelo de referência.

Outro trabalho que tem como objetivo o ajuste da geração dos estímulos durante a execução do *testbench* é apresentado no trabalho de Guzey e Wang (36),no qual é descrito o uso da técnica de *Support Vector Machine*, SVM, para o ajuste da geração de casos de teste. Este trabalho não trata de uma técnica de ajuste por cobertura, uma vez que não é utilizada uma realimentação do estado da cobertura para o gerador de estímulos. O trabalho não visa à aceleração do progresso da cobertura através do uso da informação do estado da cobertura, mas simplesmente visa à aplicação de casos de teste que sejam diferentes aos previamente aplicados. Parar isto os método, inicia a execução do *testbench* com geração aleatória, e ao observar alguma redução do progresso da cobertura, o aprendizado automático é utilizado

para criar um modelo que represente os casos de teste já aplicados. Este modelo é, então, utilizado para evitar a aplicação de casos de teste semelhantes aos previamente aplicados.

Na Tabela 1 apresentamos um resumo das técnicas de ajuste na geração de casos de teste durante a execução do *testbench*; nela observam-se duas abordagens exploradas as quais são o ajuste da geração segundo o estado da cobertura e o ajuste segundo a estimativa de similaridade entre os casos de teste. Nesta tabela são apresentados os trabalhos encontrados na literatura, junto com a proposta com SVM publicada com os resultados do presente estudo.

Tabela 1 - Técnicas de ajuste da geração de casos de teste dos *testbenches*

Ajuste por realimentação do estado da cobertura		Ajuste por similaridade entre os casos de teste
Trechos de código	Operações de hardware	Operações de hardware
• Programação genética (12) • Algoritmos genéticos (11) • Modelos Markovianos (13) • Programação por indução lógica (14)	• Redes Bayesianas (15) • *Data mining* (16) • Filtragem (17) • SVM (57)	• Análise com SVM (36)

2.3.2 Redução dos casos de teste e tempo de execução

A redução do número aplicado de casos de teste e, portanto, a redução do tempo de execução do *testbench* tem sido a principal forma de avaliação dos benefícios no uso da geração de casos de teste ajustados por cobertura. Nos trabalhos descritos anteriormente, é padrão fazer comparações entre os resultados obtidos pelas técnicas de otimização e os do esquema de geração aleatória simples, sem ajuste. Em geral, duas abordagens têm sido utilizadas: a primeira utiliza como referência o tempo requerido para atingir os objetivos definidos como de 100% de cobertura, e a segunda checa a quantidade de eventos de cobertura observados para um determinado número de casos de teste.

2.4 Aprendizado automático

2.4.1 *Data mining* com árvores de decisão

A construção de árvores de decisão ou classificação é uma das técnicas tradicionais da área de análise de grandes conjuntos de dados, mais conhecida como *data*

mining. A *data mining* busca a extração de informação tanto para aumentar a compreensão destes conjuntos quanto para a construção de modelos de predição (37). Apesar de que a *data mining* englobe várias técnicas diferentes, o termo *data mining* será utilizado neste trabalho para se referir unicamente ao uso de árvores de classificação.

2.4.1.1 Teoria

2.4.1.1.1 Árvores de classificação

Neste trabalho, dados são considerados conjuntos de objetos e classes. Os objetos são representados pelos valores designados à coleção dos seus atributos, os quais podem ser discretos, se a quantidade de valores possíveis é finita, ou contínuos, se os valores são numéricos. Adicionalmente, as classes são uma codificação dada a cada um dos objetos e, por sua vez, as classes são discretas e finitas.

A técnica *data mining* com árvores de classificação baseia-se na idéia de realizar uma divisão do espaço entre todos os possíveis objetos, em subconjuntos de acordo com as classes associadas aos objetos. Nesta técnica, a divisão é representada por meio de uma árvore de classificação, um grafo acíclico cujos vértices representam cada um dos atributos dos objetos e as arestas de saída de cada vértice representam um ou mais valores possíveis do atributo associado. A informação da classe dada a cada subconjunto é colocada nos vértices extremos ou vértices folha da árvore. Cada subconjunto é representado pelo caminho entre o vértice raiz e um vértice folha, sendo denominado de ramo. Cada ramo é também denominado de regra de classificação, uma vez que representa as condições a serem cumpridas pelos valores de cada atributo para estar associado a uma determinada classe 38. A Figura 5 apresenta um exemplo da classificação "sair para jogar tênis", cuja árvore permite observar os três atributos "Tempo", "Umidade" e "Vento". O atributo tempo pode receber os valores "Ensolarado", "Claro", e "Chuvoso"; o atributo umidade, os valores "Alta" e "Normal"; e o atributo vento, os valores de "Forte" e "Fraco". Do ponto de vista das regras, a árvore pode ser vista como uma estrutura de condicionais do tipo:

SE Tempo = Ensolarado E Umidade = Alta ENTÃO Classe → Não
SE Tempo = Ensolarado E Umidade = Normal ENTÃO Classe → Sim
SE Tempo = Claro ENTÃO Classe → Sim
SE Tempo = Chuvoso E Vento = Forte ENTÃO Classe → Não
SE Tempo = Chuvoso E Vento = Fraco ENTÃO Classe → Sim

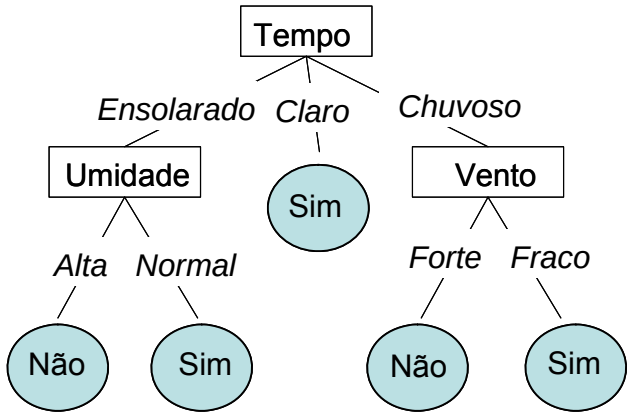

Figura 5 - Exemplo de árvore "sair para jogar tênis"

2.4.1.1.2 Aprendizado de árvores de classificação

A construção ou aprendizado com árvores de classificação visa a identificar uma divisão do conjunto de dados sob análise em subconjuntos do espaço, conformado pelas possíveis combinações de atributos e valores identificados nos objetos. O conjunto de dados sob análise é o denominado conjunto de treinamento. Além da descrição do conjunto de treinamento, é desejável que a divisão feita no aprendizado seja genérica o suficiente para que permita a classificação de outros objetos diferentes.

O ID3 (37) e o subsequente C4.5(39) são os algoritmos mais relevantes de aprendizado de árvores de classificação. Estes dois algoritmos criam recursivamente as árvores de acordo com a seguinte rotina:

SE todos os objetos do conjunto de treinamento pertencem à mesma classe
ENTÃO retorna uma árvore com um vértice folha contendo a classe
CASO CONTRÁRIO
ENTÃO
SE não há mais objetos
ENTÃO retorna uma árvore com um vértice folha com a classe por defeito
CASO CONTRARIO
ENTÃO seleciona um atributo diferente aos já utilizados para fazer a divisão e chama a criação de uma árvore por cada valor do atributo selecionado sobre os atributos restantes.

Um aspecto chave da seleção dos algoritmos de aprendizado de árvores é o critério de seleção dos atributos. Nos algoritmos ID3 e C4.5, o critério de seleção é baseado na relação entre a entropia antes de depois da seleção. A entropia é um conceito indicador que a quantidade de informação de um evento é função da probabilidade do mesmo, o qual é expresso na equação (1).

$$Info_{ev} = -\log\left(P_{ev}\right) \tag{1}$$

Para um subconjunto S o valor da probabilidade da informação de cada classe C_i esta dada pela equação (2), onde *k* é o número de classes, |*S*| é o tamanho do subconjunto e $freq(C_j, S)$ é uma função que indica a quantidade de objetos da classe C_j em *S*

$$Info(S) = -\sum_{j}^{k} \frac{freq(C_j, S)}{|S|} \times \log_2\left(\frac{freq(C_j, S)}{|S|}\right) \tag{2}$$

Aplicando-se estes conceitos para avaliar o efeito da divisão X do conjunto de treinamento T, produto da seleção de um determinado atributo, obtém-se a informação, apresentada na equação (3).

$$Info_X(T) = -\sum_{j}^{k} \frac{|T_i|}{|T|} \times Info(T_i) \tag{3}$$

O ganho de informação obtido após a divisão, apresentado na equação (4), serve como critério de divisão para o algoritmo ID3.

$$Ganho(X) = Info(T) - Info_X(T) \tag{4}$$

A pesar dos bons resultados obtidos com uso do critério de ganho de informação, ele apresenta uma grande parcialização em atributos, com muitos valores possíveis. Um exemplo desta situação é o caso hipotético de um sistema de diagnóstico médico onde existe um atributo de identificação de paciente, que é única, e leva a uma divisão em um grande número de subconjuntos, cada um tendo um só objeto. O cálculo da informação em cada subconjunto é igual a zero, fato que maximiza o valor do ganho de informação, mas que leva a uma divisão inútil do ponto de vista da predição. Para contornar esta deficiência, o algoritmo C4.5 propõe o uso do critério denominado relação de ganho.

A relação de ganho considera uma normalização sobre a informação de cada subconjunto a qual é expressa na equação (5).

$$NormInfo_X(T) = -\sum_j^k \frac{|T_i|}{|T|} \times \log_2\left(\frac{|T_i|}{|T|}\right) \quad (5)$$

O critério de relação de ganho é apresentado na equação (6)

$$Ganho(X) = \frac{Info(T)}{NormInfo_X(T)} \quad (6)$$

Em geral, os modelos de aprendizado têm a tendência a se sobreajustar aos dados do conjunto de treinamento; no caso das árvores, este efeito resulta em modelos que têm um grande número de folhas, mas com um baixo poder de predição para objetos não vistos no treinamento. Para compensar o sobreajuste, os algoritmos ID3 e C4.5 usam métodos de corte, os quais trocam algumas subárvores por vértices folha, reduzindo o número de ramos. Quando se dá a troca, uma subárvore é substituída por um vértice folha com o valor da classe com um maior número de objetos.

O critério de corte utilizado pelos algoritmos ID3 e C4.5 baseia-se numa estimativa do erro de classificação, calculado a partir dos elementos do conjunto de treinamento. O ID3 usa uma estimativa de erro que pressupõe uma distribuição uniforme dos objetos nas diferentes classes. Esta estimativa de erro é calculada com a equação (7), onde *N* é o número de objetos do subconjunto *S*, *k* é número de classes e *n* é o número de objetos associados com a classe majoritária.

$$E(S) = \frac{N - n + k - 1}{N - k} \quad (7)$$

Para estimativa de erro no algoritmo C4.5, é usado o limite de confiança superior da distribuição binomial, o qual é apresentado na equação (8) usando, por *default*, p_r=0,25.

$$U_{CF}(M, N) = \begin{cases} (1 - p_r)^N & \forall M = 0 \\ \sum_{i=0}^{M} \binom{N}{i} p_r^i (1 - p_r)^{N-i} & \forall M > 0 \end{cases} \quad (8)$$

O corte é realizado analisando a árvore das folhas até a raiz. Para isto é calculado o erro estimado de cada subárvore, somando-se o produto do número de objetos N pelo $U_{CF}(M,N)$, onde M é número de objetos classificados erroneamente. Esta soma é comparada com erro estimado da folha que poderia substituir a subárvore. O erro

estimado é calculado multiplicando-se o número de objetos cobertos pela subárvore N_f, pelo $U_{CF}(M_f,N_f)$, onde M_f é o número de erros produzidos ao realizar a troca. Quando a estimativa de erro é inferior para a folha do que para a subárvore, esta é trocada pela folha.

2.4.1.2 Vantagens e desvantagens

O aprendizado com *data mining* mostra-se vantajoso pela rapidez dos resultados, já que os algoritmos têm uma complexidade linear com o tamanho do conjunto de treinamento. O algoritmo C4.5 é uma evolução do algoritmo ID3 feita para melhorar o tratamento de atributos com um grande número de eventos, a capacidade de generalização e a capacidade de tratar atributos com valores contínuos. Por outro lado, a maximização do critério de ganho de informação faz com que os resultados dependam fortemente da etapa de seleção da ordem dos atributos. Adicionalmente, o corte pode mascarar detalhes importantes das classes menos observadas no conjunto de treinamento, aspecto que ficará patente da seção de resultados.

2.4.2 Redes Bayesianas

2.4.2.1 Teoria

As redes Bayesianas consistem de um formalismo que mistura a teoria dos grafos e a teoria da probabilidade, a partir das quais, faz-se uma representação de uma distribuição conjunta de probabilidade, P, de um conjunto de variáveis aleatórias $X = \{X_1,X_2, \ldots ,X_n\}$ (40). Cada rede Bayesiana está composta por dois componentes principais:

- A estrutura S: um grafo que define o relacionamento qualitativo causal entre os vértices;
- Os parâmetros numéricos θ: quantificam a relação probabilística causal entre os vértices.

Numa rede Bayesiana, cada vértice corresponde a uma variável $X_1, X_2, \ldots ,X_n$ e cada aresta representa a relação de dependência condicional entre estas variáveis. A probabilidade de uma instância *x* da variável X é descrita pela equação (9), onde pa_i é o conjunto de variáveis das quais X tem dependência, denominados pais de X.

$$P(x) = \prod_{i=0}^{n} P(x_i/pa_i) \tag{9}$$

Um exemplo de rede Bayesiana é apresentado na Figura 6, que pretende estabelecer a influência causal das variáveis Cartão Fraudado (CF), Idade (I) e Sexo (S) sobre as variáveis de compra de Gasolina (G) e de Jóias (J). Nesta figura, os vértices com círculos representam as variáveis do conjunto X; os arcos representam o relacionamento causal entre as variáveis.

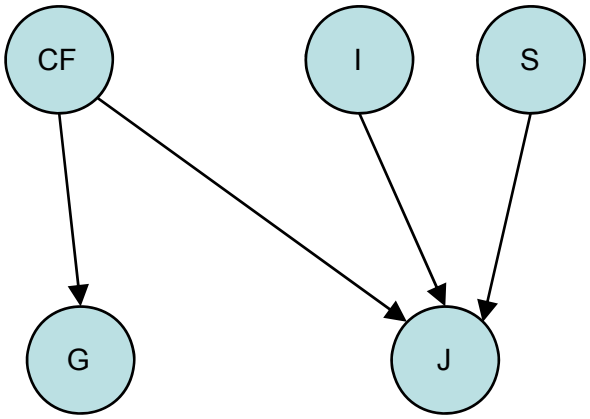

Figura 6 - Rede Bayesiana

A distribuição conjunta das variáveis da rede Bayesiana da Figura 6 pode ser expressa através da equação (10). Onde, P(CF) é representada pelos parâmetros numéricos apresentados na Tabela 2, P(I) é representada pelos parâmetros apresentados na Tabela 3, P(S) é representada pelos parâmetros apresentados na Tabela 4, P(G|CF) é representada pelos parâmetros apresentados na Tabela 5 e P(J|CF,I,S) é representada pelos parâmetros apresentados na Tabela 6. Nas tabelas 5 e 6 apresentam os valores das probabilidades condicionais, mostrando à esquerda do separado vertical os possíveis valores da variáveis aleatórias que têm dependência no vértice e à direita do separador os valores de probabilidade para cada um dos valores do vértice.

$$P(\mathrm{CF}, \mathrm{I}, \mathrm{S}, \mathrm{G}, \mathrm{J}) = \mathrm{P}(\mathrm{CF})\mathrm{P}(\mathrm{I})\mathrm{P}(\mathrm{S})\mathrm{P}(\mathrm{G}|\mathrm{CF})\mathrm{P}(\mathrm{J}|\mathrm{CF}, \mathrm{I}, \mathrm{S}) \tag{10}$$

Tabela 2 - Tabela de probabilidades da variável Cartão Fraudado P(CF)

P(CF=F)	P(CF=V)
0,99999	0,00001

Tabela 3 - Tabela de probabilidades da variável Idade P(I)

P(I=<30)	P((I>30)Λ(I<50))	P(I=>50)
0,25	0.40	0,35

Tabela 4 - Tabela de probabilidades da variável Sexo P(S)

P(S=Femenino)	P(S=Masculino)
0,5	0,5

Tabela 5 - Tabela de probabilidades da variável Gasolina P(G|CF)

CF	P(G=F)	P(G=V)
F	0,99	0,1
V	0,8	0,2

Tabela 6 - Tabela de probabilidades da variável Jóias P(J|CF,I,S)

CF	I	S	P(J=F)	P(J=V)
F	< 30	F	0,9995	0,0005
V	< 30	F	0,95	0,05
F	< 30	M	0,9999	0,0001
V	< 30	M	0,95	0,05
F	> 30 Λ < 50	F	0,998	0,002
V	> 30 Λ < 50	F	0,95	0,05
F	> 30 Λ < 50	M	0,9996	0,0004
V	> 30 Λ < 50	M	0,95	0,05
F	> 50	F	0,999	0,001
V	> 50	F	0,95	0,05
F	> 50	M	0,9998	0,0002
V	> 50	M	0,95	0,05

Numa rede Bayesiana, a partir dos dados, pode-se aprender indutivamente a estrutura da rede ou os valores dos parâmetros numéricos. Enquanto o aprendizado de parâmetros é relativamente simples, o da estrutura é em geral um assunto complexo e ainda não bem resolvido (41). O algoritmo denominado EM, do inglês *Expectation-Maximization* (42), é amplamente utilizado por permitir o aprendizado incluso quando há dados incompletos e variáveis ocultas, ou seja, cujos valores não são observados no conjunto de dados de entrada

O algoritmo EM faz uma estimativa de máxima verossimilhança dos parâmetros numéricos, ou seja, estima os valores dos parâmetros que sejam os mais consistentes como o conjunto de dados no sentido de maximizar a função de verossimilhança. (42)

O algoritmo EM pode ser aplicado nos casos onde se deseja estimar algum conjunto de parâmetros θ, que descreve uma certa distribuição de probabilidades conjunta, sendo que é dada somente uma parte (observada) dos dados produzidos por esta distribuição. Para ser mais preciso, considere que $X = \{x_1, . . . , x_m\}$ denote os dados observados em um conjunto de m instâncias ocorridas independentemente, seja $Z = \{z_1, . . , z_m\}$ os dados não observados nestas mesmas instâncias, e seja Y = X U Z, o total de dados.

Z pode ser tratada como uma variável aleatória cuja distribuição de probabilidades depende do conjunto de parâmetros desconhecido θ e dos dados observados X. Analogamente, Y é uma variável aleatória porque é definida em termos da variável aleatória Z .Para descrever a forma geral do algoritmo EM, h denota a hipótese atual dos parâmetros, e h' denota a hipótese revisada que é estimada em cada iteração do algoritmo EM.

O algoritmo EM busca a hipótese h' de máxima verossimilhança, isto é, que maximize E[ln P(Y|h')]. Este valor esperado é calculado sob a distribuição de probabilidade de Y, que é determinada pelos parâmetros desconhecidos θ. A explicação para esta função é: primeiro, P(Y|h') é a verossimilhança de todos os dados Y, dada a hipótese h' é razoável que se queira encontrar h' que maximize alguma função desta quantidade; segundo, maximizar o logaritmo desta quantidade ln P(Y|h') também maximiza P(Y|h'); e, terceiro, introduz-se o valor esperado de E[ln P(Y|h')] porque o total de dados Y é, ele próprio, uma variável aleatória

Dado que os dados Y são uma combinação dos dados observados X e não observados Z, deve-se mediar sobre os possíveis valores não observados Z, ponderando cada um de acordo com suas probabilidades. Em outras palavras, toma-se o valor esperado E[ln P(Y|h')] sobre a distribuição de probabilidade da variável aleatória Y. A distribuição de probabilidades de Y é determinada pelos valores completamente conhecidos para X, mais a distribuição de probabilidade de Z.

Em geral não se sabe qual é a distribuição de probabilidade de Y porque ela é determinada pelos parâmetros θ que se está tentando estimar. Entretanto, o algoritmo EM usa a sua hipótese atual h no lugar do parâmetro θ atual para estimar a distribuição de probabilidades de Y. Considere a definição de uma função Q(h'|h)

que dá E[ln P(Y|h')] como uma função de h', sob a suposição que θ = h e dada a porção observada X dos dados Y.

$$Q(h'|h) = E[lnP(Y|h')|h, X] \quad (11)$$

Escreve-se esta função Q na forma Q(h'|h) para indicar que ela é definida em parte pela suposição que a hipótese atual h é igual a θ.

Em sua forma geral, o algoritmo EM repete os dois passos seguintes, até a convergência:

Passo E: *Expectation* (E) calcula Q(h'|h) usando a hipótese atual h e os dados observados X para estimar a distribuição de probabilidade sobre Y. Este cálculo é realizado através da equação (11).

Passo M: *Maximization* (M) troca a hipótese h pela hipótese h' que maximiza esta função Q.

Quando a função Q é contínua, o algoritmo EM converge para um ponto estacionário da função de verossimilhança P(Y|h'). Quando esta função tem um único máximo, EM convergirá para esta estimativa de máxima verossimilhança global para h'. De outro modo, o algoritmo converge somente para um máximo local.

2.4.2.2 Vantagens e desvantagens

As redes Bayesianas apresentam como vantagem o fato de permitir a junção de informação do especialista com os dados obtidos, num ambiente probabilístico; estas características fazem que possa ser interessante para as mudanças das funções de probabilidade necessárias para o ajuste de geradores aleatórios de casos de teste. Porém as redes Bayesianas operam unicamente com eventos discretos, tendo desta forma uma tendência exponencial quando é necessário aprender dados de natureza numérica. Adicionalmente, o aprendizado com redes Bayesianas é limitado, dado que a maximização realizada durante o aprendizado é local.

2.4.3 *Support Vector Machine* (SVM)

Entre as diversas técnicas de aprendizado automático, a máquina de vetor de suporte, SVM, tem causado especial atenção nos últimos anos, devido ao fato de

fornecer uma classificação bi-classe com excelente desempenho (43). SVM funciona com as fases de treinamento e de classificação (24); na primeira, a SVM aprende a relação entre as variáveis denominada de entrada e as suas classes. Este aprendizado é realizado a partir de um conjunto de T de exemplos. Depois, durante a fase de classificação, o modelo de SVM gerado no treinamento com SVM está em condições de classificar novas entradas (diferentes do conjunto inicial).

2.4.3.1 Teoria

O SVM é uma técnica de classificação bi-classe, que identifica se um objeto pertence ou não a uma classe a partir de uma função matemática denominada hiper-plano de separação. O treinamento do SVM então consiste em encontrar uma superfície linear (hiper-plano), ou não (linear), capaz de separar o espaço de possíveis entradas de acordo com a sua pertinência a alguma das classes. De todas as superfícies possíveis, o SVM calcula aquela que fornece a margem de separação máxima entre as classes, sendo este cálculo formulado como um problema de programação quadrática. Dado que a solução do SVM para o problema de programação quadrática é global, esta técnica fornece uma classificação livre do problema de mínimos locais, portanto apresentando uma baixa dependência ao conjunto inicial de treinamento.

O cálculo do hiper-plano de separação é formalizado como um problema de otimização matemática o qual busca determinar um conjunto de coeficientes α_i, conhecidos como multiplicadores de Lagrange, que representam o hiperplano ótimo de separação dos grupos de classificação. A formulação do problema de programação quadrática para o caso de classes não separáveis linearmente é apresentado em (12)

$$\min_{\alpha_i} \left\{ L = \tfrac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j) - \sum_{i=1}^{T} \alpha_i \right\} \tag{12}$$

nas condições (13) e (14)

$$\sum_{i=1}^{T} \alpha_i y_i = 0 \tag{13}$$

$$0 \le \alpha_i \le C \quad i = 1....T \tag{14}$$

onde $K(\vec{x}_i, \vec{x}_j)$ é conhecida como a função kernel, T é o número de exemplos do conjunto de treinamento, C é um parâmetro associado ao erro de classificação aceitável (definido pelo usuário), x_j são os exemplos do conjunto de treinamento e y_j são as classes destes exemplos. No contexto da SVM, as classes são identificadas como +1 ou -1. Depois do cálculo dos multiplicadores de Lagrange, outras entradas x podem ser classificadas usando a função descrita em (15) e (16), sendo b calculado usando qualquer elemento k do conjunto de treinamento cujo multiplicador de Lagrange seja $0<\alpha_k<C$.

$$\operatorname{sgn}\left\{\sum_{i=1}^{T} \alpha_i y_i K(\vec{x}_i, \vec{x}) - b\right\} \tag{15}$$

$$b = \sum_{i=1}^{T} \alpha_i y_i K\left(\vec{x}_i, \vec{x}_k\right) - y_k \tag{16}$$

A eficiência da SVM baseia-se no fato de que, a computação é realizada apenas para os termos de índice *k* para os quais $\alpha_k \neq 0$, correspondendo ao vetor de suporte; como, em geral, uma série de parâmetros $\alpha_k=0$, a computação é simplificada sobremaneira.

Para o aprendizado com SVM o algoritmo *Sequential Minimal Optimization* foi proposto por Platt (43) para a resolução do problema de otimização apresentado na equação (12) de maneira analítica. O SMO é um algoritmo de decomposição que a cada passo soluciona o problema de otimização SVM de menor tamanho possível. Conseqüentemente, a quantidade de dados armazenados assim como a quantidade de operações aritméticas é consideravelmente reduzida. Isso resulta em um tempo menor de execução com economia de recursos aritméticos.

Uma iteração do algoritmo SMO é composta por três partes: seleção de um par de coeficientes, otimização dos coeficientes selecionados e atualização de dados. Estas etapas serão detalhadas a seguir.

Para encontrar o menor número possível de coeficientes a se otimizar por iteração, devemos considerar a restrição dada por (13). Dada esta restrição, temos que, para $0 < p < m + 1$:

$$\alpha_p y_p = -\sum_{i=1}^{p-1} \alpha_i y_i - \sum_{i=p+1}^{m} \alpha_i y_i = \text{constant} \tag{17}$$

Portanto α_p é constante e não pode ser modificado. Assim sendo, o menor subconjunto que pode ser otimizado contém dois multiplicadores de Lagrange, com a formulação, para $0 < \{p, q\} < m + 1$:

$$\alpha_p y_p + \alpha_q y_q = -\sum_{i=1}^{p-1} \alpha_i y_i - \sum_{i=p+1}^{q-1} \alpha_i y_i - \sum_{i=q+1}^{m} \alpha_i y_i = \gamma \tag{18}$$

Onde γ é constante.

A otimização é feita analiticamente e a velocidade da convergência é sujeita à escolha dos multiplicadores Langrangeanos, α_p e α_q a serem otimizados em cada iteração, assim como a velocidade de execução do algoritmo.

A equação (18) pode ser representada por uma reta com inclinação de 135 ou 45 graus em um plano de coordenadas (y_p e y_q podem ser +1 ou –1). Considerando também a restrição dada por $0 \leq \alpha_i \leq C$, os possíveis valores de α_p e α_q durante as iterações de otimização deverão estar contidos em uma superfície quadrada delimitada por 0 e C. A delimitação precisa pode ser obtida, para um par de valores α_p e α_q, pela análise em diferentes situações:

a) Se $y_p \neq y_q$: No intuito de obter os valores mínimos e os valores máximos para α partiremos das seguintes considerações: $y_p = +1$ e $y_q = -1$. Assim a equação (18) pode ser reescrita como:

$$\alpha_p - \alpha_q = \gamma \tag{19}$$

Portanto, γ variando entre $-C \leq \gamma \leq C$, como mostrado na Figura 7:

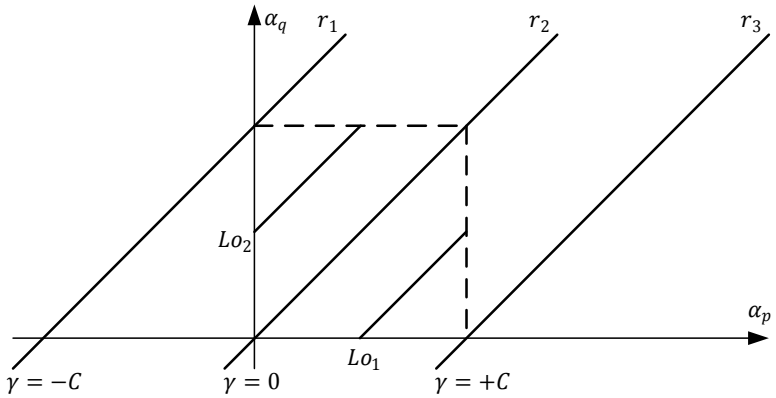

Figura 7 - Representação dos possíveis valores de α_p e α_q para $y_p \neq y_q$.

Entre r_1 e r_2 estão as retas para os valores de γ negativos permitidos e entre r_2 e r_3 estão as retas para os valores de γ positivos permitidos. Entre r_2 e r_3 o valor mínimo de α_2 é $Lo_1 = 0$. Entre r_1 e r_2 o valor mínimo de α_2 ocorre quando $\alpha_1 = 0$, portanto $Lo_p = -\gamma = \alpha_q - \alpha_p$. Portanto o valor mínimo de α_q é:

$$Lo = \max(0, \alpha_q - \alpha_p) \tag{20}$$

Seguindo o mesmo raciocínio para o valor máximo, conclui-se que:

$$Hi = \min(C, C + \alpha_q - \alpha_p) \tag{21}$$

Pode ser demonstrado que os mesmos resultados serão obtidos quando os valores de y adotados forem $y_p = -1$ e $y_q = +1$.

b) Se $y_p = y_q$: Para este segundo caso teríamos a situação indicada na Figura 8:

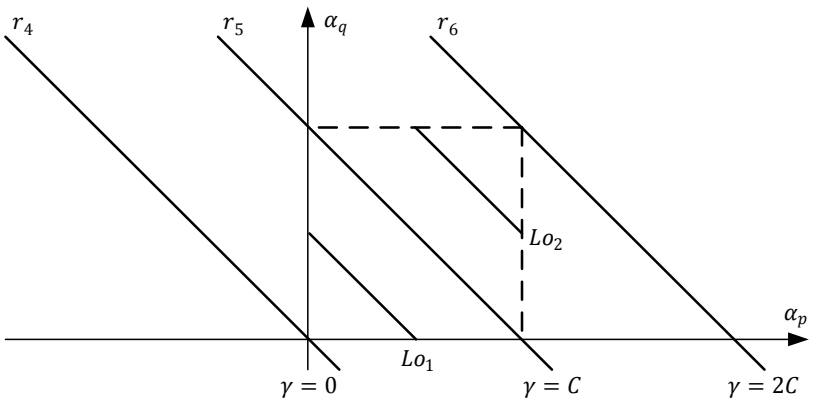

Figura 8 - Representação dos possíveis valores de α_p e α_q para $y_p = y_q$.

Aplicando novamente o raciocínio desenvolvido no item “a” obtêm-se as seguintes equações:

$$Lo = \max\left(0, \alpha_q + \alpha_p - C\right) \tag{22}$$

$$Hi = \min\left(C, \alpha_q + \alpha_p\right) \tag{23}$$

A partir das equações acima descritas é possível demonstrar através de desenvolvimento matemático (43)(44)(45) uma expressão para um novo valor de α_q no final de uma iteração:

$$\alpha_q^{new} = \alpha_q^{old} + \frac{y_q(b_{low} - b_{up})}{\eta} \tag{24}$$

Onde α_q^{old} é o valor do coeficiente α_q antes da otimização e η é equivalente a derivada segunda de $L(\alpha)$ em relação a α_q:

$$\eta = \frac{\partial^2 L(\alpha)}{\partial \alpha_2} = K\left(\vec{x}_p, \vec{x}_p\right) + K\left(\vec{x}_q, \vec{x}_q\right) - 2K\left(\vec{x}_q, \vec{x}_p\right) \tag{25}$$

Após o cálculo de α_q^{new} este deve ser corrigido de acordo com os limites estabelecidos por Hi e Lo, como definido em (26).

$$\alpha_q^{new} = \begin{cases} Hi \text{ se } \alpha_q^{new} \geq Hi \\ \alpha_q^{new} \text{ se } Lo < \alpha_q^{new} < Hi \\ Lo \text{ se } \alpha_q^{new} \leq Lo \end{cases} \tag{26}$$

E, b_{up} e b_{low} são dados por

$$b_{up} = \max\{F_i\}, i \in I_{up} \tag{27}$$

$$b_{low} = \min\{F_i\}, i \in I_{low} \tag{28}$$

Com:

$$I_{up} = \{t \mid \alpha_t < C, y_t = +1 \text{ ou } \alpha_t > 0, y_t = -1\} \tag{29}$$

$$I_{low} = \{t \mid \alpha_t < C, y_t = -1 \text{ ou } \alpha_t > 0, y_t = +1\} \tag{30}$$

e

$$F_i = -y_i \frac{\partial L(\alpha)}{\partial \alpha_i} = y_i - \sum_{j=1}^{m} \alpha_j y_j K\left(\vec{x}_j, \vec{x}_i\right) \tag{31}$$

Já o cálculo de um novo valor de α_p após uma iteração α_p^{new} pode ser definido utilizando a restrição (18):

$$\alpha_p^{new} = \alpha_p^{old} + y_p y_q \left(\alpha_q^{old} - \alpha_q^{new}\right) \tag{32}$$

O algoritmo SMO baseia-se em uma heurística para a seleção dos pares α_p e α_q a serem otimizados. Os multiplicadores selecionados ao final são aqueles que correspondem a b_{up} e b_{low}. Pode ser matematicamente provado que ao menos um dos coeficientes escolhidos não satisfaz as condições KKT se a condição $b_{up} > b_{low}$ for verdadeira (45). Se $b_{up} \leq b_{low}$, então todos os coeficientes satisfazem as condições KKT e o algoritmo pode ser finalizado. Portanto, a computação de b_{up} e b_{low} serve tanto para a escolha do par a ser otimizado, assim como para critério de parada. Essa verificação é realizada com uma tolerância ε.

A cada iteração, dados são atualizados e um novo par é selecionado. Ao final de uma iteração, α_p^{new} é definido segundo as condições de (32) e α_q^{new} é computado de acordo com equações (24), (25) e(26). Os valores de I_{up} e I_{low} devem ser atualizados, assim como o valor do novo F_i :[3]

$$F_i^{new} = F_i^{old} + y_p\left(\alpha_p^{old} - \alpha_p^{new}\right)K\left(\vec{x}_p, \vec{x}_i\right) + y_q\left(\alpha_q^{old} - \alpha_q^{new}\right)K\left(\vec{x}_q, \vec{x}_i\right) \tag{33}$$

Desta forma, b_{up} e b_{low} podem também ser atualizados para uma nova iteração se a minimização geral não tiver sido alcançada.

Como a última etapa do algoritmo, o valor de b pode ser calculado como:

$$b = \frac{b_{up} + b_{low}}{2} \tag{34}$$

Os desenvolvimentos acima descritos podem ser resumidos em três tarefas sequenciais principais, que estarão associadas a blocos de arquitetura a serem apresentados na seção 4.4. As tarefas são descritas a seguir :

a) Seleção: procedimento para a escolha do par de coeficientes a ser otimizado, caracterizado pelas equações (27), (28), (29), (30) e (31).

[3] Os multiplicadores lagrangeanos são iniciados com valor zero, enquanto que a função F_i é iniciada com os valores de y_i.

b) Otimização: procedimento para otimizar os coeficientes escolhidos na tarefa a), caracterizado pelas equações (24), (25), e (26).

c) Atualização: com os novos coeficientes otimizados, fazer a atualização dos parâmetros de computação de cada exemplo, de acordo com a equação (33).

2.4.3.2 Vantagens e desvantagens

A principal vantagem do SVM é a garantia de ter uma classificação otimizada, porém o processo de treinamento tem um alto custo computacional, tanto assim que existem estudos com implementações em hardware deste processo (54)(46). Adicionalmente, Burges (25) menciona a dificuldade do aprendizado com SVM quando a diferença de elementos das classes no conjunto de treinamento é muito grande.

2.4.4 Métodos para avaliação dos modelos aprendidos

2.4.4.1 Matriz de classificação

Para medir a exatidão dos modelos aprendidos é comum a realização de um teste de classificação. Para este teste é utilizado um conjunto de objetos já classificados, conjunto denominado conjunto de classificação. Os objetos do conjunto de classificação são aplicados ao modelo aprendido e os resultados obtidos são comparados com as classes originais. A matriz de classificação é uma forma detalhada de apresentar os resultados da comparação. Nesta matriz é feito o cruzamento do que é obtido pelo modelo e o que era esperado (obtido por simulação). Um exemplo desta matriz é apresentado na Tabela 7, onde as colunas representam as classes resultado do modelo, as filas representam as classes do conjunto de classificação e cada célula representa a quantidade de objetos correspondentes a cada fila e coluna. Por exemplo, na figura, a fila associada à classe 1 indica que, de todos os exemplos do conjunto de classificação que efetivamente referem-se à classe 1 (sabe-se por simulação), 52 deles foram corretamente classificados como tal pelo modelo aprendido, porém 10 foram erroneamente classificados como da classe 2. Igualmente, 7 exemplos foram erroneamente classificados como de classe 3 e, assim por diante.

Tabela 7 - Exemplo de matriz de classificação

		Classificado pelo modelo					
		1	**2**	**3**	**4**	**5**	**6**
Original conjunto de classificação	**1**	52	10	7	0	0	1
	2	15	50	6	2	1	2
	3	5	6	6	0	0	0
	4	0	2	0	10	0	1
	5	0	1	0	0	7	1
	6	1	3	0	1	0	24

A somatória dos valores de cada fila indica a quantidade objetos de cada classe no conjunto de classificação obtido via simulação. Já a somatória em cada coluna indica a quantidade de objetos classificados em cada classe pelo modelo aprendido. As células na diagonal que inicia na célula superior esquerda indicam os acertos da classificação e todo o resto das células indica os erros.

2.4.4.2 Erros de falso positivo e falso negativo

Do ponto de vista do classificador, existem dois tipos de erro para cada uma das classes que possui: de falso positivo, FP, e de falso negativo, FN. O erro de FN para uma determinada classe acontece quando os objetos que no conjunto de classificação obtido por simulação pertencem à classe determinada, porém, na classificação com o modelo aprendido, são associados a outra classe. Por outro lado, o erro de FP acontece quando os objetos que pertencem a outras classes no conjunto obtido por simulação são associados na classificação pelo modelo aprendido à classe determinada.

A taxa erro de FN para uma determinada classe pode ser calculada a partir da matriz de classificação, simplesmente somando-se os valores das células com erro da fila da classe e dividindo pela somatória daquela fila. A taxa de erro de FP de determinada classe é calculada somando-se as células com erro da coluna da classe e dividindo pela somatória daquela coluna. De forma geral, a taxa de erro de FP é calculada com a equação (35) e a do erro de FN pode ser calculado com a equação (36).

$$FP(classe_i) = \frac{\sum erros\ na\ classe_i}{\sum elementos\ classificados\ na\ classe_i} \tag{35}$$

$$FN(classe_i) = \frac{\sum erros\ na\ classe_i}{\sum elementos\ da\ classe_i\ no\ conjunto\ de\ classificação} \tag{36}$$

2.4.4.3 Efeito de erro de treinamento no ajuste por cobertura

Pode-se obter a redução da quantidade de casos de teste e tempo necessários para atingir determinada cobertura quando o ajuste da geração de casos de teste é feito de forma apropriada. Para isto é necessário que a realimentação feita através do modelo aprendido durante o treinamento seja bem sucedida, porém é natural, neste tipo de modelo, a presença de defeitos devidos às condições de treinamento ou as limitantes da técnica de aprendizado. A avaliação destes defeitos é comumente realizada através do cálculo de erro de classificação, do erro de FP e do erro de FN como apresentado na seção anterior.

A avaliação detalhada de erro em cada classe ajuda a descrever o comportamento do modelo aprendido na execução do *testbench*, pois, para realimentar a informação sobre quais eventos de cobertura precisam de aceleração, as decisões sobre a escolha dos casos de teste depende da predição feita pelo modelo. Nessa realimentação os defeitos do modelo afetam o ajuste de formas diferentes.

O erro de FN é prejudicial para a realimentação porque evita que sejam aplicados os casos que aceleram a classe atrasada, fazendo que não seja possível de fazer a realimentação. Neste caso o testbench funciona como se não tiver ajuste por cobertura. Portanto com o erro de FN, a aceleração é inibida voltando à progressão da cobertura observada no *testbench* sem CDV.

Para o erro de FP, o efeito é mais prejudicial, porque, na realimentação, a predição do modelo indica que seja aplicado um caso de teste que incrementa a cobertura numa outra classe diferente da necessária, fazendo com que, em muitos casos, seja uma classe associada a um evento já avançado na cobertura. Desta forma, o erro de FP, além de inibir a aceleração, torna a execução do *testbench* mais lenta, pois transfere o esforço de avanço na cobertura em eventos que não precisam dela, fazendo que os *testbench* seja incluso mais devagar que *testbench* sem CDV.

2.5 Trabalhos correlatos aos objetivos da tese

O presente trabalho de doutorado está focado na aplicação e avaliação das técnicas de geração de casos de teste ajustada por cobertura para blocos de propriedade intelectual, nos domínios de aplicação de sistemas de comunicação e sistemas

multimídia. Por este motivo os trabalhos correlatos comparáveis com a presente proposta são os trabalhos (15), (16), (21) e (17) e que descrevem o uso das técnicas de redes Bayesianas, *data mining* e filtragem de estímulos. Nestes trabalhos as técnicas de ajuste com redes Bayesianas foram utilizadas exclusivamente no domínio de processadores de propósito geral (15), enquanto o *data mining* foi utilizado nos domínios de aplicação de processadores de propósito geral (16) e de sistemas de rede (21).

As outras técnicas apresentadas na Tabela 1 para o ajuste por cobertura não são comparáveis uma vez que dependem das características dos trechos de código para o seu funcionamento, portanto sendo unicamente aplicáveis à verificação de processadores de uso geral, ou não usam a informação da cobertura para realizar o ajuste da geração de casos de teste.

3 METODOLOGIA DE GERAÇÃO DE CASOS DE TESTE AJUSTADA POR COBERTURA

Nesta seção são apresentadas duas metodologias de inclusão da geração de casos de teste ajustada por cobertura nos *testbenches* para a verificação funcional de blocos de propriedade intelectual. A Figura 9 apresenta as metodologias que descrevem os passos necessários para tornar *testbenches* com geração aleatória, cobertura como aquele apresentado na Figura 3, em *testbenches* com geração aleatória ajustada dinamicamente por cobertura, como apresentado na Figura 4. Estas duas metodologias constam das etapas de treinamento e aplicação, as quais são apresentadas nas próximas seções. A primeira metodologia, apresentada na Figura 9A, é seguida em ambos os trabalhos realizados com redes Bayesianas e *data mining* (15)(16). A segunda metodologia é a proposta deste trabalho para o uso da técnica SVM para o ajuste da geração de casos de teste (Figura 9B). A diferença entre as duas metodologias verifica-se em dois momentos: na categorização de classes, que necessita de tratamento específico no caso de SVMs, por estes serem tipicamente classificadores bi-classe, e no modelo para seleção de novos casos de teste (por modelo inverso ou por filtragem). Detalhes maiores serão fornecidos nas próximas seções.

3.1 Metodologia de geração ajustada usando redes Bayesianas e *data mining*

A metodologia para a criação de *testbenches* com ajuste automático, nos geradores de casos de teste aleatórios, baseado no estado da cobertura funcional requer duas etapas: o treinamento e a aplicação. A primeira etapa visa a construção de um modelo da relação entre casos de teste e eventos de cobertura, usando aprendizado automático sobre árvores de classificação e redes Bayesianas. A segunda etapa consiste no uso do modelo obtido na etapa anterior, já na aplicação da verificação propriamente dita, para o ajuste dos geradores aleatórios de casos de teste. As duas etapas de metodologia são descritas em detalhes nas seções seguintes.

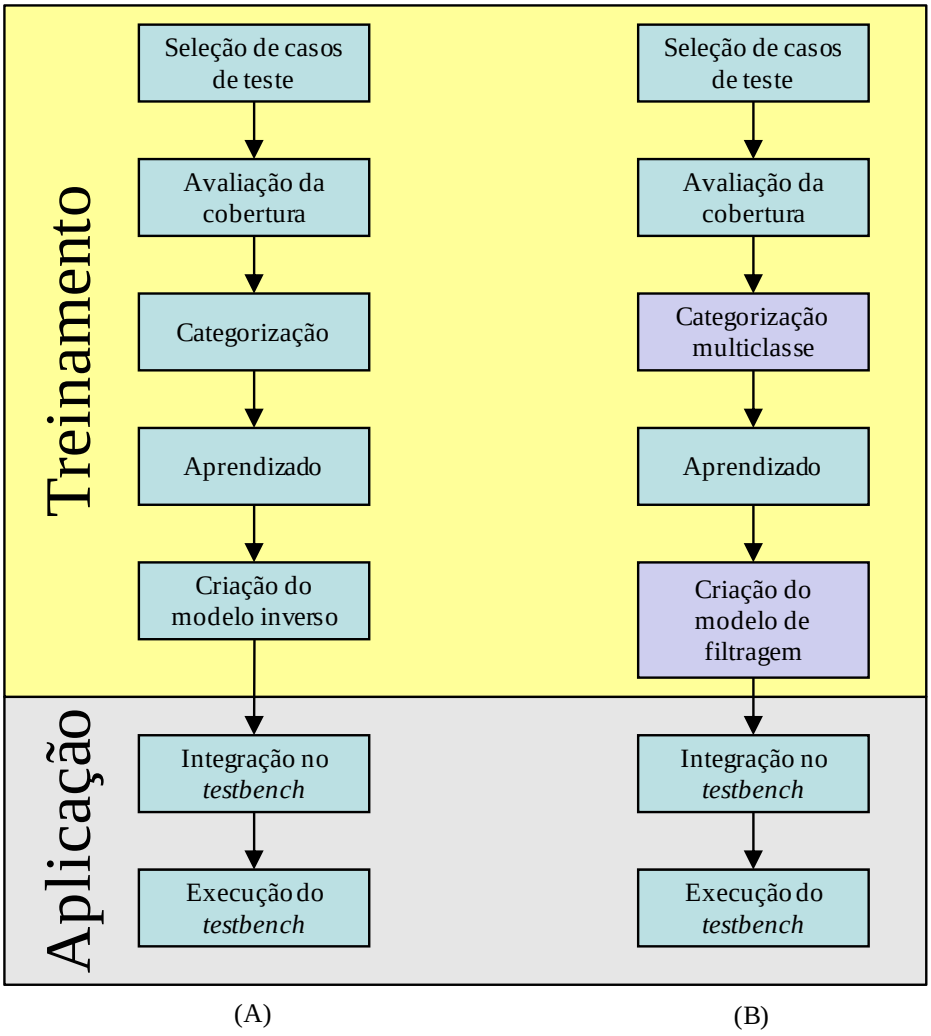

Figura 9 - Metodologias para a construção de *testbenches* ajustados por cobertura. (A) metodologia para redes Bayesiana e *data mining*. (B) Metodologia para SVM

3.1.1 Etapa de treinamento

A etapa de treinamento é composta pela seguinte sequência de tarefas:

3.1.1.1 Seleção dos casos de teste

O treinamento inicia-se com a seleção dos casos de teste que formarão o denominado conjunto de treinamento. Esta seleção pode ser feita de forma aleatória no conjunto dos casos de teste válidos, ou pode ser feita criteriosamente pelo engenheiro de verificação. A seleção de casos de teste influencia os resultados obtidos no treinamento, uma vez que uma melhor qualidade da informação fornecida nesta tarefa resulta em um melhor modelo. Portanto, nessa tarefa, o conhecimento especialista é de fundamental importância; infelizmente, a informação da relação entre os casos de teste e os eventos de cobertura não faz parte da especificação e

não costuma estar disponível até que a verificação tenha sido executada. Isto faz com que, na prática, seja necessário recorrer ao uso da seleção aleatória para se tentar cobrir a maioria dos aspetos desta relação. A seleção aleatória também auxilia na caracterização da capacidade de aprendizado das técnicas de modelagem, permitindo a análise destas técnicas independentemente da influência do engenheiro de verificação.

3.1.1.2 Avaliação da cobertura

O conjunto de casos de teste a ser utilizado para o treinamento é executado no *testbench* para se identificar os eventos de cobertura que são ativados por cada um dos casos de teste. Estes eventos são os mesmos considerados no planejamento da verificação funcional e anteriormente implementados no *testbench* com geração aleatória sem ajuste.

3.1.1.3 Categorização

O treinamento deve ser feito com o conjunto de casos de teste que estão associados a categorias de evento de cobertura. Adotam-se categorias ao invés de eventos, pois um caso de teste pode ativar mais de um evento de cobertura (dependendo como a cobertura foi planejada). O objetivo da categorização é associar a cada caso de teste uma categoria única, atribuindo códigos a cada uma das combinações dos eventos ativados com cada caso de teste. Podem, então ocorrer dois casos:

- no caso em que os eventos ativados sejam excludentes, não é necessário realizar uma codificação adicional, ou seja, cada categoria corresponde a um evento de cobertura. Um exemplo ilustrativo deste caso são os eventos E1=A[0..5] e E2= A[6..10]; então Cat1=E1 e Cat2=E2.

- quando dois ou mais eventos poder ocorrem para um mesmo caso de teste, combinações excludentes devem ser atribuídas às categorias. Um exemplo ilustrativo deste caso são os eventos E1=A[0..6] e E2= A[6..10]; então a categorização poderia resultar em Cat1=E1e^E2, Cat2=^E1eE2, Cat3=E1eE2.

O resultado da categorização é o denominado conjunto de treinamento o qual é composto pelos casos de teste selecionados juntamente com as suas categorias correspondentes.

3.1.1.4 Aprendizado

Durante o aprendizado é construído automaticamente um modelo que descreve a relação existente entre cada caso de teste e a sua categoria associada. Este modelo é criado aplicando-se as técnicas aprendizado automático ao conjunto de treinamento. No caso das redes Bayesianas é utilizado o algoritmo conhecido como EM (do inglês *Expectation-Maximization*) apresentado na página 28. Para o aprendizado de *data mining* a construção de árvores de classificação é realizada com o algoritmo denominado C4.5, apresentado na página 24.

3.1.1.5 Criação do modelo inverso

Como apresentado no *testbench* funcional da Figura 4, e aqui replicado na Figura 10 por conveniência, o processo de ajuste da geração requer a identificação do conjunto de casos de teste que tem determinada categoria associada. Como indicado na Figura 10, o módulo de decisão realiza tal papel, porém é necessário que o modelo obtido no processo de aprendizado seja invertido.

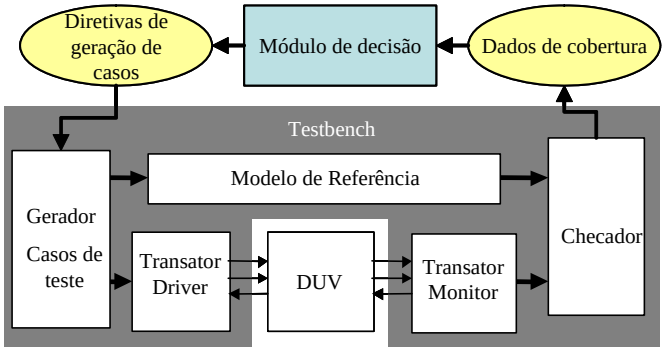

Figura 10 - Modelo detalhado de *testbench* reativo

No caso com as redes Bayesianas esta característica é inerente, ao modelo associado, apresentado na página 46, uma vez que é possível propagar os valores de probabilidade de forma bidirecional. O modelo inverso é estabelecido diretamente pela anotação da categoria desejada junto aos vértices de saída da rede e identificação dos valores dos parâmetros entrada (para do caso de teste), a partir das distribuições de probabilidade observadas nos vértices de entrada da rede.

Por outro lado, no caso de *data mining*, há a necessidade de se construir um novo modelo inverso. Isto é feito agrupando-se os ramos da árvore de classificação que apontam numa mesma categoria. Depois em cada grupo, um ramo é utilizado de

cada vez para identificar as condições que devem ser cumpridas pelos valores dos parâmetros de entrada dos casos de teste. Com este modelo inverso é possível identificar para uma determinada categoria, um caso de teste que satisfaça os condicionais de algum dos ramos do grupo e portanto da categoria.

3.1.2 Etapa de aplicação

Durante a etapa de aplicação, o modelo relacionando casos de teste e eventos de cobertura (obtido segundo a metodologia descrita na seção 3.1.1), é utilizado para ajustar a geração de casos de teste. Para este objetivo é necessária a realização das tarefas descritas nas seções seguintes.

3.1.2.1 Integração no *testbench*

A Figura 11 ilustra as adaptações feitas no modelo de *testbench* da Figura 3, para implementar os elementos requeridos para o uso da geração de casos de teste ajustada por cobertura. Observa-se que são feitos dois acréscimos através do submódulos:

- submódulo *Indicador:* é acrescentado no módulo *Checador*, realizando a operação de análise do estado da cobertura e a identificação dos eventos de cobertura menos observados, sendo realimentado para o gerador casos de teste o evento de cobertura mais atrasado.

- submódulo *Ajuste*: é acrescentado ao gerador de casos de teste, ajustando as probabilidades do diferentes valores das variáveis de entrada do DUV no gerador de casos de teste para aumentar as chances de ativar o evento identificado como mais atrasado.

Na operação de análise de cobertura, o submódulo *Indicador* identifica o evento de cobertura com o menor nível de observação. Esta informação é enviada ao módulo gerador de casos de teste no submódulo *Ajuste*, onde o modelo inverso (descrito na seção 3.1.1.5 da página 46) é utilizado para forçar a geração de casos de teste que ativem o eventos de cobertura indicado como mais atrasado.

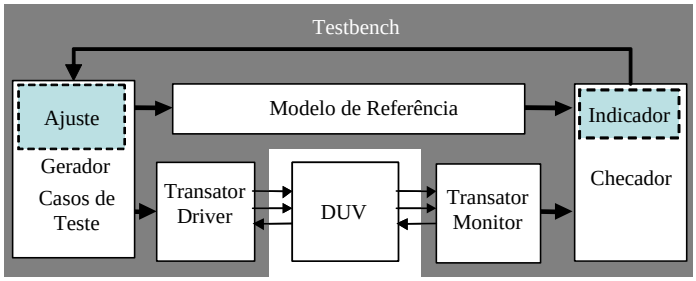

Figura 11 - Modelo de *testbench* com geração ajustada por cobertura

3.1.2.2 Execução do *testbench*

Finalmente, com o modelo de *testbench* completo, é feita a verificação funcional especificando-se o número de amostras requeridas para cada um dos eventos de cobertura. Com o objetivo de cobertura definido, o *testbench* reativo procurará atingi-lo modificando o perfil dos estímulos de acordo com as necessidades definidas pela análise da cobertura.

3.2 Metodologia de geração ajustada usando SVM

A metodologia de geração ajustada por cobertura usando SVM é similar à de redes Bayesianas e *data mining* apresentada na seção 3.1. As eventuais diferenças serão explicitadas nas seções a seguir.

3.2.1 Etapa de treinamento

Na etapa de treinamento com SVM as tarefas de seleção dos casos de teste e a avaliação da cobertura são realizadas como apresentado nas seções 3.1.1.1 e 3.1.1.2.

3.2.1.1 Categorização multi-classe

Na tarefa de categorização para o SVM, primeiro é realizada a codificação para fazer com que as categorias sejam excludentes; depois é necessária uma segunda codificação dos eventos de cobertura para tornar a classificação multi-classe a partir de várias classificações bi-classe (um modelo para cada categoria). Técnicas como redes Bayesianas e *data mining* suportam, de forma nativa, a classificação multi-classe, enquanto a técnica SVM requer esquemas adicionais para tornar a sua classificação típica, bi-classe, em uma multi-classe. Portanto no SVM são construídos vários conjuntos de treinamento, assim como é apresentado na Figura

12. Pode-se reparar que o mesmo conjunto de casos de teste selecionados é mantido e são criados vários conjuntos de treinamento, cada um correspondendo a uma categoria. A Figura 12 ilustra como o conjunto de classificação multi-classe se expressa em quatro conjuntos bi-classe correspondentes para os eventos de cobertura 1, 2, 3 e 4, apresentados de cima para baixo na parte direita da figura. Em cada um destes conjuntos, estabelece-se o código +1 para os casos de teste correspondentes à categoria do modelo, ou classe associado a esse conjunto, e -1 quando o caso de teste corresponde a outras categorias. Este esquema foi escolhido pela simplicidade, já que não requer mudanças nos casos de teste selecionados.

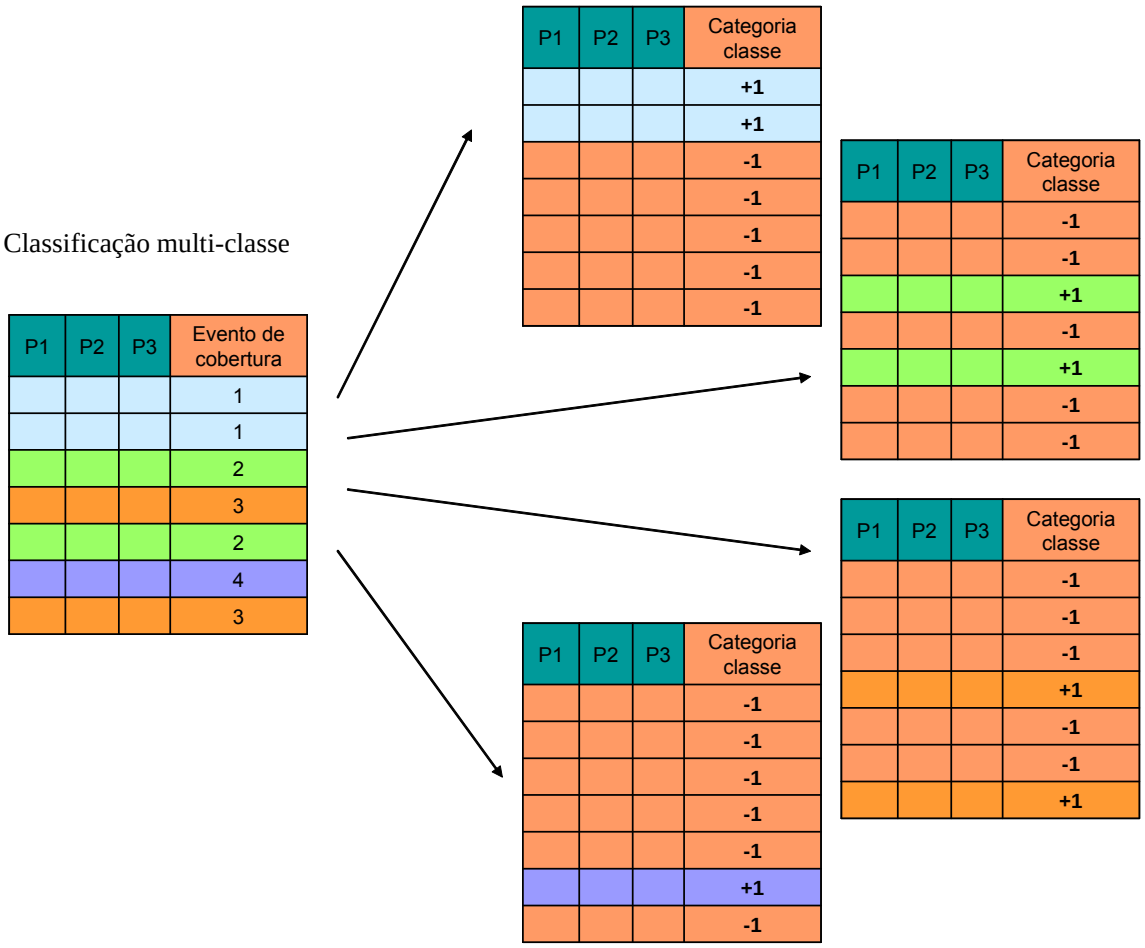

Figura 12 - Classificação multi-classe com múltiplos bi-classe

3.2.1.2 Aprendizado

Durante o aprendizado cada um dos conjuntos de treinamento bi-classe é aplicado ao algoritmo de aprendizado SMO apresentado na página 33. O resultado desta tarefa é um conjunto de modelos bi-classe que representam as relações entre os casos de teste e cada uma das categorias.

3.2.1.3 Criação do modelo de filtragem

No caso de SVM, como os coeficientes α são os resultantes do treinamento, a inferência inversa não é possível; portanto, a escolha do caso de teste no ajuste é feita por meio de um esquema de filtragem. Neste esquema, o gerador aleatório é utilizado como na forma original do *testbench*, criando um a um os casos de teste, sendo eles avaliados com os modelos SVM para descobrir a categoria a qual o caso de teste corresponde. A filtragem consiste então em escolher dos casos criados aqueles que são correspondentes com a categoria alvo e descartando o resto. Nos casos onde não é encontrada correspondência com a categoria alvo, é definido um limite de procura após o qual um caso de teste aleatório é selecionado.

3.2.2 Etapa de aplicação

3.2.2.1 Integração com o *testbench*

O *testbench* com ajuste na geração de casos de teste usando SVM segue o esquema apresentado na Figura 11. Neste *testbench* com SVM o submódulo *Indicador* é igual ao descrito na página 47, enquanto para o submódulo *Ajuste* é introduzido o esquema de filtragem descrito na página 50.

3.2.2.2 Execução do *testbench*

A tarefa de execução é realizada da mesma forma como apresentado na página 48.

4 CASOS DE ESTUDO

O presente trabalho utiliza três casos de estudo para verificar o comportamento dos *testbenches* com geração aleatória de casos de teste com ajuste por cobertura. Nesta seção, os casos de estudo são apresentados mostrando a funcionalidade própria de cada um deles, os aspectos relevantes da sua implementação e da sua verificação funcional. Na seção 4.1 é apresentado o primeiro caso do estudo processador de pacotes (STREAMPROC) do protocolo *Bluetooth* (47), que corresponde a um bloco típico do domínio de sistemas de comunicação. Na seção 4.2 são apresentados dois casos: o copiador de blocos (denominado CBP) e o preditor de coeficientes AC e DC (denominado PIACDC) do decodificador MPEG-4, casos representativos de blocos de sistemas de processamento multimídia.

4.1 Adaptador do protocolo *Bluetooth*

O *Bluetooth* (versão 2.1) é um padrão de protocolo para comunicação sem fio a curta distância, desenvolvido pelo *Bluetooth* SIG (*Special Interest Group*) (47). A arquitetura física do *Bluetooth* está composta por duas camadas que são implementadas em hardware: a de enlace RF, para comunicação via canal aéreo entre os dispositivos, e de banda base, que funciona como adaptador para executar protocolos e rotinas no nível mais baixo da implementação *Bluetooth,* acima da camada RF. A Figura 13 mostra, de forma simplificada, a arquitetura das camadas do sistema *Bluetooth.* Além das camadas anteriores, existem também: o LMP (*Link Management Protocol*), encarregado de administrar a criação, modificação, ou encerramento de enlaces com outros dispositivos; o L2CAP *(Logical Link Control and Adaptation Protocol*), que multiplexa e demultiplexa múltiplos canais através de um enlace lógico compartilhado, e executa tarefas de segmentação e reagrupação de dados; o RFCOM, que é um “protocolo de emulação de cabo serial” que permite conectar um dispositivo serial com a camada L2CAP de um dispositivo *Bluetooth*; e, finalmente, o SDP (*Service Discovery Protocol*), que gerencia recursos para que as aplicações possam descobrir que serviços estão disponíveis nos dispositivos conectados à rede, e determinarem as características de tais serviços.

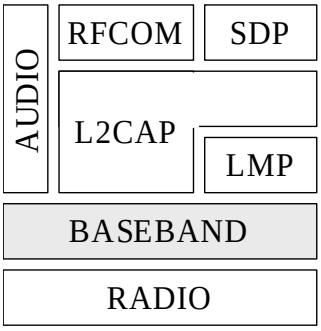

Figura 13 - Estrutura simplificada das diferentes camadas do padrão *Bluetooth.*

O adaptador do protocolo *Bluetooth* proposto com caso de estudo para este trabalho, tem como função principal o processamento de *bits* e a sincronização requerida para a comunicação de dados com outros dispositivos *Bluetooth*. A funcionalidade realizada pelo adaptador é descrita detalhadamente na seção de banda base do documento de especificação do protocolo (47). O adaptador foi implementado em *SystemC* no Grupo de Projeto de Sistemas Eletrônicos Integrados e Software Aplicado (GSEIS) do Laboratório de Microeletrônica da USP (LME-USP) para o projeto Brazil-IP (48).

O protocolo *Bluetooth* realiza a comunicação entre dispositivos por meio de troca de pacotes. Os pacotes *Bluetooth* devem seguir formato indicado na Figura 14 e são compostos por três partes fundamentais: o código de acesso, que pode conter 68 ou 72 *bits* dependendo do tipo de pacote, e permite a identificação individual de cada pacote trocado sobre o enlace físico; o cabeçalho, que contém informação para o controle do enlace como o tipo de pacote e informação para correção de erros de transmissão; e, finalmente, a carga útil, que contem também um segundo cabeçalho.

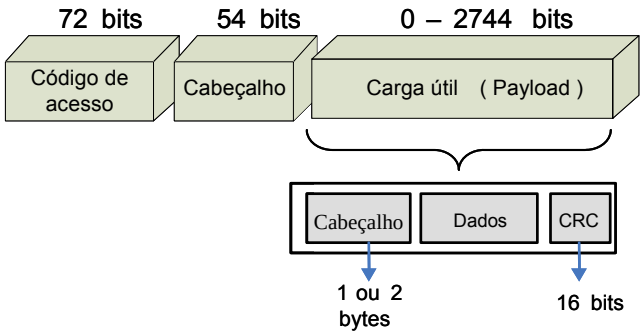

Figura 14 - Formato do pacote da camada banda base *Bluetooth*.

O protocolo *Bluetooth* estipula que os pacotes têm tamanho variável dependendo da informação contida neles. Os pacotes do tipo ID só tem o código de acesso (ACCESS CODE) na versão de 68 *bits*, sendo utilizados exclusivamente durante o estabelecimento da conexão. Outros pacotes como POLL e NULL têm código de acesso e cabeçalho, mas não têm carga útil. O resto dos pacotes são apresentados na Tabela 8, têm o seu tamanho informado dentro do cabeçalho da carga útil.

Tabela 8 - Tipos de Pacote *Bluetooth*, segundo tipo do cabeçalho.

Nome	**Carga útil**	**Taxa de transmissão simétrica (Kb/s)**	**Taxa de transmissão assimétrica (Kb/s)**
NULL	0 *bits*	0	0
POLL	0 *bits*	0	0
FHS	240 *bits*	64	64
DM1	18 *bytes*	108,8	108,8 - 108,8
DH1	28 *bytes*	172,8	172,8 - 172,8
AUX1	0 *bits*	0	0
DM3	123 *bytes*	256	384,0 – 54,4
DH3	185 *bytes*	384	576,0 – 87,4
DM5	226 *bytes*	286,7	477,8 – 36,3
DH5	341 *bytes*	432,6	721 – 57,6

O conteúdo do código de acesso depende do estágio da transmissão e, em geral, é formado a partir dos 24 *bits* da parte menos significativa do endereço do dispositivo. Quanto ao cabeçalho do pacote, ele contém informação para controle do enlace e, de forma geral, está composto por cinco campos, distribuídos ao longo de 10 *bits*, como mostrado na Figura 15. O campo HEC é adicionado pelo módulo gerador de código de correção de erros. O campo TYPE, de tipo de pacote consiste de um código de 4 *bits* que especifica qual dos diferentes tipos de pacote está em uso. Informações sobre os demais campos do cabeçalho do pacote podem ser encontrados no manual de especificação do protocolo *Bluetooth* (47).

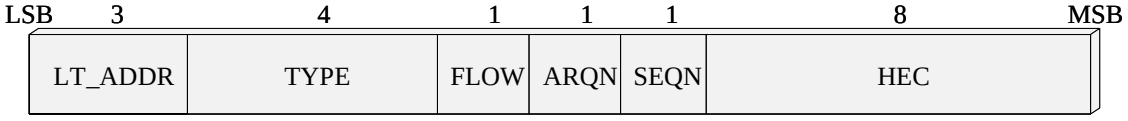

Figura 15 - Formato do cabeçalho do pacote.

4.1.1 Módulo de processamento de pacotes STREAMPROC

4.1.1.1 Funcionalidade

A função principal da camada banda base no protocolo *Bluetooth* é organizar em pacotes a informação a ser transmitida entre os diferentes pontos de conexão. Tais pacotes possuem um formato bem definido na especificação (47), como ilustrado na Figura 14. Para a montagem dos pacotes do protocolo *Bluetooth* é necessária a realização dos processamentos apresentados na Figura 16 para o cabeçalho e na Figura 17 para a carga útil.

Com o objetivo de ser enviada a um eventual receptor, a informação da conexão deve passar por um processo de codificação e formatação para garantir a confiabilidade, segurança, e exclusividade dos dados transmitidos. A Figura 16 mostra o processamento que deve ser realizado sobre o cabeçalho de qualquer pacote *Bluetooth*, transmitido ou recebido. O código de correção, HEC, é acrescentado para checar a existência de erros originados no processo de transmissão do cabeçalho. Em seguida, os *bits* do cabeçalho são embaralhados pseudo-aleatoriamente (*whitening*) para evitar padrões redundantes, e assim, minimizar a polarização DC do pacote. Finalmente, os *bits* do pacote são protegidos por uma codificação FEC 1/3 para correção de erros de transmissão. O procedimento inverso deve ser aplicado ao pacote quando der sua chegada no receptor.

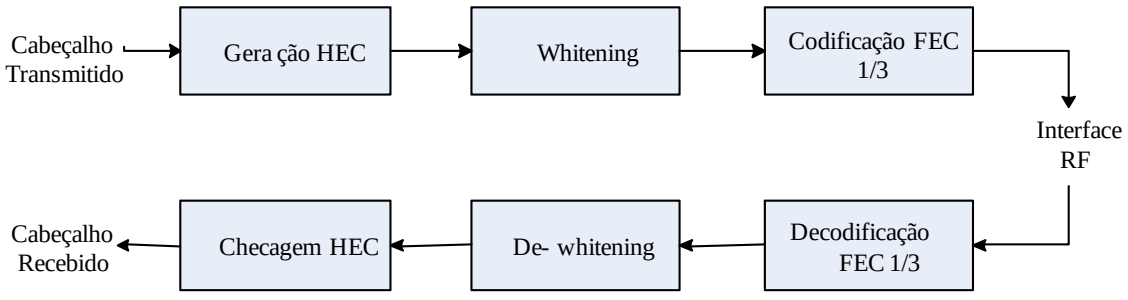

Figura 16 - Operações realizadas sobre o cabeçalho do pacote.

Similar ao cabeçalho, a informação da carga útil do pacote (Figura 17) é também processada, sendo utilizados: um código CRC para detecção de erros, uma função adicional de encriptação (ou decriptação no receptor) para proteção e segurança da informação do usuário, e o uso de codificação FEC 2/3 para correção de erros

eventualmente ocasionados na interface RF. Tanto para o cabeçalho, quanto para a carga útil, as codificações sementes para o embaralhamento e para a geração do CRC, de sete e oito *bits* respectivamente, são trocadas entre os dispositivos.

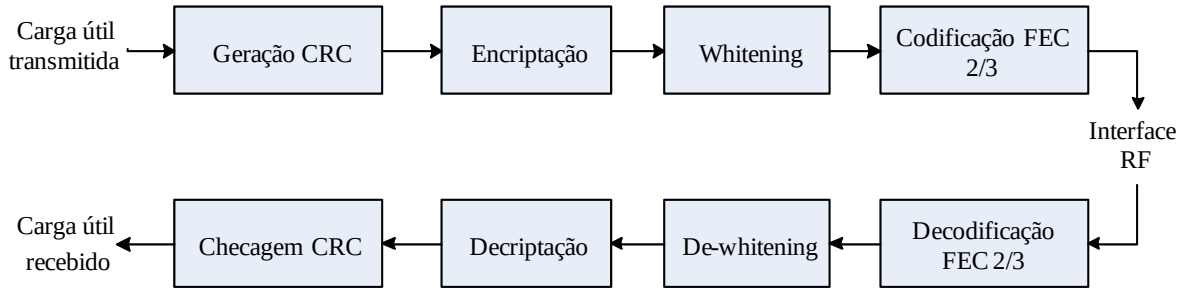

Figura 17 - Operações realizadas sobre a carga útil do pacote.

4.1.1.2 Implementação

A Figura 18 ilustra organização da implementação do adaptador de *Bluetooth*, hierarquicamente estruturada. Esta hierarquia obedece a quatro níveis de operações realizadas pela camada banda base. O nível inferior corresponde ao tratamento de *bits*; a seguir, o segundo nível aborda o tratamento de cadeias; o terceiro nível contém as operações de integração das cadeias de *bits* dentro da temporização do canal de comunicação *Bluetooth*; e o nível superior realiza os procedimentos de interface com as camadas superiores e com o transceptor de rádio.

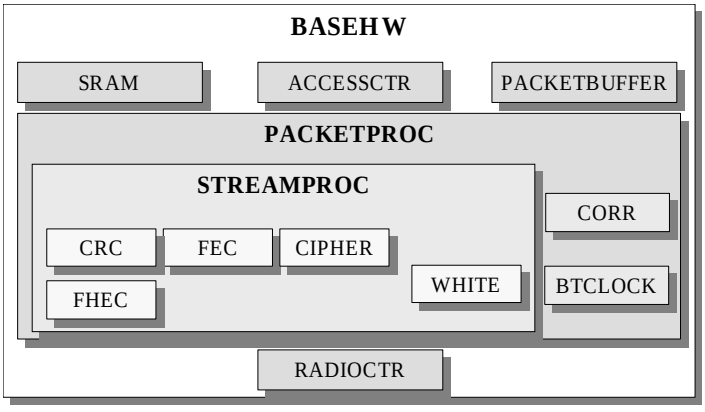

Figura 18 - Esquema da implementação da camada banda base.

O módulo que encerra a funcionalidade completa da implementação recebe o nome de BASEHW; este módulo comunica-se tanto com a camada de transmissão RF como com camadas superiores de nível lógico. Os módulos SRAM, ACCESSCTR, e PACKETBUFFER realizam, respectivamente, tarefas de armazenamento em

memória externa, a camada banda base, controle e transmissão de dados entre os módulos SRAM, PACKETBUFFER e RADIOCTR usando os protocolos certos, e armazenamento em memória interna do BASEHW.

A implementação do módulo de banda base inclui dois comportamentos diferentes, um para o modo de operação de transmissão da informação e outro no modo de informação que é passada pelo dispositivo mestre. A operação de transmissão se constitui em uma sequência de etapas e operações específicas. A configuração da transmissão é iniciada através do mecanismo seguinte:

1) Escrita, no módulo BASEHW, dos *bits* de carga útil, do cabeçalho, da configuração do canal e da configuração do transceptor de rádio.
2) Armazenamento da informação anterior no submódulo SRAM.
3) O submódulo ACCESSCTR transfere esta informação para a memória interna do módulo BASEHW, que se encontra no submódulo PACKETBUFFER.
4) Posteriormente, o submódulo RADIOCTL acessa o PACKETBUFFER e lê os valores da frequência de transmissão e do nível de potência a serem utilizados, além dos *bits* de configuração específicos para o transceptor de rádio.
5) O submódulo PACKETPROC acessa o PACKETBUFFER, para obter a configuração do enlace de *Bluetooth*; esta informação inclui os seguintes indicadores de condição:
    - o enlace está estabelecido, ou não.
    - o enlace é cifrado, ou não, determinado por um flag; em caso de cifragem, a chave de cifragem de 128 *bits* deve ser carregada.
    - o módulo deve atuar como mestre ou como escravo.
6) Uma vez que esta informação é obtida pelo PACKETPROC, este permite aos seus módulos internos acessarem o PACKETBUFFER para obter informações de configuração de cada um. O submódulo CORR lê do PACKEBUFFER a palavra de sincronização, enquanto o submódulo STREAMPROC lê a informação do tipo de pacote para determinar quais dos seus submódulos são necessários para a operação de transmissão.

7) Em seguida, o STREAMPROC gerencia a configuração destes submódulos, dando-lhes acesso ao PACKETBUFFER.

Depois do término da etapa de configuração, começa a etapa de transmissão:

1) Os pacotes *Bluetooth* passam por operações sobre *bits*, processados um de cada vez, para daí serem enviados pelo canal sem fio.
2) Durante a transmissão, o PACKETBUFFER fica ligado ao submódulo STREAMPROC e atua como fonte de dados para a operação.
3) Os pacotes são transmitidos da seguinte maneira:
    - primeiro, os bits da palavra de sincronização são enviados desde o submódulo CORR.
    - depois, os bits do cabeçalho do pacote são enviados pela saída do submódulo FHEC.
    - finalmente, os bits da carga útil são enviados desde o STREAMPROC.

Similar à transmissão, a operação de recepção é efetuada em uma sequência de etapas:

1) Escrita no módulo BASEHW da informação da configuração do canal e da configuração do transceptor de rádio.
2) Esta informação é transferida através do ACCESSCTR, para o PACKETBUFFER para que seja lida pelos demais submódulos do BASEHW.
3) O submódulo RADIOCTR lê a informação da configuração do transceptor de rádio, e, depois, o PACKETPROC lê a informação de configuração do enlace.
4) Quando o PACKETPROC tiver encerrado a fase de configuração, ele permite ao STREAMPROC acessar o PACKETBUFFER para transferir aos seus submódulos a informação de configuração; isso acontece unicamente no caso de o enlace estar previamente estabelecido (caso contrário, não é necessário).

A Figura 18 ilustra o STREAMPROC, módulo que engloba todo o processamento necessário para a construção e interpretação de pacotes *Bluetooth* (composto pelos submódulos FEC, FHEC, CRC, WHITE e CIPHER). Os submódulos que processam o cabeçalho e a carga útil não são os mesmos e os fluxos de informação são

aqueles ilustrados na Figura 16 e na Figura 17 respectivamente. As operações são realizadas tanto para a transmissão quanto para a recepção. O módulo do STREAMPROC, em linhas gerais, realiza as seguintes tarefas:

- CRC: Obtém o código de redundância cíclica da informação no pacote, com o objetivo de se detectar erros de transmissão ou recepção;

- FEC: Para cada 10 *bits* do pacote de dados, 5 outros são agrupados no FEC; os novos *bits* contêm informação necessária para a localização e correção de erro. A relação entre os tipos de *bits* é comumente expressa como FEC 2/3 (de três partes, duas são dados);

- FHEC: Efetua a validação do cabeçalho do pacote de dados;

- White: Altera os bits pseudoaleatoriamente, seguindo um certo padrão, para evitar-se longas sequências de bits repetidos, que poderiam prejudicar a transmissão;

- Cipher: Como medida de segurança, aplica um algoritmo de criptografia sobre os dados.

4.1.1.3 Verificação funcional

O módulo STREAMPROC foi escolhido como caso de teste da verificação funcional dentre os módulos do adaptador *Bluetooth*, pela sua razoável complexidade e engloba a composição de pacotes, que é uma das principais funcionalidades do adaptador. Para se ter idéia de sua complexidade, depois de sintetizado e implementado num FPGA Virtex-II da Xilinx (49), houve um consumo de 458 flip-flops, 1606 LUTs de quatro entradas e 891 faixas da FPGA. A verificação do módulo STREAMPROC foi realizada como parte do esquema de verificação hierárquica projetado para a verificação do adaptador de *Bluetooth*. Especificamente para a verificação do módulo STREAMPROC foram projetados dois *testbenches* para testar a construção e interpretação de pacotes. No presente estudo a análise foi centrada no *testbench* da composição de pacotes.

4.1.1.3.1 Transações

O módulo STREAMPROC opera no processamento de pacotes, portanto foi definido que na verificação das transações deveriam ser definidas neste nível. Para o *testbench* de composição de pacotes foi definida a transação com abstração de operação de composição de pacotes em função das suas entradas e saídas. A Tabela 9 descreve as variáveis de entrada e saída que definem a operação de composição de pacotes, para cada conjunto de valores de variáveis de entrada é obtido um pacote de *Bluetooth* na saída.

Tabela 9 - Transação para o *testbench* da verificação do módulo STREAMPROC

Entradas	
Variáveis	**Valores possíveis**
Tipo de pacote	{NULL, POLL, FHS, DM1, DH1, DM3, DH3, DM5, DH5}
Tamanho do pacote	{2-339 *bytes*}
Cifragem ativa	{ativa,não ativa}
Inicialização do embarulhamento	{0 - 2^{7}-1}
Inicialização do CRC	{0 - 2^{8}-1}
Relógio de *Bluetooth*	{0 - 2^{26}-1}
Endereço do mestre	{0 - 2^{48}-1}
Chave de segurança	{0 - 2^{128}-1}
Saídas	
Pacote de *Bluetooth*	{54 - 2745 bits}

4.1.1.3.2 Cobertura modular

Aproveitando a característica hierárquica da verificação do adaptador de *Bluetooth* e, uma vez que cada um dos cinco submódulos já havia sido verificado independentemente, o objetivo da verificação do modulo STREAMPROC foi, então, observar se a lógica de acoplamento estava correta e se os submódulos eram capazes de interagir apropriadamente. Para concretizar este objetivo de verificação o modelo de cobertura modular apresentado na página 13 foi utilizado. Este modelo de cobertura considera como eventos de cobertura os ciclos de relógio durante os quais cada uma das unidades funcionais está ativa; no contexto da verificação funcional do STREAMPROC, os seus submódulos correspondem às unidades funcionais como é apresentado a seguir:

- Evento 1: corresponde aos ciclos de relogio do submódulo FEC
- Evento 2: corresponde aos ciclos de relogio do submódulo FHEC

- Evento 3: corresponde aos ciclos de relogio do submódulo CIPHER
- Evento 4: corresponde aos ciclos de relogio do submódulo WHITE

O sub-módulo CRC não foi considerado porque ele está ativo em todos os bits do pacote e portanto não é necessário checar se esta operando. Para a verificação funcional do STREAMPROC, como alvo de cobertura foi definida a observação de 10.000.000 de amostras em cada evento de cobertura do modelo de cobertura modular, ou seja, tal número de ciclos ativos para cada submódulo. Esta quantidade de amostras foi escolhida dado que é bastante maior que o número médio de ciclos de cada transação.

4.2 Decodificador MPEG-4

O MPEG-4 é um padrão para o tratamento de vídeo de ISO/IEC (50). Este padrão define um pacote composto de 24 partes, especificando ferramentas e algoritmos para a codificação, decodificação e manipulação de conteúdo multimídia (51). A segunda parte do padrão descreve os algoritmos para o tratamento de áudio e vídeo digital visando alta compressão e portabilidade do conteúdo multimídia.

O padrão MPEG-4 tem uma estrutura orientada a objetos, que permite a decomposição de cenas de vídeo em múltiplos objetos de forma arbitrária, codificados de forma separada e eficiente. Segundo a especificação da segunda parte do padrão, o sistema MPEG-4 baseia-se em uma estrutura hierárquica de objetos, para assim, conseguir processar os dados com eficiência (50). Esta hierarquia é ilustrada pela Figura 19, sendo cada um dos objetos que compõem a estrutura hierárquica explicado resumidamente, a seguir:

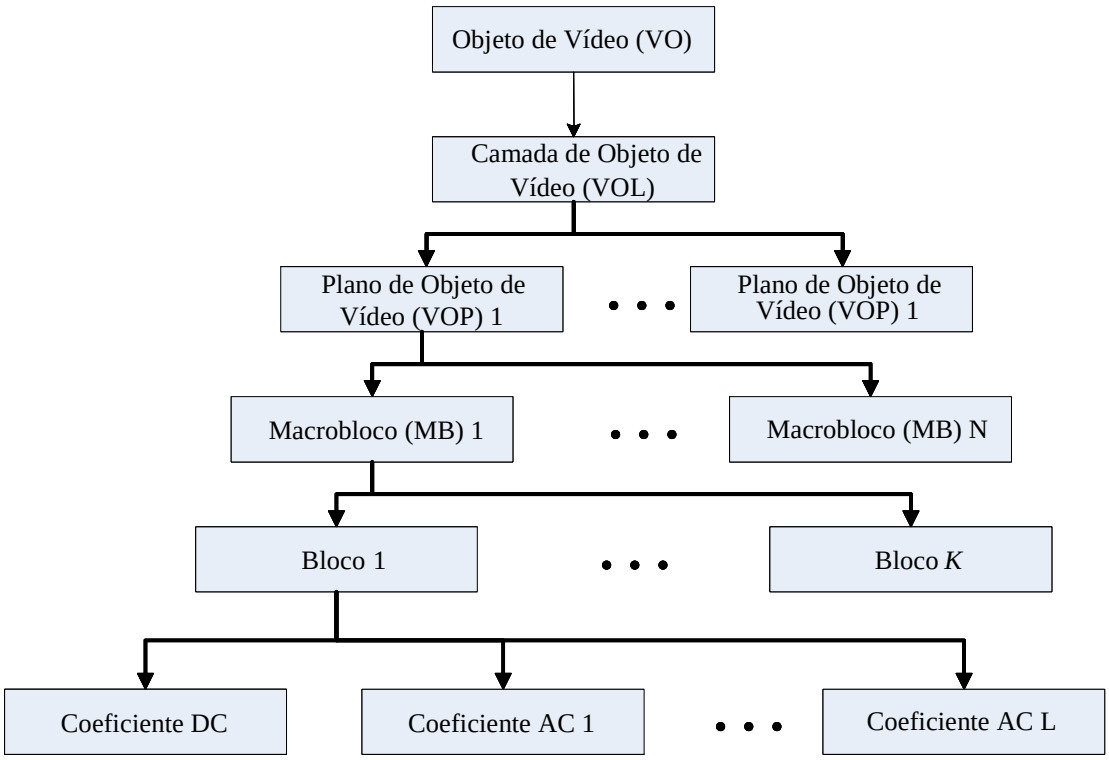

Figura 19 - Composição hierárquica dos objetos de vídeo MPEG-4 (51).

- Objeto de Vídeo (VO): um Objeto de Vídeo, ou Objeto Visual (VO), é uma sequência de representações gráficas bidimensionais, cada uma delas, em um ponto diferente no tempo. Após seu processamento, o VO é transmitido pelo codificador na forma de uma cadeia de *bits*, que inclui a correspondente informação de textura, movimento, e forma. As cadeias de *bits* de vários VOs podem ser multiplexadas, de forma que o decodificador receba toda a informação, convertendo-a, assim, em uma cena completa de vídeo decodificada.

- Camada de Objeto de Vídeo (VOL): camada de informações diversas como textura, movimento, forma, tamanho dos quadros de vídeo, sombras, transparências, etc., de vários VOPs que pertencem ao mesmo VO. A VOL foi projetada especificamente para aumentar a resolução temporal ou espacial de um VO, permitindo tornar os parâmetros anteriores escaláveis.

- Plano de Objeto de Vídeo (VOP): consiste em cada uma das representações gráficas das quais está composto um VO. Existem três tipos de VOPs dependendo do tipo de informação de codificação que possuam. Primeiro,

encontram-se os VOPs intra (I-VOPs), que são codificados sem nenhum tipo de referência a outras imagens; eles informam o ponto da sequência codificada onde a decodificação pode começar, mas contêm apenas uma taxa de compressão moderada. Em segundo lugar estão os VOPs preditivos (P-VOPs), codificados mais eficientemente usando predição de compensação de movimento a partir de um I-VOP ou um P-VOP anterior. Finalmente, os VOPs preditivos bidirecionais (B-VOPs) provêm o maior grau de compressão, mas precisam de VOPs de referência tanto prévios, quanto futuros.

- Macrobloco (MB): cada imagem (VOP) é dividida em vários macroblocos que, por sua vez, estão compostos por quatro blocos de luminância e dois blocos de crominância. O número de macroblocos dentro de um VOP depende do tamanho e da forma do VOP.

- Bloco: o bloco é a unidade de informação mínima do padrão MPEG-4 (com a exceção do pixel). Cada bloco é criado através de um processo de amostragem das imagens, e está constituído por oito colunas e oito fileiras de *pixels* para um total de 64 *pixels*, ou 64 *bits* de coeficientes da transformada discreta de cosseno (do inglês, *discrete cosine transform*, DCT) após o processo de codificação.

- Coeficientes DC e AC: Valores binários resultantes da aplicação da transformada de cosseno a cada bloco. A obtenção destes coeficientes é o ponto de partida para realizar todo o processo de compressão de vídeo MPEG-4.

O padrão MPEG-4 define dois modos fundamentais para codificar as sequências de imagens que compõem o vídeo a comprimir. O primeiro modo, denominado “intra” (referente à informação contida por dentro de cada imagem), aproveita a redundância espacial inerente a todo tipo de imagem digital. O modo “intra” realiza operações de compressão das cadeias de *bits* que compõem cada imagem ou plano individual do vídeo.

Adicionalmente, o padrão explora a redundância temporal na informação das formas (presente entre diferentes planos). Este esquema de codificação denomina-se modo “inter” e é baseado em algoritmos de compensação e predição com vetores de movimento. Desta forma, é possível comprimir sequências de vídeo excluindo-se

quadros não relevantes, e incluindo somente a informação relacionada às mudanças entre eles.

O decodificador foi implementado na linguagem SystemC no nível RTL, pelos integrantes do laboratório de arquiteturas dedicadas da Universidade Federal de Campina Grande (UFCG), durante a execução do projeto Brazil-IP (48). A implementação foi realizada para a decodificação de vídeos com resolução fixa de 176x144 *pixels*. A Figura 20 apresenta a organização do decodificador MPEG-4, podendo-se observar a separação das duas decodificações "inter" e "intra". Para os casos de estudo foram selecionados blocos representativos de cada um dos tipos de decodificação: no caso do tipo "inter" foi escolhido o módulo copiador de blocos (CBP) e no caso do tipo "intra" foi escolhido o módulo preditor de coeficientes AC e DC. Detalhes destes dois módulos são apresentados nas seções 4.2.1 e 4.2.2

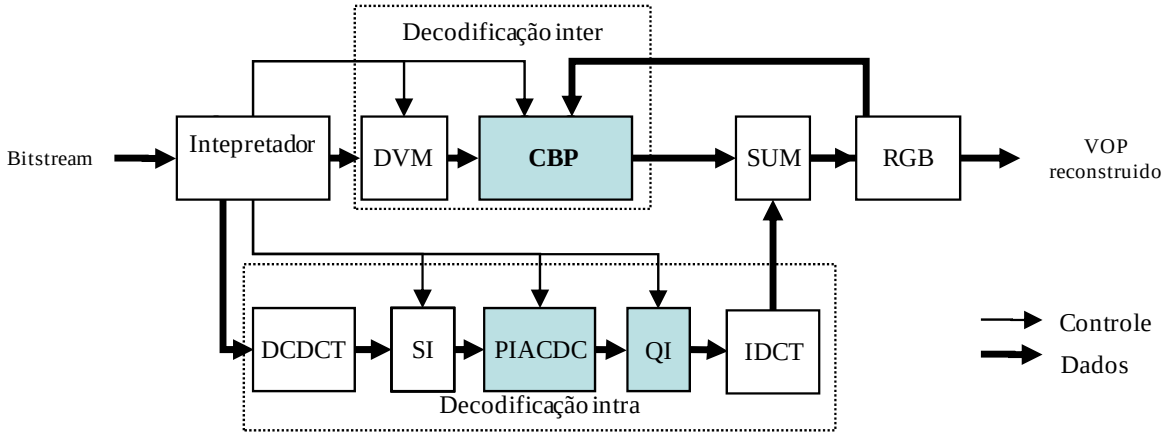

Figura 20 - Esquema de implementação do decodificador de MPEG-4

De forma geral os módulos do decodificador MPEG-4 realizam as seguintes tarefas:

- Interpretador das cadeias de dados: separa os diferentes VOPs e controla a decodificação.

- Decodificador de vetores de movimento (DVM): decodifica os vetores que contém os resíduos dos movimentos dos blocos pertencentes ao VOP.

- Decodificador de coeficientes da DCT (DCDCT): realiza as operações de decodificação dos coeficientes para a transformação de cosseno.

- Scan Inverso (SI): transforma a cadeia de coeficientes numa matriz, de acordo com o modo determinado pelo interpretador

- Preditor de coeficientes AC e DC (PIACDC): calcula os valores dos coeficientes dos blocos de acordo com os valores dos blocos vizinhos.

- Quantizador Inverso (QI): inverte a quantização feita nos valores de coeficientes.

- Transformada inversa de cosseno discreto (IDCT): realiza as operações de transformação de cosseno para o cálculo dos componentes espaciais dos *pixels* que formam os blocos.

- Compositor de quadros (CBP): compõe os planos do vídeo a partir dos planos anteriores, colocando de volta a redundância temporal.

- Somador (SUM): reúne a informação proveniente das decodificações temporais e espaciais.

- Formatador (RGB): converte os planos do formato YcrCb ao formato RGB.

4.2.1 Módulo copiador de blocos (CBP)

4.2.1.1 Funcionalidade

Para explicar a funcionalidade do módulo CBP é necessário apresentar como o padrão MPEG-4 realiza a codificação "inter". A codificação "inter" funciona a partir do principio de compensação de movimento entre dois planos consecutivos. Esta operação é realizada, procurando-se para cada macrobloco (MB) do plano, codificar um macrobloco semelhante no plano anterior. No caso de se achar um que seja coincidente, a diferença de posição é calculada e apresentada em forma de um vetor de movimento, com duas componentes nos eixos X e Y. O padrão permite que os deslocamentos indicados nos vetores de movimento tenham resoluções de um *pixel*, meio *pixel* e um quarto de *pixel*. Adicionalmente, durante a codificação MPEG-4, o bloco indica que para a codificação "inter' os vetores de movimento devem ser utilizados na compressão, com o valor médio em cada um dos componentes dos macroblocos adjacentes, acima e à direita, só permanecendo a diferença.

A funcionalidade específica do módulo CBP é a compensação temporal a partir dos vetores de movimento previamente decodificados no módulo DVM. Para isto, o CBP procura para cada macrobloco os valores dos *pixels* no plano anterior, com a ajuda da posição do macrobloco e o vetor de movimento. No caso de estimativas temporais de um meio e um quarto de *pixel*, o módulo CBP realiza também a interpolação necessária para determinar os valores dos *pixels*.

4.2.1.2 Verificação funcional

A verificação funcional do módulo CBP foi realizada com um *testbench* desenvolvido em SystemC pelos integrantes do laboratório de arquiteturas dedicadas da Universidade Federal de Campina Grande (UFCG). Este *testbench* tem como objetivo testar a funcionalidade descrita na página 64. Para o presente estudo foram realizadas algumas mudanças no *testbench* na definição das transações e no modelo de cobertura, sendo estas apresentadas a seguir.

4.2.1.2.1 Transações

Originariamente, o *testbench* do CBP tinha como escopo a verificação funcional de planos completos. Entretanto, no presente estudo foi considerado necessário obter maiores detalhes sobre as operações realizadas em cada macrobloco. Portanto, modificações no testbench foram feitas definindo-se como transações as operações realizadas em cada macrobloco, as quais, em função das variáveis de entrada e saída do módulo CBP, são apresentadas na Tabela 10.

Tabela 10 - Transação do *testbench* para o CBP

Entradas	
Variáveis	**Valores possíveis**
Decodificação intra (INTRA)	{ativa, não ativa}
Arredondar (RND)	{ativa, não ativa}
Vetor movimento componente X (MOVX)	{-64 a 64}
Vetor movimento componente Y (MOVY)	{-64 a 64}
Posição MB componente X (POSX)	{1 a 11}
Posição MB componente Y (POSY)	{1 a 9}
Plano anterior Y (Vetor de 256)	{0 a 2^8-1} [256]
Plano anterior U (Vetor de 256)	{0 a 2^8-1} [256]
Plano anterior V (Vetor de 256)	{0 a 2^8-1} [256]
Saídas	
Plano Y (Vetor de 256)	{0 a 2^8-1} [256]
Plano U (Vetor de 256)	{0 a 2^8-1} [256]
Plano V (Vetor de 256)	{0 a 2^8-1} [256]

4.2.1.2.2 Modelo de cobertura

A análise da funcionalidade do módulo CBP foi centrada nos diversos modos de operação, sendo observado que as principais mudanças no comportamento do DUV ocorrem quando os vetores de movimento apontam para fora da área do quadro de referência. Para permitir a checagem desta situação durante a execução do *testbench* foi montado um modelo de cobertura por itens com os seguintes eventos:

- Evento 1: Indica quando o componente no eixo X do endereçamento quadro de referência ultrapassa o limite inferior.
- Evento 2: Indica quando o componente no eixo X do endereçamento quadro de referência supera o limite superior.
- Evento 3: Indica quando o componente no eixo X do endereçamento quadro de referência está dentro dos limites do quadro anterior.
- Evento 4: Indica quando o componente no eixo Y do endereçamento quadro de referência ultrapassa o limite inferior.
- Evento 5: Indica quando o componente no eixo Y do endereçamento quadro de referência supera o limite superior.
- Evento 6: Indica quando o componente no eixo Y do endereçamento quadro de referência está dentro dos limites do quadro anterior.

4.2.2 Módulo preditor de coeficientes AC e DC (PIACDC)

4.2.2.1 Funcionalidade

A funcionalidade do módulo PIACDC faz parte do processo inverso à codificação "intra". Portanto um plano codificado no modo "intra" dever ter passado pelas seguintes tarefas:

1) Cálculo da transformada DCT bidimensional de cada bloco.
2) Quantização da matriz de coeficientes DCT de cada bloco. A quantização consiste em dividir, um por um, os coeficientes DCT entre os valores de uma matriz de quantização, e, como resultado, a maior parte dos quocientes dos coeficientes relacionados a cores de frequências altas será zero. Isto permitirá

reduzir significativamente a quantidade de dados de vídeo a processar pelo decodificador.

3) Reordenamento dos coeficientes quantificados utilizando algoritmos de mapeamento vertical, horizontal, ou zigue-zague. O resultado de tal reordenamento é um arranjo unidimensional de valores quantificados.
4) Aos coeficientes do bloco, é aplicada uma codificação diferencial; tabelas para realizar tal codificação encontram-se na especificação do padrão (50).
5) Finalmente é feita a codificado em relação ao número de zeros que antecede a cada coeficiente (codificação *Run-Length*).

A função do PIACDC é a operação inversa à descrita na tarefa número 4. Para a decodificação dos coeficientes é primeiro identificada a direção da predição, a qual é feita de acordo com os blocos apresentados na Figura 21A. A direção é obtida através do condicional apresentado na equação (37), onde DC_A, DC_B e DC_C são os coeficientes decodificados nos blocos A, B e C, respectivamente. Esta direção é aplicada para a predição do coeficiente DC e para os AC, se essa opção foi marcada durante a codificação.

$$\begin{aligned} &se\ \left(\left|DC_A - DC_B\right| < \left|DC_B - DC_C\right|\right) \\ &\quad \text{Predição do bloco C} \\ &se\ não \\ &\quad \text{Predição do bloco A} \end{aligned} \tag{37}$$

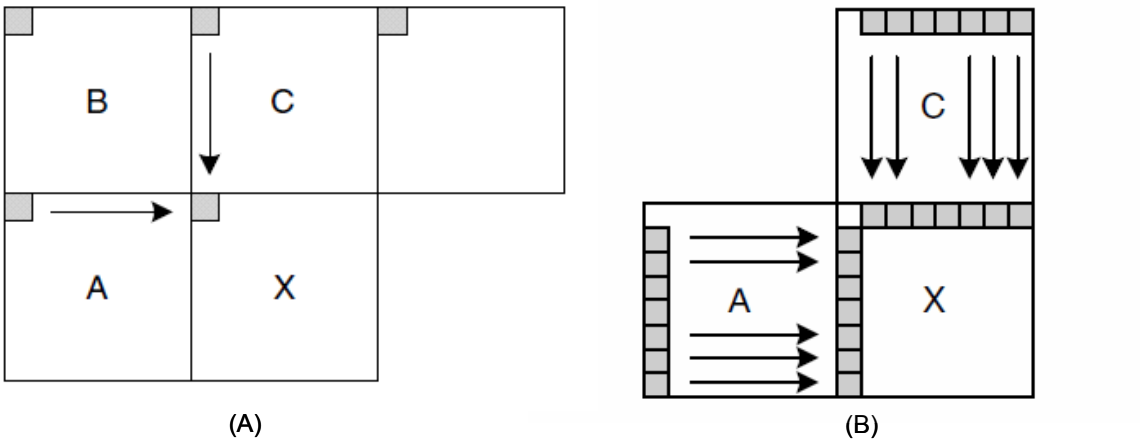

Figura 21 - Predição intra (A) coeficientes DC (B) coeficientes DC

Finalmente a decodificação da predição é realizada usando a equação (37), onde QDC_X é o coeficiente quantizado DC do bloco X, QDC_s indica o coeficiente

quantizado indicado pela equação (38), SF é um fator de escala fixo para cada quadro e CDC é valor a decodificar.

$$QDC_X = \frac{QDC_S}{SF} + CDC \quad (38)$$

A predição dos coeficientes AC utiliza a mesma direção da predição DC para indicar se coeficientes AC são decodificados por filas ou por colunas. A Figura 21 ilustra quais coeficientes são tomados como fonte para cada coeficiente decodificado. A decodificação dos coeficientes AC é realizada também usando a equação (38).

4.2.2.2 Verificação funcional

Na implementação do decodificador MPEG-4 utilizado neste trabalho o módulo PIACDC trabalha cooperativamente com o módulo QI e, por este motivo, os dois módulos compartilham o mesmo *testbench* o qual será denominado simplesmente de *testbench* PIACDC por facilidade.

4.2.2.2.1 Transações

Como no caso do módulo CBP, a verificação funcional no presente estudo foi reestruturada com algumas mudanças em relação à verificação feita pelos integrantes do laboratório de arquiteturas dedicadas da Universidade Federal de Campina Grande. Uma das principais mudanças no *testbench* do PIACDC foi na definição do escopo das transações, que passaram do escopo de blocos para o escopo de coeficientes. Para isto, cada transação foi definida como apresentado na Tabela 11. Na primeira coluna, as variáveis são listadas enquanto os seus valores são indicados na segunda coluna.

A Tabela 11 apresenta as variáveis de entrada e saída que definem uma operação do módulo PIACDC no escopo de coeficientes. No escopo de quadros, os valores das variáveis dos coeficientes DC_A e DC_C são determinados de acordo com os coeficientes previamente decodificados. No escopo de coeficientes, estas variáveis são controladas diretamente e para isto, foi necessário mudar a estrutura do *testbench*, acrescentado-se um *driver* que substitui a memória de armazenamento destes coeficientes.

Tabela 11 - Transação do *testbench* para o PIACDC

Entradas	
Variáveis	**Valores possíveis**
Decodificação intra (INTRA)	{ativa, não ativa}
Cabeçalho curto (SVH)	{não ativa}
Predição AC (AC_PRED)	{ativa, não ativa}
Coeficiente A (DC_A)	{-2048 a 2047}
Coeficiente C (DC_c)	{-2048 a 2047}
Coeficiente codificado (CDC)	{-2048 a 2047}
Coeficiente DC ou AC	{0,1}
Saídas	
Direção de predição	{0,1}
Coeficiente decodificado e dequantificado	{-2048 a 2047}

4.2.2.2.2 Modelo de cobertura

A verificação funcional sobre os módulos PIACDC e QI se concentrou na checagem dos valores de saída nas duas diferentes condições de direção e valores de saída para as decodificações AC e DC. Para isto foram utilizados os seguintes eventos de cobertura:

- Evento 1: indica a decodificação na direção de A.
- Evento 2: indica a decodificação na direção de C.
- Evento 3: indica a ocorrência de um coeficiente maior que zero na decodificação DC.
- Evento 4: indica a ocorrência de um coeficiente maior que zero na decodificação AC.
- Evento 5: indica a ocorrência de um coeficiente menor ou igual a zero na decodificação DC.
- Evento 6: indica a ocorrência de um coeficiente menor ou igual a zero na decodificação AC.

5 RESULTADOS

Nesta seção mostramos os resultados obtidos na aplicação das metodologias de geração de casos de teste ajustadas por cobertura aos módulos selecionados como casos de estudo. Devido às características particulares dos módulos STREAMPROC, CBP e PIACDC, os experimentos foram classificados em dois tipos, identificados como os casos com tamanho de espaço pequeno e espaço grande de entrada.

5.1 Método de experimentação

A avaliação no desempenho das técnicas de geração de casos de teste ajustada por cobertura foi realizada seguindo os passos abaixo:

1) Avaliação do tamanho do espaço de procura: neste passo, a quantidade de combinações de entrada do DUV que tem influência sobre os eventos de cobertura é analisada.
2) Estabelecimento das variações de tamanho dos conjuntos de treinamento: determina-se, neste passo, as variedades de tamanhos dos conjuntos de treinamento a serem usadas na avaliação da influência delas no comportamento dos *testbenches* com ajuste por cobertura. Quando o espaço de procura é demasiado grande para ser considerado completo é necessária uma análise estatística para se realizar este passo.
3) Estabelecimento das condições de aprendizado: determinam-se as técnicas de aprendizado e as suas possíveis variantes a serem aplicadas para a construção dos *testbenches* com geração de casos de teste ajustada por cobertura.
4) Experimentação: a metodologia apresentada na seção 3 é aplicada na construção dos *testbenches* necessários para avaliar a influência das variações de tamanho dos conjuntos de treinamento estabelecidas nas técnicas de aprendizado selecionadas.
5) Coleta de resultados: neste passo, os resultados das etapas de aprendizado e execução dos *testbenches* são colhidos e apresentados. Os resultados do aprendizado são apresentados em função dos erros médios de classificação e do tipo falso positivo, FP, e falso negativo, FN. Os resultados da execução

são apresentados com vista à redução de casos de teste e tempo de execução, com respeito aos *testbenches* sem ajuste.

6) Análise dos resultados: os resultados da execução são cruzados com as variações de tamanho nos conjunto de treinamento e técnica de aprendizado, para se tirar as conclusões do estudo.

5.2 Caso com espaço pequeno de entrada - módulo STREAMPROC

Os casos de estudo com espaço pequeno de entrada correspondem àqueles DUVs para os quais, a quantidade de combinações das entradas que afetam a ocorrência dos casos de cobertura é de um tamanho tal que permite que sejam rapidamente observadas no *testbench* um número de amostras superiores ao 1% do espaço. Este valor sinaliza que é possível coletar uma quantidade de amostras comparável com o tamanho do espaço de casos de teste. Dos módulos selecionados para casos de estudo, o módulo STREAMPROC apresenta esta característica.

5.2.1 Os experimentos

A seguir, os experimentos realizados com o módulo STREAMPROC são descritos, com os aspectos principais considerados.

5.2.1.1 Variações de tamanho nos conjuntos de treinamento

No módulo STREAMPROC, foi observado que o conjunto de entradas (apresentado na Tabela 9 da página 59) resulta num espaço com $1.28x10^{69}$ combinações possíveis de casos de teste. Mas também observa-se que só as entradas "tipo de pacote", "tamanho do pacote" e "cifragem ativa" têm efeito sobre o modelo de cobertura modular utilizado. Portanto o espaço de interesse para estas entradas é de 6102 combinações; na realidade, como muitos tipos de pacote não incluem todos os tamanhos ou opções de cifragem, o número de combinações válidas para os casos de teste é de 1819.

Como estamos tratando de um espaço de entrada de tamanho pequeno, podemos ter conjuntos de treinamento com alta representatividade dos parâmetros que têm influência sobre os eventos de cobertura. Para analisar a influência do tamanho do conjunto de treinamento foram estabelecidas três possíveis condições (subconjuntos do conjunto de todas as combinações possíveis):

- Conjunto pequeno: na condição denominada de conjunto pequeno, 20% das 1819 combinações totais dos casos de teste foram utilizadas para compor o conjunto de treinamento.

- Conjunto médio: nesta condição, 50% das combinações dos casos de teste foram utilizadas para compor o conjunto de treinamento.

- Conjunto grande: nesta condição, 80% das combinações dos casos de teste foram utilizadas para compor o conjunto de treinamento.

Quanto maior o conjunto de treinamento, maior é a quantidade de informação fornecida para o treinamento, portanto espera-se que o modelo aprendido seja também melhor, ou seja, quando usado na fase de classificação, gere decisões com menor taxa de erro.

A Tabela 12 apresenta a distribuição das categorias dos eventos de cobertura (na primeira coluna) para os conjuntos de treinamento utilizados durante a avaliação do módulo de STREAMPROC. A categorização, descrita em na página 45, é feita considerando como categoria associada a cada evento o fato ter o maior número de ciclos de relógio na transação. A razão para isto é que adotamos, para este caso, tornando a cobertura modular (página 59), excludentes entre as suas categorias. As categorias estão enumeradas, a partir do número 1, representando, respectivamente, os sub-módulos FEC, FHEC, CIPHER e WHITE. Nas colunas seguintes, são apresentadas as porcentagens de exemplos de treinamento correspondentes a cada categoria em cada conjunto de treinamento; as colunas de 2 a 5 referem-se, respectivamente aos casos de conjunto pequeno, médio, grande e conjunto total. O conjunto total corresponderia ao caso de 100%, ou seja, ao espaço total de combinações válidas.

Tabela 12 - Distribuição das categorias nos conjuntos de treinamento

Categorias	C. pequeno (%)	C. médio (%)	C. grande (%)	C. completo (%)
1	28,10	22,33	25,77	25,29
2	0,55	0,11	0,27	0,22
3	43,25	44,33	43,51	44,58
4	28,10	30,25	30,45	29,91

5.2.1.2 Técnicas de aprendizado

Para a avaliação segundo a técnica de aprendizado, foram desenvolvidos *testbenches* com geração ajustada com classificadores baseados nas técnicas de *data mining*, redes Bayesianas e SVM. Os modelos foram construídos usando as seguintes ferramentas:

- A implementação do algoritmo C4.5 da ferramenta WEKA (52) foi utilizada para construir o modelo de *data mining*.
- A implementação do algoritmo EM da ferramenta GENIE (53) foi utilizada para construir o modelo de redes Bayesianas.
- A implementação do algoritmo SMO foi implementada no GSEIS do LME-USP (54).

As técnicas de aprendizado foram aplicadas com as suas configurações padrão, fornecidas pelas ferramentas; no caso do SVM, foram utilizados parâmetros típicos e, especificamente para o kernel, foi adotado o modelo Gaussiano com parâmetro C=10.

5.2.1.3 Os *testbenches*

O comportamento do *testbench* com ajuste por cobertura, para o caso do módulo STREAMPROC, foi acompanhado em cada técnica de aprendizado com as três variedades de tamanho de conjunto de treinamento, resultando em 9 situações diferentes. A Tabela 13 lista na primeira coluna os *testbenches* montados para os experimentos, com os seus códigos de identificação. As características do testbench em termos de tamanho do conjunto de treinamento e da técnica de aprendizado utilizada, são dadas nas colunas 2 e 3, respectivamente. O primeiro *testbench*, em particular, trata do caso aleatório, sem ajuste por cobertura, o qual serve de referência de comparação para os demais. Para todos os *testbenches* foi especificado como meta, um número de 10.000.000 amostras por cada evento de cobertura, ou seja, para este caso de cobertura modular do STREAMPROC, 10.000.000 de ciclos ativos para cada submódulo. Este valor foi escolhido porque permite que todas as combinações dos parâmetros de interesse são contempladas.

Tabela 13 - Listado de *testbenches* realizados para o STREAMPROC

Código de identificação	Condição de tamanho	Técnica de aprendizado
TB_SP_SEM_AJUSTE		
TB_SP_1	Conjunto pequeno	*Data mining*
TB_SP_2	Conjunto médio	*Data mining*
TB_SP_3	Conjunto grande	*Data mining*
TB_SP_4	Conjunto pequeno	Redes Bayesianas
TB_SP_5	Conjunto médio	Redes Bayesianas
TB_SP_6	Conjunto grande	Redes Bayesianas
TB_SP_7	Conjunto pequeno	SVM
TB_SP_8	Conjunto médio	SVM
TB_SP_9	Conjunto grande	SVM

Os experimentos realizados com módulo STREAMPROC foram executados num computador com as seguintes configurações: processador AMD Athlon 2800+, com quatro Gigabytes de memória RAM do tipo DDR e executado no sistema operacional Linux OpenSUSE 9.

5.2.2 Resultados do treinamento e análise

Como foi apresentado nas páginas 46 e 49, o processo de aprendizado gera como resultado, modelos que representam a relação entre os casos de teste e as categorias associadas aos eventos de cobertura. Nesta sub-seção, apresentaremos, preliminarmente, uma avaliação da exatidão destes modelos gerados em função dos erros de classificação, de FP e de FN, sendo realizada uma análise segundo as observações apresentadas na página 41. Para o módulo STREAMPROC, esta avaliação é feita tomando como conjunto de classificação o espaço completo de casos de teste avaliados no *testbench*.

Os resultados de erro de classificação são apresentados na Tabela 14, mostrando na primeira coluna as variações dos conjuntos de treinamento e, na segunda, terceira e quarta colunas, as porcentagens totais de erros para os modelos construídos com as técnicas de *data mining*, redes Bayesianas e SVM, respectivamente.

Tabela 14 - Erro de classificação dos modelos de aprendizado do STREAMPROC

Condição	*Data mining* (%)	Redes Bayesianas (%)	SVM (%)
C. pequeno	0,44	33,75	0,99
C. médio	0,16	21,94	0,33
C. grande	0,06	13,03	0,27

Os erros apresentados na Tabela 8 são detalhados para cada categoria dos eventos de cobertura, através do cálculo dos erros de FP e FN, os quais são apresentados na Tabela 15. Ela apresenta na primeira coluna a variedade de conjunto de treinamento, enquanto, na segunda, os casos de classificação para os quais a categoria de evento de cobertura é apontada como resultante; nas terceira e quarta colunas, os erros de FP e FN para a técnica de *data mining* são mostrados; o mesmo ocorre nas quinta e sexta colunas, para a técnica redes Bayesianas, e nas sétima e oitava colunas, para a técnica de SVM.

Tabela 15 - Erros de falso positivo e falso negativo do STREAMPROC

Condição	**Cat.**	***Data mining***		**Redes Bayesianas**		**SVM**	
		F. P. (%)	**F. N. (%)**	**F. P. (%)**	**F. N. (%)**	**F. P. (%)**	**F. N. (%)**
C. pequeno	1	0,00	0,00	50,21	49,57	0,22	1,71
	2	50,00	66,67	97,85	50,00	100,00	100,00
	3	0,00	0,12	18,06	19,98	1,60	0,00
	4	0,74	0,55	31,26	40,88	0,00	0,00
C. médio	1	0,00	0,00	22,66	28,04	0,65	0,00
	2	75,00	0,00	99,19	75,00	75,00	0,00
	3	0,00	0,00	12,90	20,10	0,00	0,12
	4	0,00	0,55	16,06	19,15	0,00	0,37
C. grande	1	0,00	0,00	9,49	10,87	0,65	0,00
	2	0,00	20,00	95,45	0,00	50,00	0,00
	3	0,00	0,00	7,58	14,30	0,00	0,12
	4	0,00	0,00	10,10	13,08	0,00	0,37

Na Tabela 14 observa-se que os modelos obtidos com o treinamento com redes Bayesianas apresentam o maior erro, seguidos pelos da SVM e depois pelos da técnica de *data mining*. Também, pode-se observar que o erro diminui à medida que o conjunto de treinamento aumenta, como seria de se esperar, uma vez que trabalha-se com mais dados no treinamento. Nos dados detalhados do erro de FN e FP, apresentados na Tabela 15, é observado que as porcentagens maiores se apresentam na categoria de evento 2, mas tal fato é devido à sua pequena participação no espaço de casos de teste (ver distribuição na Tabela 12). Por outro lado, tanto para a *data mining* e como para a SVM, as porcentagens são baixas nos eventos diferentes a 2. As porcentagens apresentadas na tabela mostram que os erros são maiores para as redes Bayesianas e estas tendem a diminuir quando o conjunto de treinamento aumenta. O pior desempenho das redes Bayesianas pode ser explicado pelo fato de elas não considerarem a continuidade das variáveis de

natureza numérica, requerendo um grande número de amostras no treinamento para representá-las apropriadamente, como visto na página 32.

5.2.3 Resultados dos *testbenches*

A geração de casos de teste ajustada por cobertura tem como consequência a redução do tempo de execução dos *testbenches*, quando comparado a *testbenches* tradicionais puramente aleatórios, já que são necessários menos casos de teste para se obter 100% de cobertura. Os resultados da aplicação das técnicas SVM, redes Bayesianas e *data mining* nos *testbenches* ajustados por cobertura com o módulo STREAMPROC (como previsto na seção 4.1) são apresentados na Tabela 16. Nela são listados na primeira coluna os *testbenches* construídos com a inclusão dos modelos obtidos a partir dos diferentes conjuntos de treinamentos. Na segunda coluna são apresentadas as quantidades de casos de teste gerados e aplicados em cada *testbench* para se cumprir a meta de cobertura definida na página 73, enquanto, na terceira coluna são apresentados os tempos de execução correspondentes. Na quarta coluna são apresentadas as reduções de quantidade de casos de teste com respeito ao *testbench* tradicional aleatório, sem ajuste, enquanto, na quinta coluna, são apresentadas as reduções de tempo correspondentes com respeito ao *testbench* sem ajuste.

Tabela 16 - Resultados de redução de tempo e casos de teste

Testbench	**Casos de teste**	**Tempo (s)**	**Redução casos (%)**	**Redução tempo (%)**
Testbench sem ajuste	13.868	287.858		
SVM (Pequeno)	8.890	218.308	36,00	24,00
SVM (Médio)	9.452	226.308	32,00	21,00
SVM (Grande)	9.510	229.466	31,00	20,00
Redes Bayesianas (Pequeno)	-	-	-	-
Redes Bayesianas (Médio)	9.824	258.648	29,00	10,00
Redes Bayesianas (Grande)	9.639	258.612	30,00	10,00
Data Mining (Pequeno)	9.203	223.094	33,64	22,50
Data Mining (Médio)	8.775	203.565	36,72	29,28
Data Mining (Grande)	8.851	204.875	36,18	28,83

O uso de redes Bayesianas no direcionamento da geração de estímulos, sob a condição de conjunto *pequeno*, não permitiu o término da execução do *testbench*, uma vez que o ajuste na geração levou à aplicação de testes não válidos; este fato explica-se pelo alto erro de classificação mostrado na página 74, que levou ao fim da execução do *testbench* antes de se atingir 100% de cobertura.

Pode-se observar na Tabela 16 que os resultados de redução de casos de teste e de tempo de execução apresentam resultados semelhantes em cada uma das técnicas, mesmo em condições de treinamento diferentes, ainda que os resultados sejam mais próximos para as técnicas de *data mining* e SVM. Isto mostra que as reduções observadas são mais dependentes da capacidade de aprendizado da técnica do que no tamanho conjunto de treinamento.

Comparativamente os resultados com as diferentes técnicas mostram que os melhores resultados foram obtidos com a técnica de *data mining*, enquanto a SVM apresentou resultados intermediários e a técnica com redes Bayesianas mostrou os piores resultados. Tal resultado é compatível com os erros de classificação observados nas tabelas da página 74, em que o treinamento com a SVM apresenta erros de FP maiores os obtidos com a técnica de *data mining*, fato que afeta negativamente a execução dos *testbenches* com SVM com respeito aos com *data mining*. Outro aspecto que reduz o desempenho no caso da SVM é que ela apresenta um processamento extra devido ao esquema de filtragem (página 50), já que é necessário um tempo para geração aleatória, o qual é aproveitado unicamente quando ocorra o caso de teste requerido para o evento de cobertura focado; portanto, se o evento for incomum, poderá ser preciso gerar um maior número de casos de teste até se obter aquele de interesse.

5.2.4 Análise geral dos resultados

Os resultados do treinamento mostraram que, no caso de estudo STREAMPROC, as técnicas de aprendizado de *data mining* e SVM conseguiram produzir modelos de *testbenches* reativos com baixos níveis de erro. Adicionalmente foi observado que os *testbenches* tiveram sucesso na redução do tempo de execução e da quantidade de casos teste necessários para se alcançar a meta de cobertura estabelecida, quando comparados a resultados de execução de *testbenches* tradicionais sem ajuste por cobertura.

Para a técnica redes Bayesianas, foi observado que os níveis de erro tiveram uma forte dependência do tamanho do conjunto de treinamento. No caso de conjunto de tamanho pequeno, foram obtidos altos níveis de erro, os quais fizeram com que o *testbench* nesta condição não fosse executado com sucesso. Já nas condições de conjunto médio e grande, o erro menor fez com que os *testbenches* tivessem

reduções de tempo e casos de teste, porém com valores inferiores aos observados com as técnicas de *data mining* e SVM.

5.3 Casos com espaço de entrada grande

Diferente do caso apresentado na seção 5.2, os casos de estudo com espaço grande de parâmetros de entrada cobrem aqueles DUV's em que há uma grande quantidade de combinações de entradas que afetam a ocorrência dos eventos de cobertura. Diferente da seção 5.2, trata-se de um espaço, cujo tamanho tal que não permite que sejam computadas rapidamente quantidades de amostras superiores ao 1% do tamanho do espaço de entrada. Neste trabalho de tese, os módulos CBP e PIACDC apresentam esta característica; como apresentado na seção 4.2, estes dois módulos são representativos do domínio aplicação de sistemas multimídia.

5.3.1 Caso de estudo módulo CBP

5.3.1.1 Os experimentos

A presente seção descreve os aspectos relevantes da construção dos *testbenches* e a formulação da análise experimental.

5.3.1.1.1 Categorização dos eventos de cobertura

Como foi apresentado na página 45, o treinamento requer uma categorização dos eventos de cobertura para garantir que trabalhemos com classes que sejam excludentes. Para a aplicação das técnicas de geração com ajuste por cobertura no módulo CBP, a ocorrência dos eventos foi categorizada considerando-os como uma variável binária de 6 dígitos, onde o evento 1 corresponde ao dígito menos significativo e, assim seguidamente, até a ocorrência do evento 6 que corresponde ao dígito mais significativo. Nesta codificação, a categoria 9 (=’001001’) significa a ocorrências dos eventos 1 e 4. Por esta variável foi observado que unicamente as categorias 9, 10, 12, 17, 18, 20, 33, 34 e 36 tiveram amostras nos conjuntos de treinamento.

5.3.1.1.2 Variações de tamanho nos conjuntos de treinamento

Para a realização dos experimentos com o módulo CPB, primeiramente foram analisadas as entradas do módulo e foi identificada a existência de $1{,}788 \times 10^{12}$ combinações possíveis. Dado a que a entrada "INTRA" indica que o módulo é

inabilitado, então o total de combinações válidas é 8,504x10^{11}. Mesmo com esta redução, o alto número de combinações válidas fez que fosse necessária uma análise estatística para se determinar o tamanho mínimo do conjunto de treinamento para se ter um aprendizado significativo. Tal análise foi realizada a partir do trabalho de Chaudhuri (55), no qual especifica-se que a quantidade de amostras, necessárias para refletir a distribuição do espaço, é calculada com a equação (39), onde *r* representa o tamanho da amostra, *n* representa o tamanho do espaço, *k* representa o número de faixas do histograma, $1-\gamma$ representa a probabilidade das amostras terem um erro *f*. A quantidade de amostras calculada para o conjunto de treinamento do módulo CBP foi de 80.922, considerando-se *k* =200 γ=0,01 e *f* =0,15. Com o objetivo de aumentar a confiabilidade dos resultados, os experimentos foram duplicados, gerando-se dois conjuntos diferentes de amostras, porém do mesmo tamanho (o que ficará explícito na Tabela 17).

$$r \geq \frac{4k \ln\left(\frac{2n}{\gamma}\right)}{f^2} \tag{39}$$

Para os experimentos do CPB foram consideradas as seguintes condições básicas:

- Conjunto médio: nesta variedade, foi considerado que o número de casos de teste do conjunto de treinamento fosse igual à quantidade de amostras calculada em (16).

- Conjunto grande: variedade de conjunto de treinamento com número de casos de teste igual ao dobro da quantidade de amostras calculada.

A seguir foi verificada a necessidade de abertura de novas variações nos conjuntos de treinamento, pois, após uma avaliação da cobertura nos dois tipos de conjuntos descritos acima, foi observado que a distribuição, apresentada na Tabela 17, dos eventos de cobertura executados pelo *testbench* do CBP estavam muito focados na categoria 36 considerada dominante. Nesta tabela observam-se na primeira coluna as categorias utilizadas no *testbench* do CBP com ajuste, na segunda e terceira colunas se observam as distribuições para dois conjuntos de tamanho médio (como forma de aumentar a confiabilidade), e na quarta e quinta colunas se observam as distribuições, também, para os dois conjuntos de tamanho grande. Em todos os conjuntos observa-se que existe a mesma tendência de dominância da categoria 36.

Tabela 17 - Distribuição dos eventos de cobertura para o módulo CBP

Categorias	C. médio 1 (%)	C. médio 2 (%)	C. grande 1 (%)	C. grande 2 (%)
9	0,13	0,14	0,14	0,15
10	0,04	0,05	0,05	0,05
12	4,03	4,28	4,04	4,26
17	0,05	0,07	0,05	0,06
18	0,02	0,02	0,01	0,02
20	1,30	1,60	1,34	1,58
33	3,18	3,46	3,26	3,45
34	1,13	1,28	1,12	1,26
36	90,11	89,11	89,98	89,17

Portanto, foi conjecturado que seria positivo compensar a forte diferença na distribuição das categorias, pois seria capaz de evitar que a categoria dominante mascarasse a presença das outras nos modelos resultantes da aprendizagem. No presente estudo, foram propostas então, adicionalmente, as seguintes quatro condições de treinamento:

- Conjunto médio balanceado a 50%: nesta condição, a categoria majoritária do conjunto médio é reduzida até ficar com 50% do tamanho. As outras categorias compõem proporcionalmente os outros 50%.
- Conjunto médio balanceado a 70%: a categoria majoritária do conjunto médio é reduzida até ficar com 70% do tamanho. As outras categorias compõem proporcionalmente os outros 30%.
- Conjunto grande balanceado a 50%: a categoria majoritária do conjunto grande é reduzida até ficar com 50% do tamanho. As outras categorias compõem proporcionalmente os outros 50%.
- Conjunto grande balanceado a 70%: a categoria majoritária do conjunto grande é reduzida até ficar com 70% do tamanho. As outras categorias compõem proporcionalmente os outros 30%.

5.3.1.1.3 Técnicas de aprendizado

Assim como na seção 5.2, no caso de estudo com o módulo CBP considerou-se a aplicação das técnicas de *data mining*, redes Bayesianas e SVM para o aprendizado das características dos conjuntos de treinamento, e em condições similares. O aprendizado foi realizado com as mesmas ferramentas WEKA e GENIE, na implementação dos algoritmos para a modelagem nas técnicas de *data mining* e

redes Bayesianas, porém para o primeiro, ambas a estratégias de “com corte” e “sem corte” foram avaliadas. Devido à velocidade necessária para o treinamento, diferente do realizado na página 78, a ferramenta WEKA foi utilizada também para o treinamento na SVM e na implementação do classificador no *testbench*. As técnicas de aprendizado foram utilizadas com as suas configurações padrões, fornecidas pelas ferramentas, como já citado anteriormente.

5.3.1.1.4 Os *testbenches*

Os experimentos com o CBP foram realizados com os *testbenches*, cujos códigos de identificação são apresentados na primeira coluna da Tabela 18. Pela coluna 2, os *testbenches* estão discriminados pelo tamanho dos conjuntos de treinamento, i.e., número de casos de teste (médio e grande), assim como pelas combinações utilizadas de casos de teste, seguindo as proporções apresentadas em na página 78. Além disso, os *testbenches* são discriminados pela técnica de aprendizado, como indicado na coluna 3. Nesta tabela, para cada condição com determinado tamanho do conjunto de treinamento e técnica de aprendizado, há dois *testbenches* diferentes com conteúdos dos conjuntos de treinamento diferentes, como forma de aumentar a confiabilidade; por exemplo, os *testbenches* identificados com TB_CBP1 e TB_CBP2 foram criados com modelos treinados com conjuntos de tamanho médio e a técnica de *data mining*, porém os casos de teste que os conformam são diferentes.

Para todos os *testbenches* do módulo CBP foi definido como meta de cobertura a observação de 10.000 ocorrências para cada um dos eventos de cobertura descritos na página 66. Este valor foi escolhido após se proceder a execução do *testbench* com geração aleatória, sem ajuste para a cobertura, observando-se o número de casos de teste necessários para atingir as metas individuais dos diversos eventos de cobertura (ver Tabela 19). Como o número total de casos de teste possíveis é muito alto, há a necessidade de se ter uma verificação estatisticamente confiável. Sabe-se que quanto menor a meta de cobertura estabelecida, menos casos de teste são necessários e mais distorção pode haver nos resultados. Por isto, esta avaliação foi considerada importante.

Tabela 18 - Listado de *testbenches* realizados para o CBP

Código de identificação	Variedade de tamanho	Técnica de aprendizado
TB_CBP_1	Conjunto médio	*Data mining*
TB_CBP_2	Conjunto médio	*Data mining*
TB_CBP_3	Conjunto grande	*Data mining*
TB_CBP_4	Conjunto grande	*Data mining*
TB_CBP_5	Conjunto médio	*Data mining* sem corte
TB_CBP_6	Conjunto médio	*Data mining* sem corte
TB_CBP_7	Conjunto grande	*Data mining* sem corte
TB_CBP_8	Conjunto grande	*Data mining* sem corte
TB_CBP_9	Conjunto médio	Redes Bayesianas
TB_CBP_10	Conjunto médio	Redes Bayesianas
TB_CBP_11	Conjunto grande	Redes Bayesianas
TB_CBP_12	Conjunto grande	Redes Bayesianas
TB_CBP_13	Conjunto médio	SVM
TB_CBP_14	Conjunto médio	SVM
TB_CBP_15	Conjunto grande	SVM
TB_CBP_16	Conjunto grande	SVM
TB_CBP_17	Conjunto médio balanceado 50	*Data mining*
TB_CBP_18	Conjunto médio balanceado 50	*Data mining*
TB_CBP_19	Conjunto médio balanceado 50	*Data mining*
TB_CBP_20	Conjunto médio balanceado 50	*Data mining*
TB_CBP_21	Conjunto médio balanceado 50	*Data mining* sem corte
TB_CBP_22	Conjunto médio balanceado 50	*Data mining* sem corte
TB_CBP_23	Conjunto médio balanceado 50	*Data mining* sem corte
TB_CBP_24	Conjunto médio balanceado 50	*Data mining* sem corte
TB_CBP_25	Conjunto médio balanceado 50	Redes Bayesianas
TB_CBP_26	Conjunto médio balanceado 50	Redes Bayesianas
TB_CBP_27	Conjunto médio balanceado 50	Redes Bayesianas
TB_CBP_28	Conjunto médio balanceado 50	Redes Bayesianas
TB_CBP_29	Conjunto médio balanceado 50	SVM
TB_CBP_30	Conjunto médio balanceado 50	SVM
TB_CBP_31	Conjunto médio balanceado 50	SVM
TB_CBP_32	Conjunto médio balanceado 50	SVM
TB_CBP_33	Conjunto médio balanceado 70	*Data mining*
TB_CBP_34	Conjunto médio balanceado 70	*Data mining*
TB_CBP_35	Conjunto médio balanceado 70	*Data mining*
TB_CBP_36	Conjunto médio balanceado 70	*Data mining*
TB_CBP_37	Conjunto médio balanceado 70	*Data mining* sem corte
TB_CBP_38	Conjunto médio balanceado 70	*Data mining* sem corte
TB_CBP_39	Conjunto médio balanceado 70	*Data mining* sem corte
TB_CBP_40	Conjunto médio balanceado 70	*Data mining* sem corte
TB_CBP_41	Conjunto médio balanceado 70	Redes Bayesianas
TB_CBP_42	Conjunto médio balanceado 70	Redes Bayesianas
TB_CBP_43	Conjunto médio balanceado 70	Redes Bayesianas
TB_CBP_44	Conjunto médio balanceado 70	Redes Bayesianas
TB_CBP_45	Conjunto médio balanceado 70	SVM
TB_CBP_46	Conjunto médio balanceado 70	SVM
TB_CBP_47	Conjunto médio balanceado 70	SVM
TB_CBP_48	Conjunto médio balanceado 70	SVM

A Tabela 19 ilustra algumas porcentagens de casos de teste para cada categoria com respeito à última em atingir a meta de observação. Por exemplo, com a definição da meta de cobertura de 1.000.000 de itens para cada evento, o evento 2

corresponde ao último a completar a meta, enquanto o evento 1 leva 99,3% e o evento 3, 1,9%, do mesmo montante de casos de teste para atingirem a meta. A meta de 10.000 ocorrências foi adotada por ser a menor entre as observadas que mostrava sinais de estabilidade nas porcentagens de arribo dos eventos de cobertura às suas respectivas metas.

Os experimentos do CPB foram executados numa estação de trabalho com um processador Intel CORE2DUO, quatro Gigabytes de memória DDR-2 e executando o sistema operacional Linux OpenSUSE 10.3.

Tabela 19 - Porcentagem de casos de teste gastos para atingir os alvos de cobertura

Alvo	Evento 1 (%)	Evento 2 (%)	Evento 3 (%)	Evento 4 (%)	Evento 5 (%)	Evento 6 (%)
1000	92,42	100	2,30	90,12	76,60	5,19
10000	95,37	100	1,90	88,20	78,20	2,04
1000000	99,30	100	1,90	81,59	80,33	1,93

Para efeitos de simplificar a apresentação dos resultados, os valores apresentados nas seções seguintes correspondem ao valor médio dos obtidos com modelos e *testbenches*, desenvolvidos e executados nas mesmas condições de tamanho e técnica de aprendizado; por exemplo, os resultados dos modelos e *testbenches* indicados como de conjunto médio e *data mining* correspondem ao valor médio dos resultados associados aos *testbenches* TB_CBP_1 e TB_CBP_2.

5.3.1.2 Resultados do treinamento

Os resultados com as diversas variedades de conjuntos de treinamento, para cada uma das técnicas de aprendizado, são apresentadas nesta seção, tendo como objetivo fazer a classificação de um conjunto-exemplo padrão de casos de teste. Tal conjunto padrão é composto por 323.689 casos de teste e suas respectivas categorias de cobertura. As técnicas de treinamento de *data mining*, *data mining* sem corte, redes Bayesianas e SVM tiveram seus comportamentos avaliados a partir do erro total acumulado obtido na classificação do conjunto padrão e do erro de FP e FN nas categorias de classificação.

A Tabela 20 apresenta os erros de classificação dos modelos do módulo CBP e está organizada da seguinte forma: a primeira coluna apresenta a variedade do conjunto de treinamento utilizado no aprendizado, a segunda, terceira e quarta colunas

apresentam as porcentagens de erro dos modelos aprendidos com as técnicas de *data mining*, *data mining* sem corte, redes Bayesianas e SVM.

Tabela 20 - Erro de classificação dos modelos de aprendizado do CBP

Condição	*Data mining* (%)	*Data mining* sem corte (%)	Redes Bayesianas (%)	SVM (%)
C. médio	10,42	10,35	10,42	11,40
C. grande	10,42	10,39	10,42	12,14
C. médio bal 70	16,72	18,20	10,42	16,63
C. grande bal 70	16,84	17,23	10,42	17,00
C. médio bal 50	28,10	28,07	10,52	15,73
C. grande bal 50	28,66	28,75	10,55	16,98

A Tabela 21 apresenta detalhadamente os erros de FP e FN para cada uma das categorias de cobertura, tendo sido calculados da maneira apresentada na página 40. A Tabela 21 está organizada da seguinte forma: a primeira coluna apresenta a variedade de conjunto de treinamento utilizada, a segunda coluna apresenta as categorias de cobertura apontadas pelos modelos de aprendizado quando da classificação com os casos de teste, a terceira e quarta colunas apresentam, respectivamente, a porcentagem de erro de FPs e FNs para a técnica de *data mining*; o mesmo padrão é seguido nas colunas 5/6, 7/8 e 9/10, para as técnicas de *data mining* sem corte, redes Bayesianas e SVM, respectivamente.

Durante o cálculo dos erros de FP, foi possível observar que, em alguns modelos, este valor não foi possível de se determinar, dado que os modelos gerados pelo treinamento não conseguiram classificar nenhum caso de teste dentro de determinada categoria existente. Estes dados indicam um comportamento prejudicial dos modelos quando inseridos nos *testbenches* ajustados por cobertura, já que o esperado avanço na cobertura para algumas categorias selecionadas como meta, nunca ocorrerão. Os resultados nestes casos foram simbolizados pelo valor de 200%, como indicativo da condição desfavorável nas análises posteriores.

Na Tabela 20 observa-se que não há mudanças significativas nos valores dos erros acumulados quando substitui-se o conjunto médio pelo grande, para qualquer técnica de aprendizado, isto devido a que valor do erro é dominado pelo erro das classes majoritarias. De outro lado, quando o nível de balanceamento é intensificado, ou seja, a proporção da categoria majoritária passa dos 90,01% originais para os 70% e, depois, para os 50%, os erros de classificação das técnicas

de *data mining* e *data mining* sem corte também são aumentados. Uma possível explicação é que a redução de casos de teste na categoria majoritária faz com que algumas características não sejam treinadas e, portanto, o erro na classificação aumenta.

No caso das redes Bayesianas, a Tabela 20 mostra que o erro é mantido em todas as condições observadas, indiferente da condição do conjunto de treinamento e do balanceamento. Quando se observa a Tabela 21, os resultados mostram que o erro é devido ao fato de modelos com redes Bayesianas conseguirem unicamente incluir no modelo de aprendizado as categorias majoritárias. Já para os modelos com SVM, observa-se que o erro de classificação aumenta quando o balanceamento é utilizado.

Tabela 21 - Erros de falso positivo e falso negativo do CBP

Varied.	Cat	*Data mining* (DM)		DM sem corte		R. Bayesianas		SVM	
		F.P. (%)	F.N. (%)	F.P. (%)	F.N. (%)	F.P. (%)	F.N. (%)	F.P. (%)	F.N. (%)
C. médio	9	200,00	100,00	69,13	94,38	200,00	100,00	98,56	94,73
	10	200,00	100,00	61,07	94,03	200,00	100,00	99,63	97,50
	12	200,00	100,00	51,79	98,58	200,00	100,00	45,01	36,20
	17	200,00	100,00	50,48	92,94	200,00	100,00	98,81	96,05
	18	200,00	100,00	69,05	96,94	200,00	100,00	100,00	100,00
	20	200,00	100,00	59,56	98,96	200,00	100,00	97,23	91,04
	33	200,00	100,00	53,64	98,09	200,00	100,00	88,60	71,57
	34	200,00	100,00	56,40	99,08	200,00	100,00	96,77	90,72
	36	10,42	0,00	10,25	0,10	10,42	0,00	3,69	9,55
C. grande	9	200,00	100,00	67,27	92,98	200,00	100,00	98,27	95,44
	10	200,00	100,00	43,13	92,33	200,00	100,00	98,86	97,08
	12	200,00	100,00	41,49	98,03	200,00	100,00	83,27	71,49
	17	200,00	100,00	66,67	96,61	200,00	100,00	46,47	41,43
	18	200,00	100,00	58,33	95,92	200,00	100,00	100,00	100,00
	20	200,00	100,00	56,84	99,32	200,00	100,00	95,58	90,83
	33	200,00	100,00	63,91	97,17	200,00	100,00	83,60	71,70
	34	200,00	100,00	69,82	98,91	200,00	100,00	94,74	89,41
	36	10,42	0,00	10,18	0,21	10,42	0,00	5,01	9,19
C. médio bal 70	9	85,81	93,52	85,81	90,50	200,00	100,00	89,95	95,25
	10	200,00	99,15	60,35	92,05	200,00	100,00	96,36	97,36
	12	68,21	15,40	48,21	16,60	200,00	100,00	59,38	76,67
	17	200,00	99,44	87,77	92,94	200,00	100,00	91,46	96,91
	18	93,41	96,94	95,66	86,73	200,00	100,00	95,49	97,68
	20	88,09	33,71	88,09	32,68	200,00	100,00	80,34	92,01
	33	68,62	16,84	49,64	19,64	200,00	99,75	58,62	77,30
	34	87,99	32,66	47,99	32,45	200,00	100,00	79,68	92,51
	36	2,15	28,76	2,21	29,25	10,42	0,12	11,18	8,12
C. grande bal 70	9	78,27	91,47	65,24	85,42	200,00	100,00	89,97	96,03
	10	89,53	97,16	87,00	87,22	200,00	100,00	88,09	97,78
	12	67,66	15,71	37,63	16,02	200,00	99,87	47,87	76,46
	17	93,65	99,44	94,88	95,59	200,00	100,00	93,52	96,96
	18	90,79	94,90	90,94	83,67	200,00	100,00	96,81	97,83
	20	87,90	25,49	87,87	26,70	200,00	100,00	68,99	92,20
	33	69,15	15,48	49,53	15,45	200,00	99,98	46,51	77,42
	34	87,75	33,32	67,80	32,84	200,00	100,00	66,86	92,66
	36	2,02	29,54	2,03	29,64	10,42	0,15	12,99	7,64
C. médio bal 50	9	81,89	91,58	81,86	88,88	200,00	100,00	81,12	96,34
	10	200,00	100,00	92,11	90,34	200,00	100,00	88,91	97,74
	12	67,29	16,29	67,46	17,55	200,00	100,00	38,60	74,02
	17	200,00	99,44	86,11	96,89	200,00	100,00	81,09	97,88
	18	200,00	100,00	89,30	92,86	200,00	100,00	70,31	91,25
	20	86,02	98,61	87,75	92,06	200,00	100,00	67,89	92,77
	33	67,34	17,93	67,34	19,38	200,00	100,00	48,89	79,40
	34	200,00	96,24	87,36	83,42	200,00	100,00	65,82	93,25
	36	4,24	14,07	4,05	15,90	10,42	0,00	17,63	7,29
C. grande bal 50	9	73,96	96,44	80,59	93,52	200,00	100,00	82,93	96,01
	10	200,00	99,15	80,45	85,80	200,00	100,00	88,09	97,08
	12	67,23	16,66	67,06	17,41	200,00	100,00	53,32	76,59
	17	200,00	100,00	81,83	96,05	200,00	100,00	83,52	96,77
	18	88,06	91,84	88,24	81,63	200,00	100,00	91,81	97,14
	20	84,76	97,41	84,55	93,47	200,00	100,00	68,99	92,21
	33	67,54	16,69	67,64	17,20	200,00	100,00	46,50	77,42
	34	85,31	96,02	64,53	91,83	200,00	100,00	66,86	92,16
	36	4,14	14,15	4,08	14,77	10,42	0,00	12,99	7,64

5.3.1.3 Resultados dos *testbenches*

Os resultados da execução dos *testbenches* com geração de casos de teste ajustada por cobertura são apresentados na Tabela 22, que está organizada da seguinte forma: a primeira coluna apresenta as técnicas usadas nos *testbenches* com ajuste na geração; a segunda coluna apresenta as variedades de tamanho dos conjuntos de treinamento; a terceira coluna apresenta a quantidade de casos de teste para se atingir a cobertura de 10.000 itens, definida em na página 66; a quarta coluna apresenta o tempo gasto na execução do *testbench*; as quinta e sexta colunas apresentam a porcentagem de redução de casos de teste e de tempo, respectivamente, com respeito a *testbench* sem ajuste. Adicionalmente, para efeitos de comparação, a Tabela 22 apresenta na segunda fila o resultado do *testbench* sem ajuste na geração de casos de teste, sendo utilizado como referência de comparação no momento de determinar as reduções nas características observadas.

Tabela 22 - Resultados dos *testbenches* do CBP

Técnica	Condição	Casos de teste	Tempo (s)	Redução casos (%)	Redução tempo (%)
Testbench sem ajuste		87.381	6.864		
Data mining	C. médio	87.381	6.869	0,00	-0,07
	C. grande	87.381	6.929	0,00	-0,94
	C. médio bal 70	82.301	6.543	5,81	4,68
	C. grande bal 70	73.499	5.791	15,89	15,64
	C. médio bal 50	49.212	3.914	43,68	42,98
	C. grande bal 50	48.504	3.937	44,49	42,65
Data mining sem corte	C. médio	75.479	6.006	13,62	12,50
	C. grande	68.278	5.637	21,86	17,88
	C. médio bal 70	60.766	4.823	30,46	29,73
	C. grande bal 70	57.400	4.603	34,31	32,94
	C. médio bal 50	51.456	4.144	41,11	39,63
	C. grande bal 50	48.504	3.937	44,49	42,65
Redes Bayesianas	C. médio	215.984	21.298	-147,17	-210,28
	C. grande	234.202	23.123	-168,02	-236,87
	C. médio bal 70	201.143	19.970	-130,19	-190,93
	C. grande bal 70	205.495	20.383	-135,17	-196,96
	C. médio bal 50	175.225	17.216	-100,53	-150,81
	C. grande bal 50	185.304	18.192	-112,06	-165,03
SVM	C. médio	108.437	315.844	-24,10	-4.501,45
	C. grande	101.852	325.941	-16,56	-4.648,55
	C. médio bal 70	202.524	38.665	-131,77	-463,30
	C. grande bal 70	269.277	70.450	-208,16	-926,37
	C. médio bal 50	186.668	31.598	-113,62	-360,34
	C. grande bal 50	267.495	69.447	-206,12	-911,76

Dos resultados apresentados na Tabela 22 se observa as técnicas de *data mining* e *data mining* sem corte tiveram valores positivos e portanto satisfatórios tanto de ganho de casos de teste quanto de tempo de execução, como detalhado nas colunas 5 e 6. Nas outras técnicas foi observado que a geração com ajuste não conseguiu acelerar o *testbench* e pelo contrario o desacelerou ficando ainda pior que o *testbench* sem ajuste. Sendo, este efeito mostrado nas filas inferiores das colunas 5 e 6.

5.3.1.4 Análise geral dos resultados

A análise dos resultados com o módulo CBP foi feita com o cruzamento dos dados obtidos na avaliação dos modelos obtidos com a aprendizagem e na execução do *testbench*. Considerando-se que os erros de classificação calculados na página 83 são relativos ao conjunto de classificação utilizado e que os valores do erro absoluto requereria do espaço de casos de teste completo, nesta análise optou-se por fazer uma comparação qualitativa entre os modelos aprendidos. A análise utilizada para comparar dos resultados de erro de cada categoria foi baseada no efeito dos erros de FP e FN, sendo consideradas as seguintes premissas:

- Um modelo é relativamente muito bom, quando comparado a um segundo modelo, se tanto os erros de FP quanto os de FN forem menores.
- Um modelo é relativamente bom, quando comparado a um segundo modelo, se o erro de FP é menor e o erro de FN é maior ou igual.
- Um modelo é relativamente ruim, quando comparado a um segundo modelo, se o erro de FP é maior e o erro de FN é menor ou igual.
- Um modelo é relativamente muito ruim, quando comparado a um segundo modelo, se tanto os erros de FP e FN forem maiores.
- Um modelo é equivalente, quando comparado a um segundo modelo, se tanto os erros de FP e de FN forem iguais.

A comparação entre modelos completos é realizada dando às qualidades os seguintes valores e somando: muito bom = 2, bom = 1, ruim = -1, muito ruim = -2, equivalente = 0. Com resultado da soma das qualidades das categorias minoritárias é considerado que um modelo é relativamente “melhor” com respeito a outro se a

soma é maior de zero, “igual” se o valor é igual a zero e “pior” se a soma é menor que zero. Esta análise qualitativa mostra tendência dos efeitos dos erros de FP e FN, porém ela é só uma tendência dado que os erros de FP e FN são estimativas relativas ao conjunto padrão de classificação utilizado. O valor real dos erros de FP e FN requereria o uso do espaço total de casos de teste, como foi utilizado no caso de estudo do STREAMPROC.

Nesta análise foram observados quais dos eventos de cobertura requereram o estímulo para a aceleração durante a execução do *testbench* e as categorias que nunca estiveram atrasadas por conta dos eventos de cobertura foram retiradas da análise qualitativa. Da observação, a Tabela 23 foi obtida, onde tem-se na primeira coluna os eventos de cobertura, enquanto, na segunda, as categorias às quais eles pertencem são listadas, e, por fim, são apresentadas na terceira coluna, as médias das ocasiões em que os eventos de cobertura correspondentes estavam atrasados. Ilustrando, a Tabela 23 mostra que os eventos 3, 4 e 6 nunca estiveram atrasados portanto nunca necessitaram de aceleração, fazendo que os modelos de aprendizado não fossem utilizados durante a execução dos *testbenches* nas suas categorias 12 e 36. Por este motivo estas duas categorias não fazem parte da analise qualitativa dos erros de FP e FN realizada para comparar os erros dos modelos.

Tabela 23 - Quantidade média de ocasiões de atraso no *testbench* CBP

Eventos	Categorias	Média de ocasiões em atraso
Ev. 1	9, 17, 33	21
Ev. 2	10, 18, 34	56.312
Ev. 3	12, 20, 36	0
Ev. 4	9, 10,12	0
Ev. 5	17, 18, 20	43.884
Ev. 6	33, 34, 36	0

Com a análise acima e com a diferença entre os resultados de ganho dos *testbenches*, três comparações foram feitas: segundo as condições de tamanho do conjunto de treinamento, segundo o nível de balanceamento e segundo a técnica de aprendizado utilizada no treinamento.

5.3.1.4.1 Comparativo segundo o tamanho do conjunto

O comparativo entre os resultados do treinamento e da execução dos *testbenches* segundo o tamanho de treinamento visa mostrar o efeito de aumentar a quantidade de informação utilizada durante o aprendizado. Foi comparada a qualidade dos modelos obtidos através dos treinamentos realizados com conjuntos grandes e médios, assim como, foram feitas comparações entres resultados de testbenches utilizando-se tais modelos. Os resultados comparativos numéricos foram tabelados[4], porém, ilustrações como nas figuras 22, 23, 24 e 25 mostram com mais clareza as tendências verificadas com o uso dos modelos. As figuras apresentam comparações de resultados quando os conjuntos grandes são utilizados ao invés dos conjuntos médios na etapa de treinamento (usando-se estes últimos sempre como referência). As figuras apresentam duas facetas dos resultados: uma correspondente ao comparativo qualitativo, com a apresentação de uma barra azul de valor positivo quando o modelo de conjunto grande é melhor que o modelo médio, um valor negativo quando o conjunto grande é pior que o modelo médio e nenhuma barra quando o modelo grande é equivalente ao modelo médio. A segunda faceta corresponde às diferenças entre os resultados dos *testbenches* utilizando-se modelos aprendidos com os conjuntos grande e médio, sendo apresentada em vermelho a diferença entre a redução do número de casos de teste e em verde a diferença entre a redução do tempo de execução. As figuras 22, 23, 24 e 25 mostram as comparações feitas com resultados obtidos com as técnicas de *data mining*, *data mining* sem corte, redes Bayesianas e SVM, respectivamente.

A Figura 22 mostra três comparações entre os conjuntos grandes e médios para as condições de treinamento com *data mining* sem balanceamento, *data mining* com balanceamento 70% e *data mining* com balanceamento 50%. Nesta figura mostra-se que para a técnica *data mining* com conjuntos sem balanceamento, a qualidade dos modelos aprendidos são equivalentes, enquanto os resultados dos *testbenches* foram iguais no número de casos de teste e um pouco piores no tempo de execução, isto devido a que os modelos com conjuntos grandes têm a tendência a serem maiores e requererem maior tempo de execução. Para os experimentos com conjuntos com balanceamento a 70% e 50%, observou-se que os modelos grandes

[4] As tabelas correspondentes podem ser vistas no Apêndice 1 (A1.1.).

são melhores que os modelos médios, enquanto que, do ponto de vista dos resultados dos testbenches, observou-se uma melhoria para o balanceamento de 70% e uma melhoria parcial para o balanceamento de 50%. Isto é devido ao fato de que os resultados do balanceamento 50% são os melhores obtidos com os testbenches mostrando um limite na capacidade de melhora do ajuste por cobertura. Este efeito deve-se principalmente a que nem todos os casos de teste são factíveis de serem acelerados; dadas determinadas restrições associadas à posição do pixel no quadro completo.

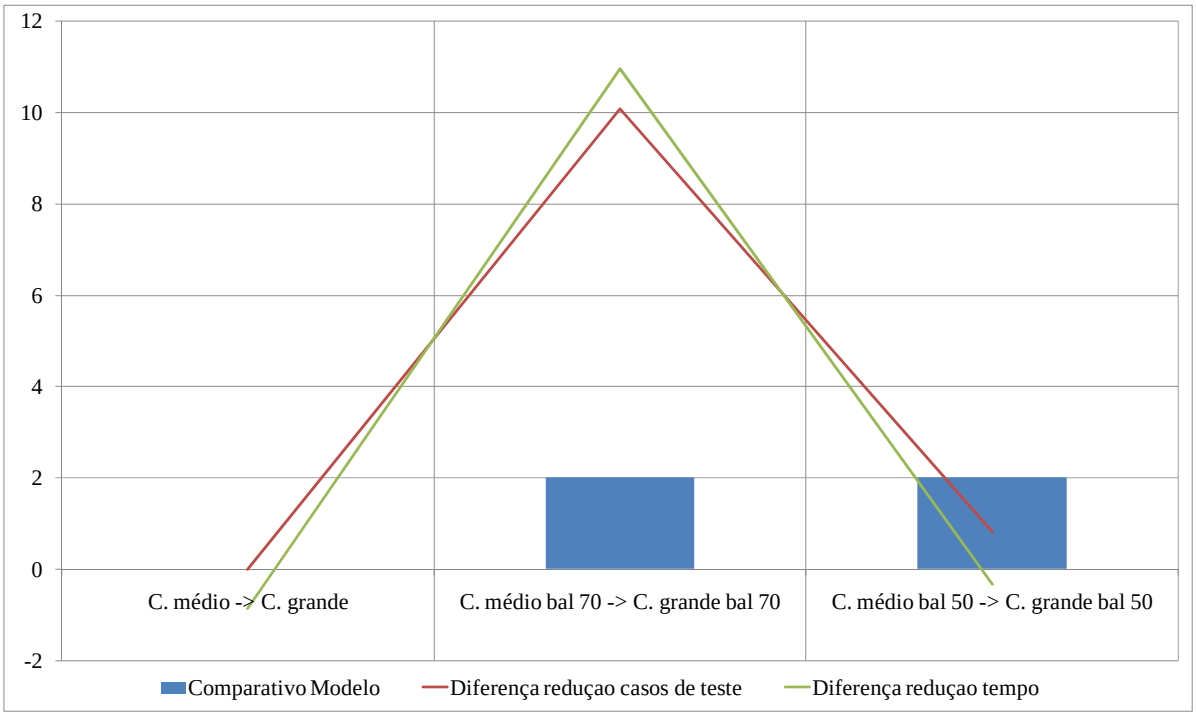

Figura 22 - Comparativo dos resultados do treinamento com *data mining* e do *testbench* entre os conjuntos de tamanho médio e grande, para o CBP

A Figura 23 mostra as comparações realizadas com os modelos construídos usando a técnica de *data mining* sem a opção de corte. Neste caso, também são comparados os conjuntos aprendidos sem balanceamento e com o balanceamento de 70% e 50%, na primeira, segunda e terceira coluna respectivamente. Neste caso, observa-se que em todas as comparações há uma melhoria na qualidade do modelo aprendido quando o conjunto grande é utilizado ao invés do conjunto médio. Nesta figura se mostra que em todos os casos se observa uma diferença positiva tanto de quantidade de casos de teste quanto de tempo de execução dos *testbenches*, significando que para esta técnica o aumento do tamanho representa uma melhora

nos resultados dos *testbenches*. A Figura 23 mostra também que as melhoras segundo o tamanho são maiores quando o balanceamento não é utilizado; quando o balanceamento é aplicado, os resultados são mais uniformes, já que estão mais próximos aos limites de melhoria, mencionados previamente.

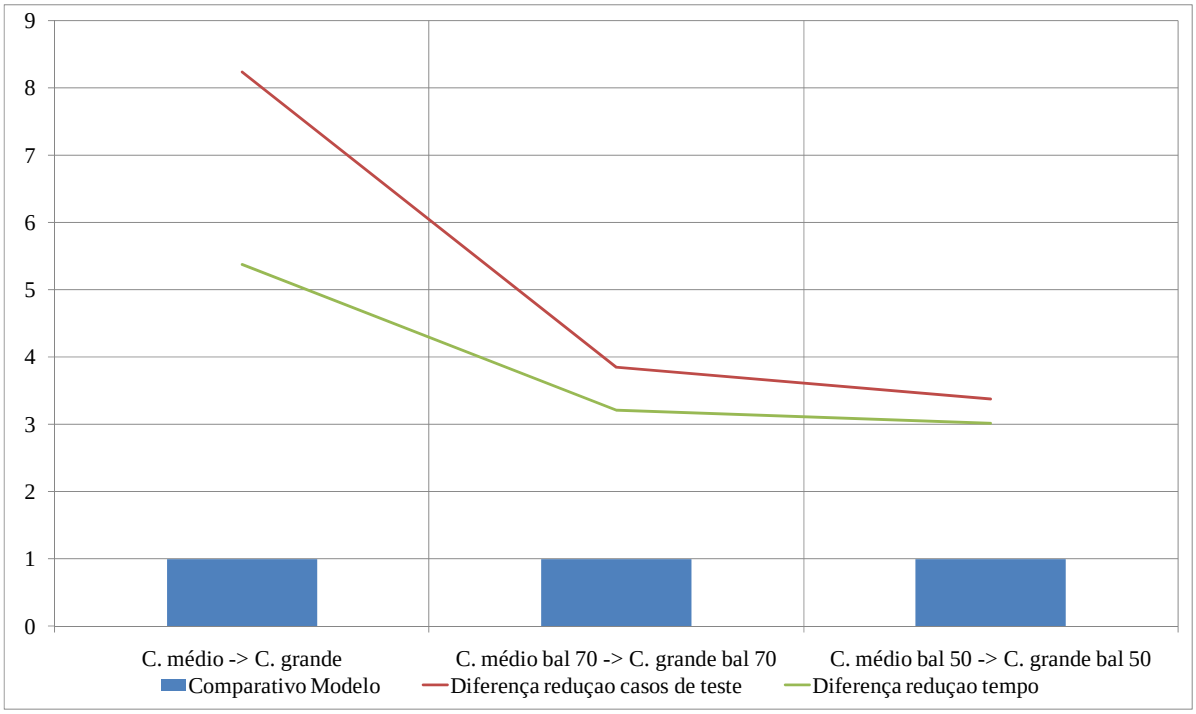

Figura 23 - Comparativo dos resultados do treinamento com *data mining* sem corte e do *testbench* entre os conjuntos de tamanho médio e grande, para o CBP

A Figura 24 apresenta as comparações com respeito à variação do tamanho dos conjuntos de treinamento nos modelos e *testbenches* que utilizaram redes Bayesianas para a sua construção. Estas comparações mostram, da esquerda para direita, os resultados do uso dos conjuntos grandes ao invés dos conjuntos médios para as condições sem balanceamento, balanceamento 70% e balanceamento 50%. Nesta figura, observa-se que os modelos obtidos são equivalentes exceto para o caso com balanceamento 70% no qual é melhor. Também verifica-se que o aumento do tamanho do conjunto de treinamento piora os resultados dos *testbenches*. Observa-se na figura que os resultados dos *testbenches* são melhores no caso do balanceamento 70%, ou seja, aquele modelo que teve melhora no aumento do tamanho do conjunto de treinamento. Estes resultados podem ser atribuídos ao fato de que os erros de FN para o *testbench* com redes Bayesianas estão muito próximos do valor de 100% em todas as categorias minoritárias; isto significa que a

geração de casos de teste realizada de acordo com os modelos de redes Bayesianas obtidos no treinamento não levam às categorias esperadas, fazendo com que a componente aleatória de geração seja dominante.

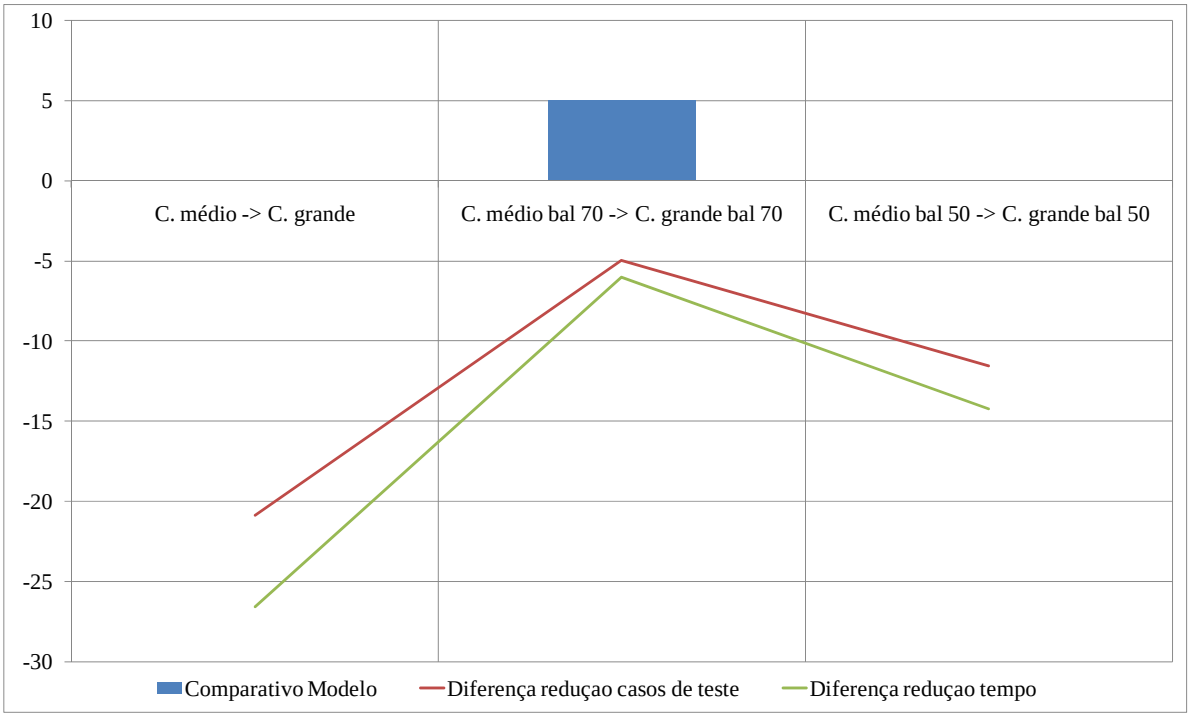

Figura 24 - Comparativo dos resultados do treinamento com redes Bayesianas e do *testbench* entre os conjuntos de tamanho médio e grande, para o CBP

A Figura 25 apresenta as comparações realizadas variando do tamanho do conjunto de treinamento para os modelos e *testbenches* construídos com a técnica SVM. Nesta figura são apresentadas, da esquerda para a direita, as comparações do uso de conjuntos grandes e invés dos conjuntos médios nas condições sem balanceamento, com balanceamento 70% e com balanceamento 50%. A Figura 25 mostra que para o caso sem balanceamento o modelo aprendido com o conjunto grande é melhor que modelo aprendido com o conjunto médio; já para os casos com balanceamento, os modelos aprendidos com o conjunto grande são piores. Entretanto os resultados dos *testbenches* mostram que, para a comparação sem balanceamento, observa-se uma pequena melhoria na quantidade de casos de teste enquanto percebe-se uma piora no tempo de execução. Já nos casos com balanceamento, os modelos aprendidos são piores, e os resultados dos *testbenches* são piores tanto na quantidade de casos de teste quanto no tempo de execução. As diferenças negativas nos tempos de execução são devidas principalmente ao fato de

que os modelos SVM de conjuntos grandes têm a tendência a serem maiores que os obtidos com conjuntos médios; adicionalmente, no SVM o tempo de processamento é sensível ao tamanho do modelo fazendo com que se apresentem grandes diferenças entre os resultados obtidos com conjuntos grandes e médios.

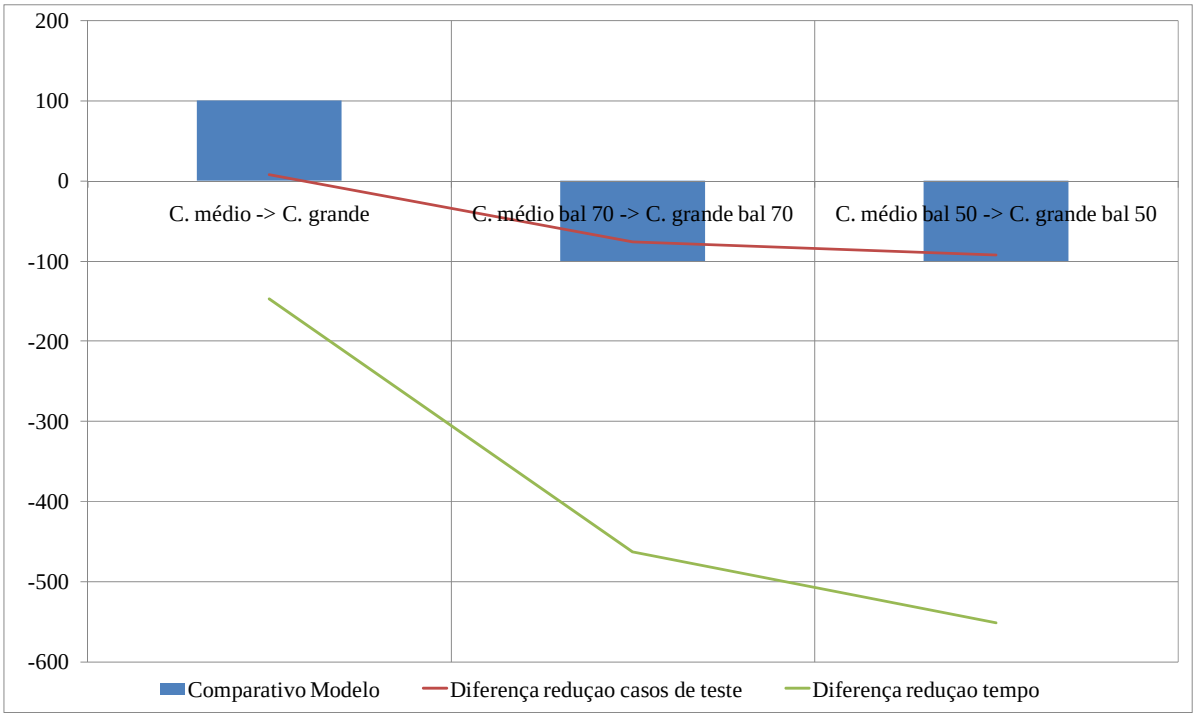

Figura 25 - Comparativo dos resultados do treinamento com SVM e do *testbench* entre os conjuntos de tamanho médio e grande, para o CBP

5.3.1.4.2 Comparativo segundo o balanceamento

Nesta seção é feito um comparativo entre os resultados das estimativas de erro dos modelos aprendidos e entre os resultados dos *testbenches* que usam tais modelos para o ajuste da geração dos casos de teste. As figuras 26, 27, 28 e 29 apresentam as comparações dos modelos e *testbenches* com *data mining*, *data mining* sem corte, redes bayesianas e SVM, respectivamente, enquanto dados numéricos completos podem ser vistos nas tabelas do Apêndice 1 (A1.2.). De forma semelhante ao comparativo por variação de tamanho (visto na seção anterior) estas figuras apresentam em forma de barras azuis os resultados do comparativo qualitativo entre modelos aprendidos e em forma de linhas vermelhas e verdes as diferença da redução na quantidade de casos de teste e de tempo de execução. As figuras estão organizadas da esquerda para a direita, com as seguintes comparações: a primeira parte apresenta os resultados de se usar o balanceamento

70% no conjunto médio ao invés do conjunto médio sem balanceamento, a segunda parte apresenta a comparação de se usar o conjunto grande com balanceamento 70% ao invés do conjunto grande sem balanceamento, a terceira parte apresenta a comparação de se usar o conjunto médio com balanceamento 50% ao invés do conjunto médio com balanceamento 70% e a quarta parte apresenta a comparação de se usar o conjunto grande com balanceamento 50% ao invés do conjunto grande com balanceamento 70%. Com estas comparações visa-se mostrar o efeito do aumento do nível de balanceamento já que o balanceamento 50% é mais restritivo que o balanceamento 70% e que o balanceamento 70% é mais restritivo que sem balanceamento.

A Figura 26 apresenta as comparações realizadas entre os diferentes níveis de balanceamento para os modelos e *testbenches* construídos com a técnica *data mining*. Pode-se observar por esta figura que, em todas as comparações, os modelos aprendidos com maior balanceamento são melhores, tanto para os conjuntos médios como grandes. A figura também mostra que, pelas diferenças verificadas nos resultados dos diferentes *testbenches*, elas se mostram melhores tanto para a redução da quantidade de casos de teste quanto para a redução de tempo de execução.

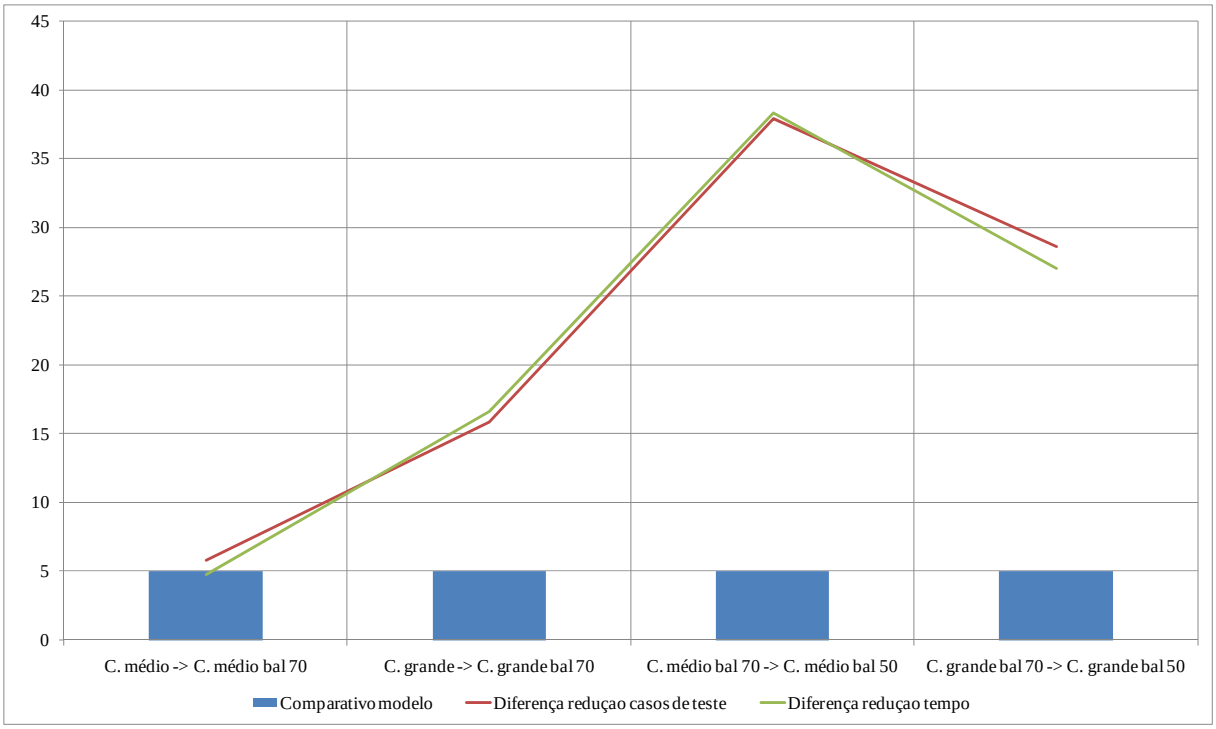

Figura 26 - Comparativo dos resultados do treinamento com *data mining* e do *testbench* entre os conjuntos com diferentes balanceamentos, para o CBP

A Figura 27 apresenta as comparações realizadas entre os diferentes níveis de balanceamento para os modelos e *testbenches* construídos com a técnica *data mining* sem a opção de corte. Nesta figura, pode-se mostrar que as comparações entre os modelos aprendidos são melhores à medida em que o nível de balanceamento é aumentado. A figura mostra melhorias também nos resultados dos *testbenches* quando o nível de balanceamento é aumentado. Nesta figura, percebemos também uma tendência a se obter cada vez diferenças menores; tal fato deve-se principalmente a que os resultados com o balanceamento 50% e 70% estão mais perto dos melhores obtidos em todo o comparativo, portanto sendo a diferença entre eles menor.

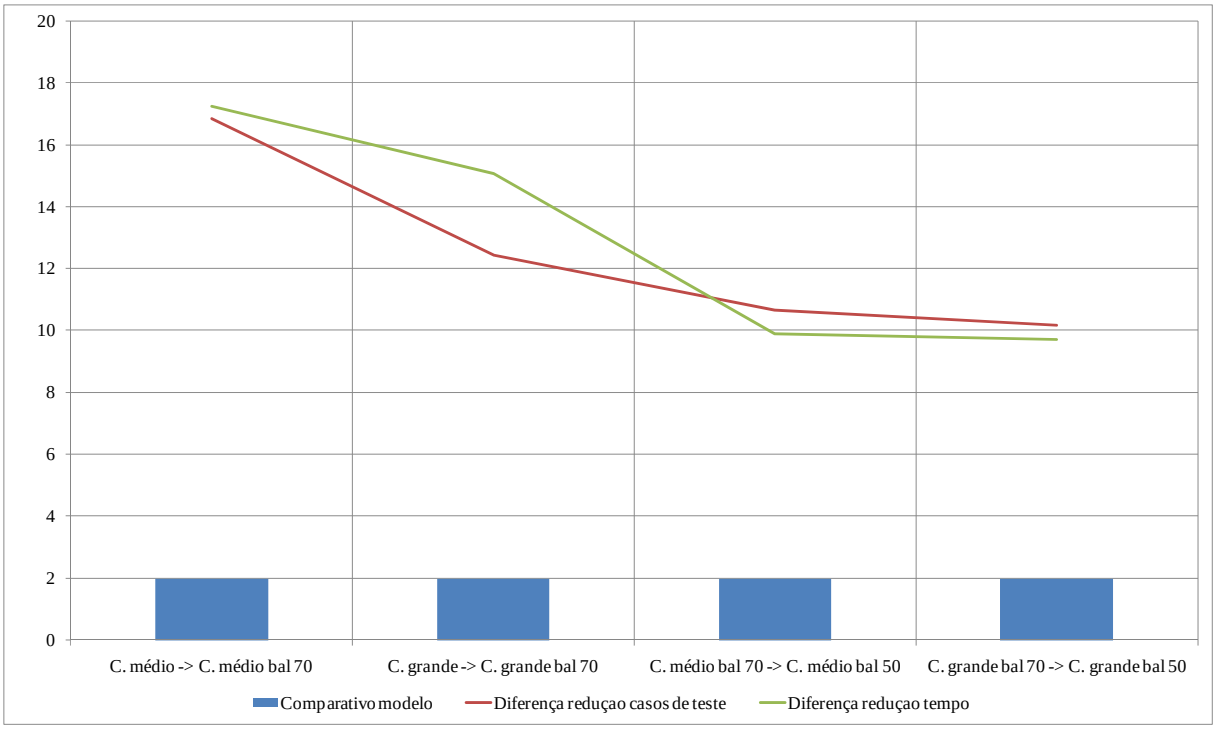

Figura 27 - Comparativo dos resultados do treinamento com *data mining* sem corte e do *testbench* entre os conjuntos com diferentes balanceamentos, para o CBP

A Figura 28 apresenta as comparações realizadas entre os diferentes níveis de balanceamento para os modelos e *testbenches* construídos com a técnica redes Bayesianas. Nesta figura mostra-se que os modelos aprendidos só conseguem melhorar no caso de se começar a utilizar o balanceamento; porém quando são comparados modelos com balanceamento, os resultados são equivalentes. Por outro lado, os resultados dos testbenches apresentam melhorias em todos os casos comparativos mesmo quando não ocorre melhoria nos modelos aprendidos. A Tabela 21 mostra nas colunas sétima e oitava que, para as categorias minoritárias, FP e FN são praticamente 200% e 100% respectivamente, significando que, quando estas categorias são realimentadas, a rede Bayesiana não mostra um patrão consistente, fazendo que os resultados dos *tesbenches* sejam inconsistentes.

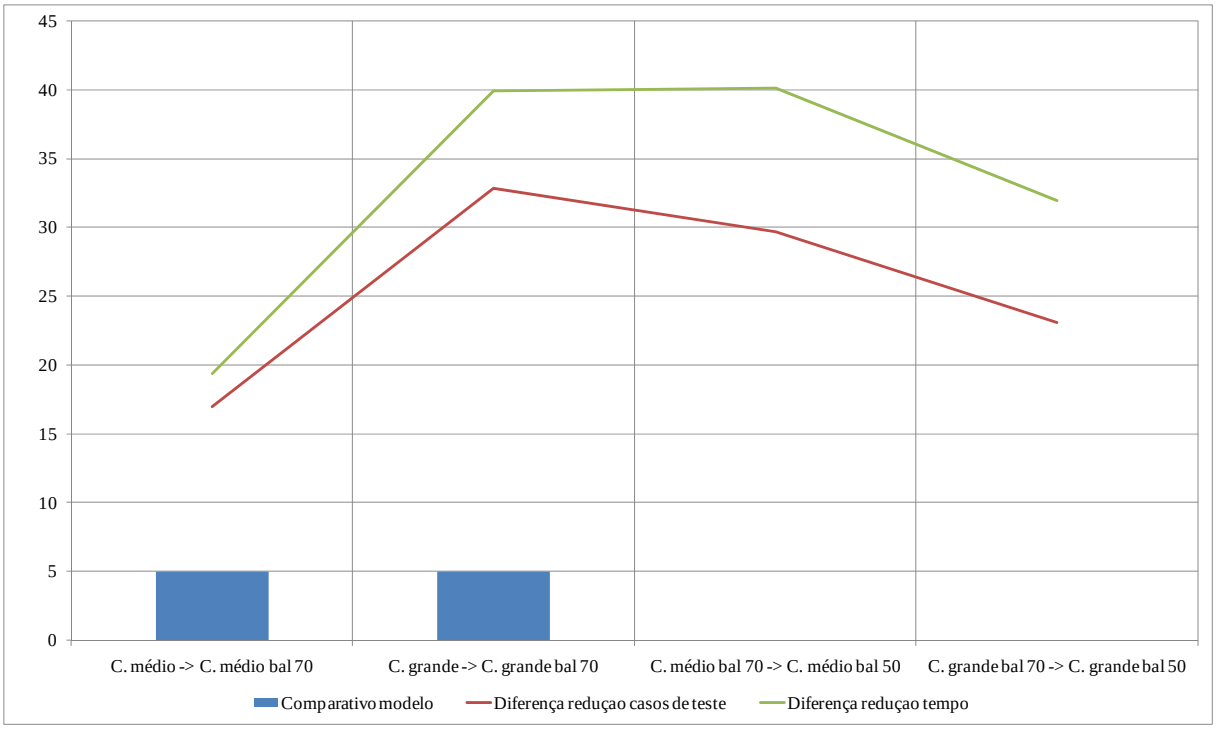

Figura 28 - Comparativo dos resultados do treinamento com redes Bayesianas e do *testbench* entre os conjuntos com diferentes balanceamentos, para o CBP

A Figura 29 apresenta as comparações realizadas entre os diferentes níveis de balanceamento para os modelos e *testbenches* construídos com a técnica SVM. Nesta figura, pode-se observar que todos os modelos aprendidos são melhores enquanto o nível de balanceamento é maior. Com respeito aos resultados dos *testbenches* a figura mostra uma grande diferença entre os seus resultados, com e sem balanceamento, especificamente na diferença da redução do tempo. Isto deve-se principalmente ao fato de que o custo computacional da técnica SVM com modelos grandes é alto e dado que os modelos com balanceamento são significativamente menores, os tempos de execução dos mesmos são mais rápidos que os modelos sem balanceamento. Entretanto a diferença da redução dos casos de teste é negativa dos testbenches com balanceamento 70% com respeito aos *testbenches* sem balanceamento e esta diferença é perto de zero entre os *testbenches* com balanceamento; este efeito acontece devido a que a técnica de filtragem mascara as melhorias dos modelos aprendidos na execução dos *testbenches*. Quanto à redução do tempo de execução, observa-se a tendência de ela ser mais expressiva com o nível de balanceamento, uma vez que o balanceamento faz com que os modelos SVM sejam menores e, portanto mais

rápidos. Dado que o balanceamento reduz o número de casos de teste das categorias majoritárias assim como o tamanho do conjunto, então os modelos de SVM tendem a reduzir a quantidade vetores de suporte. Dado que nos testbenches com SVM, o sobre-processamento devido à filtragem ao próprio modelo SVM pode chegar a ser dominante, então o tempo de execução é reduzido à medida em que os modelos SVM são menores e mais rápidos.

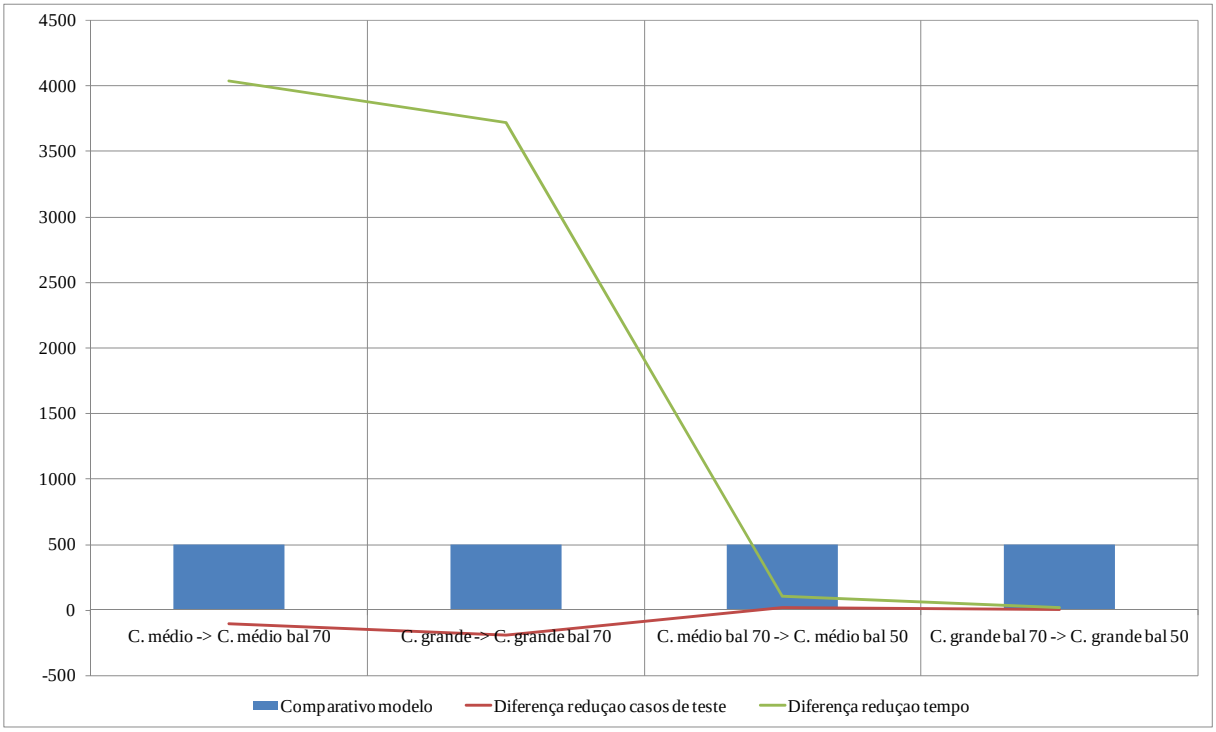

Figura 29 - Comparativo dos resultados do treinamento com SVM e do *testbench* entre os conjuntos com diferentes balanceamentos, para o CBP

5.3.1.4.3 Comparativo segundo a técnica de aprendizado

A análise dos resultados do caso de estudo do CBP considerou um último aspecto importante que é a dependência dos resultados com respeito á técnica de aprendizado utilizada, tendo-se a técnica de *data mining* como referência de comparação. Resultados numéricos em tabelas são apresentados no Apêndice 1 (A1.3.), enquanto que, para melhor visualização, resultados comparativos são ilustrados em figuras. Os modelos aprendidos e os resultados dos *testbenches* foram comparados da seguinte forma: a Figura 30 apresenta as comparações feitas com os resultados modelos e testbenches construídos quando a técnica *data mining* sem corte é usada ao invés da técnica *data mining*; a Figura 31 apresenta as comparações dos resultados quando a técnica redes Bayesianas é utilizada ao invés

da técnica *data mining*; e a Figura 32 apresenta as comparações dos resultados com o uso de SVM ao invés de *data mining*. Estas figuras seguem o mesmo protocolo utilizado nas seções anteriores, onde os resultados das comparações dos modelos aprendidos foram apresentados com barras azuis, e as diferenças dos resultados dos testbenches com linhas vermelhas e verdes, para a quantidade de casos de teste e tempos de execução, respectivamente. As figuras estão organizadas da esquerda para a direita segundo as seguintes condições de treinamento: conjunto médio sem balanceamento, conjunto grande sem balanceamento, conjunto médio com balanceamento 70%, conjunto grande com balanceamento 70%, conjunto médio com balanceamento 50% e conjunto grande com balanceamento 50%.

A Figura 30 apresenta as comparações dos resultados de uso de *data mining* sem corte ao invés de *data mining*. Nesta figura se mostra que os modelos aprendidos com a técnica de *data mining* sem corte são melhores para todas as condições de treinamento comparado à técnica de *data mining* (com corte); a razão para isto é que o corte faz com que os modelos sejam mais generalizados, reduzindo-se os detalhes das categorias minoritárias. Este fato faz com que os erros de FN e FP sejam maiores no *data mining* que na *data mining* sem corte. Entretanto os resultados dos *testbenches* mostram que quando são utilizados conjuntos sem balanceamento e balanceamento 70% os resultados são melhores com a técnica *data mining* sem corte. Já para o balanceamento em 50%, os resultados dos *testbenches* são muito semelhantes; tal fato deve-se a que estes *testbenches* têm os melhores resultados de redução no número de casos de teste e de tempo (ver Tabela 22), indicando que existe uma limitante, que mascara as melhorias do modelo de aprendizado.

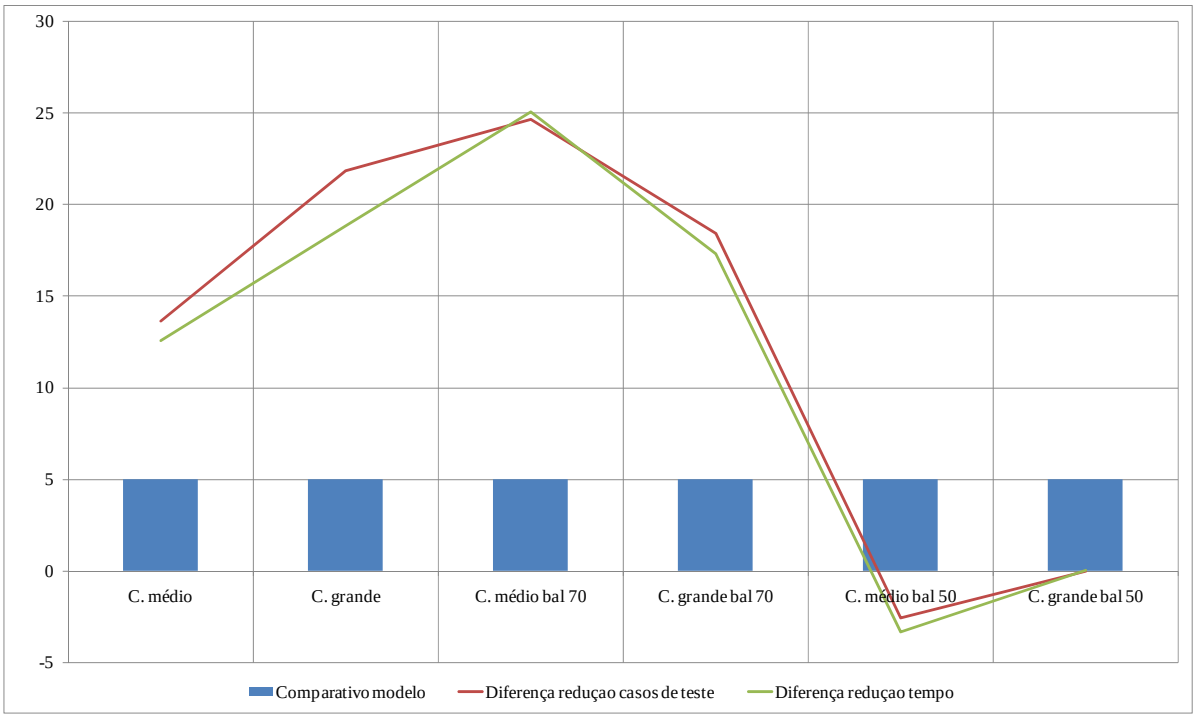

Figura 30 - Comparativo dos resultados do treinamento e do *testbench* para o CBP entre as técnicas de *data mining* sem corte e *data mining*

A Figura 31 apresenta as comparações realizadas no uso da técnica de redes Bayesianas ao invés da técnica *data mining*. As comparações dos resultados dos modelos aprendidos mostram que para os modelos sem balanceamento os resultados são equivalentes, e para os modelos com balanceamento os resultados das redes Bayesianas são piores. Entretanto os resultados dos testbenches mostra que as redes Bayesianas são sempre priores que os resultados com *data mining*, sendo comparativamente melhores quando os conjuntos médios são utilizados. A explicação para tal é a forma diferenciada que as técnicas de *data mining* e redes Bayesianas tratam as categorias minoritárias. Na técnica de *data mining*, quando o modelo não possui informação sobre uma categoria, ele não aponta para nenhum caso de teste, fazendo que seja necessário utilizar a geração aleatória para criar o caso de teste (que pode resultar ou não no evento esperado). Entretanto, nas redes Bayesianas, o modelo inverso sempre aponta para algum caso de teste, normalmente dentro das categorias que ela possui informação. Como mostra a Tabela 21, os erros de FP das redes Bayesianas são comumente altos para todas as categorias minoritárias e baixos para as categorias majoritárias. Portanto, o ajuste com redes Bayesianas tende a colocar casos de teste das categorias majoritárias

quando são as minoritárias que necessitam de aceleração, aumentando desta maneira o tempo de execução do *testbench*.

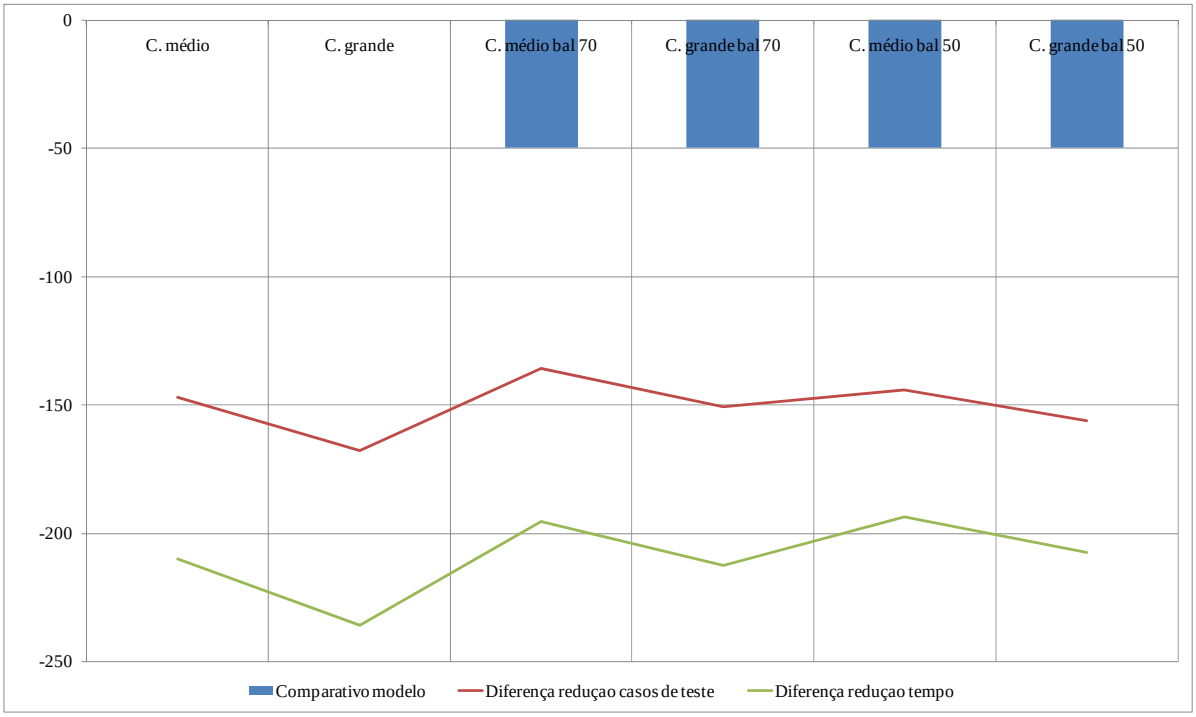

Figura 31 - Comparativo dos resultados do treinamento e do *testbench* para o CBP entre as técnicas de redes Bayesianas e *data mining*

A Figura 32 apresenta as comparações entre os resultados dos modelos e *testbenches* construídos com a técnica SVM ao invés da técnica *data mining*. Nesta figura, observa-se que os modelos aprendidos com SVM são sempre melhores que os aprendidos com data mining. Entretanto, os resultados dos *testbenches* são equivalentes para a diferença na redução de casos de estudo nos conjuntos sem balanceamento e piores nos conjuntos com balanceamento. Em geral se observa que, para todas as variedades de conjuntos de treinamento, ocorre uma melhoria nos resultados quando a técnica de SVM é usada ao invés de *data mining,* porém o efeito da filtragem de casos de teste, típico do *testbench* com SVM, faz com que as características positivas do modelo não sejam completamente exploradas. Isto ocorre porque que a filtragem usa como fonte o gerador de aleatório o qual está focado nos eventos de cobertura das categorias majoritárias. Nesse sentido, a técnica de *data mining* é mais eficiente porque ele consegue propor casos de teste independente de outros fatores; por isto as mudanças com na geração de casos de teste são mais fortes quando a *data mining* é utilizada. Mesmo com um modelo

melhor, os resultados com dos *testbenches* do SVM apresentam maior número de casos de teste que os de *data mining*. Já no caso de tempo de execução, a filtragem requer que os modelos SVM sejam executados mais vezes que os de *data mining,* piorando a situação para o SVM. Além disso, o efeito é mais critico com os modelos SVM com maior número de vetores de suporte, ou seja, aqueles sem balanceamento.

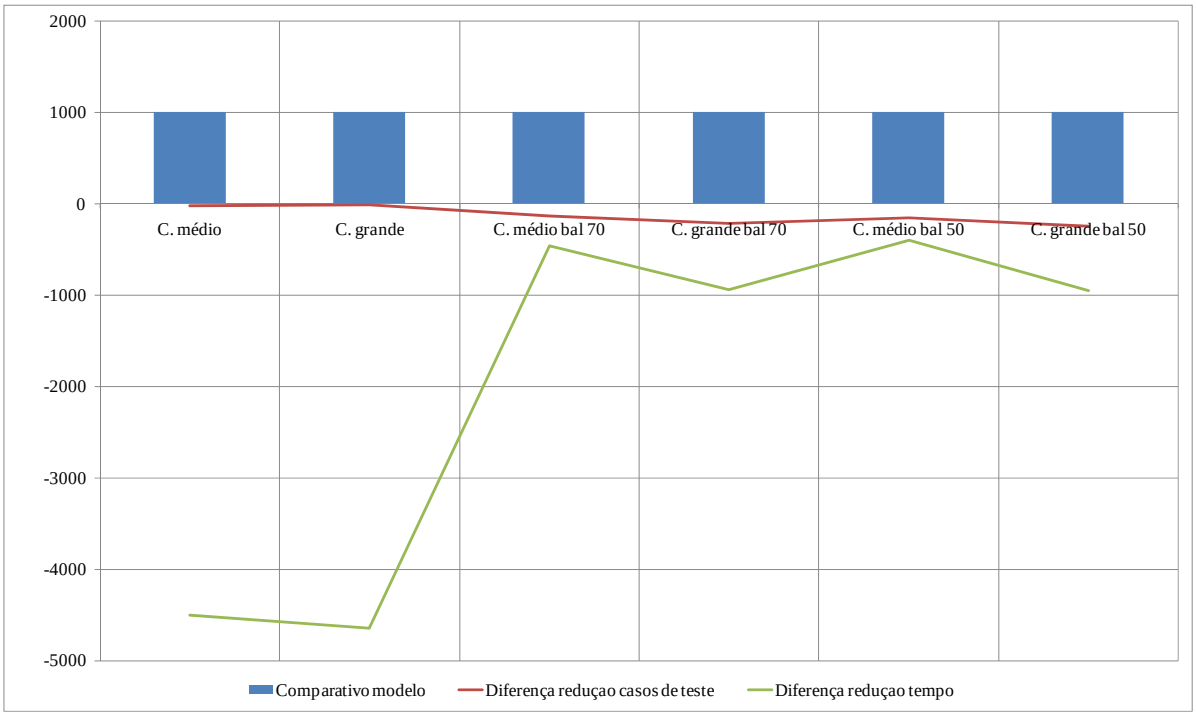

Figura 32 - Comparativo dos resultados do treinamento e do *testbench* para o CBP entre as técnicas de SVM e *data mining*

5.3.1.5 Conclusões da análise

Pela análise realizada com o caso de estudo do módulo CBP, observa-se que a técnica com melhores resultados é a de *data mining* sem corte, sendo a única a superar, na avaliação do treinamento e execução dos *testbenches,* a técnica de *data mining*, tomada como referência de comparação. Este resultado foi devido ao fato de que a inferência inversa usada indica, de imediato, o caso de teste a ser aplicado para a aceleração da cobertura e a que os resultados do treinamento apresentaram melhores resultados de erro de FP e FN nas categorias de interesse na aceleração.

De forma geral, foi observado que os resultados do treinamento foram melhores quando o nível de balanceamento e o tamanho do conjunto foram aumentados,

porém tal melhoria não redundou em melhorias nos resultados dos *testbenches* nas técnicas redes Bayesianas e SVM. Nas redes Bayesianas, os modelos apresentaram grandes limitações nas categorias de interesse para a aceleração. Na SVM, a filtragem faz com que o ajuste dependa mais da geração aleatória e do limite imposto para evitar que seja infinita a procura por características não representadas no modelo.

5.3.2 Caso de estudo módulo PIACDC

5.3.2.1 Detalhes dos experimentos

A presente seção apresenta a formulação da análise experimental realizada no módulo PIACDC, mostrando as considerações relevantes sobre as variedades das de tamanho do conjunto de treinamento, das técnicas de aprendizado e dos *testbenches* construídos

5.3.2.1.1 Categorização dos eventos de cobertura

Como foi apresentado na página 45 o treinamento requer de uma categorização dos eventos de cobertura, para garantir que sejam excludentes. Para a aplicação das técnicas de geração com ajuste por cobertura no módulo PIACDC, a categorização foi feita como na página 78, considerando a ocorrência dos eventos como uma variável binária de 6 dígitos, onde o evento 1 corresponde ao dígito menos significativo. Nesta variável foi observado que unicamente as categorias 5, 6, 8, 17, 18, e 32 tiveram amostras nos conjuntos de treinamento.

5.3.2.1.2 Condições de tamanho

De forma semelhante ao caso de estudo do módulo CBP, a análise do módulo PIACDC necessita de uma avaliação da quantidade de combinações de entradas possíveis e válidas. Para as entradas do módulo PIACDC, apresentadas na Tabela 11 da página 68, observa-se que a quantidade de combinações possíveis é $4{,}29 \times 10^9$, porém, dado que, para a entrada "INTRA" igual a zero, a saída é independente do resto das entradas, então a quantidade de combinações válidas é $2{,}15 \times 10^9$. Para determinar o tamanho do conjunto de treinamento foi utilizada a equação (39) da página 79, com γ=0,01 e f =0,15, sendo definida a quantidade de amostras como 64.286. Assim como para o CBP, para aumentar a confiabilidade dos

resultados, os experimentos foram duplicados, gerando-se dois conjuntos diferentes de amostras, porém do mesmo tamanho.

De forma similar ao feito para o CPB, na análise do PIACDC foram consideradas duas variações de tamanhos de conjuntos de casos de teste para treinamento. As condições de tamanho básicas para o módulo PIACDC foram as seguintes:

- Conjunto médio: na condição de conjunto médio foi considerado que o número de casos de teste fosse igual à quantidade de amostras acima calculada.

- Conjunto grande: foi considerado que o número de casos de teste fosse igual ao dobro da quantidade de amostras calculada.

Depois de se avaliar a cobertura dos casos de teste no *testbench* foi observada que a distribuição das ocorrências estava focada em duas categorias especificas, 8 e 32, como mostra a Tabela 24. Dado que juntas, as categorias 8 e 32 contêm uma fração do conjunto de casos ao redor do 98%, estas duas categorias foram consideradas como majoritárias. Nesta tabela observam-se na primeira coluna as categorias utilizadas no *testbench* do PIACDC com ajuste, na segunda e terceira colunas se observam as distribuições para dois conjuntos de tamanho médio (como forma de aumentar a confiabilidade), e na quarta e quinta colunas se observam as distribuições, também, para os dois conjuntos de tamanho grande. Em todos os conjuntos observa-se que existe a mesma tendência de dominância das categorias 8 e 32.

Tabela 24 - Distribuição dos eventos de cobertura para o módulo PIACDC

Categorias	C. médio 1 (%)	C. médio 2 (%)	C. grande 1 (%)	C. grande 2 (%)
5	0,36	0,44	0,39	0,36
6	0,62	0,67	0,71	0,65
8	54,72	54,48	57,65	52,96
17	0,18	0,16	0,13	0,19
18	0,40	0,29	0,33	0,36
32	43,72	43,96	40,79	45,48

Como esta concentração poderia fazer com que as categorias dominantes mascarassem a presença das outras nos modelos resultantes da aprendizagem, então, foram acrescentadas à análise outras variedades de tamanho apresentadas a seguir:

- Conjunto médio balanceado a 50%: nesta condição, as categorias majoritárias do conjunto médio foram reduzidas até ficarem com 50% do tamanho total do conjunto de treinamento.
- Conjunto médio balanceado a 70%: as categorias majoritárias do conjunto médio foram reduzidas até ficarem com 70% do tamanho total do conjunto de treinamento.
- Conjunto grande balanceado a 50%: as categorias majoritárias do conjunto grande foram reduzidas até ficarem com 50% do tamanho total do conjunto de treinamento.
- Conjunto grande balanceado a 70%: as categorias majoritárias foram do conjunto grande reduzidas até ficarem com 70% do tamanho total do conjunto de treinamento.

5.3.2.1.3 Técnicas de aprendizado

A análise do módulo PIACDC é feita sobre as mesmas técnicas apresentadas na página 80 para módulo CBP, ou seja, *data mining*, *data mining* sem corte e SVM. A técnica de redes Bayesianas não foi utilizada devido ao fato de que a ferramenta GENIE/SMILE não conseguiu realizar o aprendizado em nenhuma das condições de treinamento. Este problema aconteceu porque as redes Bayesianas requerem para os seus cálculos de probabilidade, matrizes com valores das probabilidades conjuntas. No caso do processamento dos modelos do PIACDC, matrizes de mais 4098x4098x4098 (= 68.820.189.192) células são requeridas, valor superior à capacidade da ferramenta GENIE/SMILE.

5.3.2.1.4 Os testbenches

Os experimentos com o módulo PIACDC foram realizados com os *testbenches* apresentados na coluna 1 da Tabela 25, com o intuito de confrontar as técnicas de aprendizado nas condições de tamanho especificadas.

Tabela 25 - Listado de *testbenches* realizados para o PIACDC

Código de identificação	Condição de tamanho	Técnica de aprendizado
TB_PIACDC_1	Conjunto médio	*Data mining*
TB_PIACDC_2	Conjunto médio	*Data mining*
TB_PIACDC_3	Conjunto grande	*Data mining*
TB_PIACDC_4	Conjunto grande	*Data mining*
TB_PIACDC_5	Conjunto médio	*Data mining* sem corte
TB_PIACDC_6	Conjunto médio	*Data mining* sem corte
TB_PIACDC_7	Conjunto grande	*Data mining* sem corte
TB_PIACDC_8	Conjunto grande	*Data mining* sem corte
TB_PIACDC_9	Conjunto médio	SVM
TB_PIACDC_10	Conjunto médio	SVM
TB_PIACDC_11	Conjunto grande	SVM
TB_PIACDC_12	Conjunto grande	SVM
TB_PIACDC_13	Conjunto médio balanceado 50	*Data mining*
TB_PIACDC_14	Conjunto médio balanceado 50	*Data mining*
TB_PIACDC_15	Conjunto médio balanceado 50	*Data mining*
TB_PIACDC_16	Conjunto médio balanceado 50	*Data mining*
TB_PIACDC_17	Conjunto médio balanceado 50	*Data mining* sem corte
TB_PIACDC_18	Conjunto médio balanceado 50	*Data mining* sem corte
TB_PIACDC_19	Conjunto médio balanceado 50	*Data mining* sem corte
TB_PIACDC_20	Conjunto médio balanceado 50	*Data mining* sem corte
TB_PIACDC_21	Conjunto médio balanceado 50	SVM
TB_PIACDC_22	Conjunto médio balanceado 50	SVM
TB_PIACDC_23	Conjunto médio balanceado 50	SVM
TB_PIACDC_24	Conjunto médio balanceado 50	SVM
TB_PIACDC_25	Conjunto médio balanceado 70	*Data mining*
TB_PIACDC_26	Conjunto médio balanceado 70	*Data mining*
TB_PIACDC_27	Conjunto médio balanceado 70	*Data mining*
TB_PIACDC_28	Conjunto médio balanceado 70	*Data mining*
TB_PIACDC_29	Conjunto médio balanceado 70	*Data mining* sem corte
TB_PIACDC_30	Conjunto médio balanceado 70	*Data mining* sem corte
TB_PIACDC_31	Conjunto médio balanceado 70	*Data mining* sem corte
TB_PIACDC_32	Conjunto médio balanceado 70	*Data mining* sem corte
TB_PIACDC_33	Conjunto médio balanceado 70	SVM
TB_PIACDC_34	Conjunto médio balanceado 70	SVM
TB_PIACDC_35	Conjunto médio balanceado 70	SVM
TB_PIACDC_36	Conjunto médio balanceado 70	SVM

Para cada combinação de tamanho do conjunto de treinamento e técnica de aprendizado, dois *testbenches* conformados por diferentes exemplos são adotados, para aumentar a confiabilidade dos resultados; por exemplo, os *testbenches* TB_PIACDC_1 e TB_PIACDC_2 são realimentados com modelos construídos com conjuntos com tamanho médio e técnica de aprendizado de *data mining*, porém com exemplos de treinamento diferentes.

Os *testbenches* com o módulo PIACDC foram executados considerando como metas para os eventos de cobertura (descritos na página 69), 22.000 ocorrências para os eventos 1 e 2, ; 40.000 ocorrências para os eventos 3 e 5, e 1.280.000 ocorrências

para os eventos 4 e 6. Estes valores foram escolhidos após executar o *testbench* com geração aleatória sem ajuste para diferentes valores proporcionais, observando o número de casos de teste necessários para atingir os alvos individuais dos eventos de cobertura. A Tabela 26 apresenta as porcentagens de casos de teste para cada evento com respeito ao último evento em atingir o alvo de observação. Para o modelo de cobertura do PIACDC se observou que os eventos associados a coeficiente AC (eventos 3 e 5) são mais frequêntes que os eventos associados as coeficientes DC, então foram valores de meta maiores para os eventos 3 e 5. Para a análise do tamanho de amostras foi definido um fator de multiplicação para equalizar as metas dos diferentes eventos. O valor de proporção escolhido foi 2.000 por ser o primeiro caso a mostrar sinais de estabilização o qual resulta nos valores de meta apresentados no começo deste parágrafo. Os experimentos do PIACDC foram executados numa estação de trabalho com um processador Intel CORE2DUO, com quatro Gigabytes de memória DDR-2 e executando o sistema operacional Linux OpenSUSE 10.3.

Tabela 26 - Porcentagem dos casos de teste gastos para atingir as metas de cobertura

Fator Multp. Meta	Evento 1 (%)	Evento 2 (%)	Evento 3 (%)	Evento 4 (%)	Evento 5 (%)	Evento 6 (%)
x1000	50,78	24,35	42,66	26,50	100	35,14
x2000	52,73	24,60	43,12	27,04	100	35,20
x10000	52,33	24,47	43,53	27,38	100	35,30

Os valores dos resultados apresentados nas seções seguintes correspondem ao valor médio dos modelos e *testbenches* realizados com as mesmas condições de tamanho do conjunto de treinamento e técnica de aprendizado; por exemplo, os resultados dos modelos e *testbenches* indicados como de conjunto médio e *data mining* correspondem ao valor médio dos resultados associados aos *testbenches* TB_PIACDC_1 e TB_ PIACDC_2 e aos modelos usados para a realimentação da cobertura.

5.3.2.2 Resultados do treinamento

Os modelos obtidos no aprendizado com as técnicas de *data mining*, *data mining* sem corte e SVM são avaliados nesta seção em função dos erros de classificação observados. Para o cálculo dos erros de classificação foi utilizado um conjunto

padrão de classificação composto por 257.144 casos de teste com suas respectivas categorias de cobertura avaliadas. Os resultados sobre os conjuntos de treinamento são avaliados através do calculo do erro observado ao classificar o conjunto padrão com cada um dos modelos aprendidos; erros de FP e FN são computados em cada uma das categorias de cobertura.

A Tabela 27 mostra os valores médios de erro de classificação para as condições de treinamento consideradas neste caso de estudo. A tabela está organizada da seguinte maneira: a primeira coluna apresenta a variedade do conjunto de treinamento utilizada; as segunda, terceira e quarta colunas apresentam a porcentagem de erro de classificação para *data mining*, *data mining* sem corte e SVM, respectivamente.

Tabela 27 - Erro de classificação dos modelos de aprendizado do PIACDC

Condição	***Data mining* (%)**	***Data mining* sem corte (%)**	**SVM (%)**
C. médio	2,03	2,03	1,84
C. grande	1,96	1,96	1,79
C. médio bal 70	2,03	2,06	2,15
C. grande bal 70	1,96	1,96	2,14
C. médio bal 50	2,03	2,01	2,53
C. grande bal 50	1,96	1,96	2,49

A Tabela 28 apresenta as porcentagens de FP e FN obtidas depois de realizar o treinamento com o conjunto de classificação padrão. A tabela está organizada da seguinte forma: a primeira coluna apresenta a variedade de conjunto de treinamento; a segunda coluna apresenta as categorias de cobertura apontadas pelos modelos de aprendizado quando da classificação com os casos de teste; a terceira e quartas colunas apresentam, respectivamente, a porcentagem de erro de FPs e FNs para a técnica de *data mining*; o mesmo padrão é seguido nas colunas 5/6 e 7/8, para as técnicas de *data mining* sem corte e SVM.

Tabela 28 - Erros de falso positivo e falso negativo do PIACDC

Varied.	Cat.	*Data mining*		*Data mining* sem corte		SVM	
		F.P. (%)	F.N. (%)	F.P. (%)	F.N. (%)	F.P. (%)	F.N. (%)
C. médio	5	59,71	61,28	59,68	63,10	98,26	65,62
	6	35,42	27,75	35,48	26,78	83,53	44,06
	8	1,40	1,13	1,40	1,13	0,94	27,47
	17	71,61	89,81	72,84	88,96	100,00	100,00
	18	27,79	21,14	27,57	20,97	94,83	24,61
	32	1,45	1,80	1,45	1,80	48,56	2,29
C. grande	5	46,84	72,19	46,38	69,07	99,89	50,00
	6	35,72	13,29	34,63	14,80	71,39	49,15
	8	1,40	1,13	1,40	1,13	1,09	24,33
	17	58,35	86,41	58,39	84,47	100,00	100,00
	18	27,72	18,26	27,58	17,98	92,50	29,96
	32	1,45	1,80	1,45	1,80	40,88	2,39
C. médio bal 70	5	59,70	61,23	59,67	63,05	0,00	76,63
	6	35,41	27,77	35,47	26,81	0,00	65,82
	8	1,40	1,13	1,40	1,13	2,56	23,58
	17	71,61	89,81	72,84	88,96	0,00	89,93
	18	27,79	21,14	27,57	20,97	0,00	77,97
	32	1,45	1,80	1,45	1,80	48,57	2,29
C. grande bal 70	5	46,80	72,14	46,31	69,02	0,00	76,05
	6	35,72	13,32	34,64	14,80	0,00	56,26
	8	1,40	1,13	1,40	1,13	1,09	24,32
	17	58,35	86,41	58,39	84,47	0,00	89,75
	18	27,72	18,26	27,58	17,98	0,00	77,92
	32	1,45	1,80	1,45	1,80	40,88	2,39
C. médio bal 50	5	59,68	61,23	59,64	63,05	0,00	76,64
	6	35,41	27,75	35,46	26,78	0,00	55,47
	8	1,40	1,13	1,40	1,19	0,94	27,48
	17	71,61	89,81	72,84	88,96	0,32	89,93
	18	27,79	21,14	27,57	20,97	0,00	77,96
	32	1,45	1,80	1,53	1,79	48,57	2,29
C. grande bal 50	5	46,84	72,19	46,28	69,07	0,00	76,05
	6	35,72	13,29	34,63	14,80	0,00	56,27
	8	1,40	1,13	1,40	1,13	1,09	24,32
	17	58,35	86,41	58,39	84,47	0,00	89,75
	18	27,72	18,26	27,58	17,98	0,00	77,92
	32	1,45	1,80	1,45	1,80	40,88	2,39

5.3.2.3 Resultados dos *testbenches*

Os resultados da execução dos *testbenches* com ajuste na geração com o módulo PIACDC são apresentados na Tabela 29, em função da quantidade de casos de teste e do tempo gasto para atingir uma determinada cobertura. Adicionalmente os resultados dos *testbenches* com ajuste são comparados entre si, com respeito aos resultados de redução dos parâmetros acima.

A Tabela 29 está organizada da seguinte forma: a primeira coluna apresenta a técnica de aprendizado usada no treinamento dos modelos; a segunda coluna apresenta as variedades de tamanho dos conjuntos de treinamento; a terceira coluna apresenta a quantidade de casos de teste gastos na execução do *testbench* para se atingir as coberturas definidas na página 106; a quarta coluna apresenta o tempo de execução do *testbench;* as quinta e sexta colunas apresentam a porcentagem de redução de casos de teste e de tempo, respectivamente, com respeito a *testbench* sem ajuste.

Tabela 29 - Resultados dos *testbenches* do PIACDC

Técnica	Condição	Casos de teste	Tempo (s)	Redução casos (%)	Redução tempo (%)
Testbench sem ajuste		5.122.305	2.383		
Data mining	C. médio	5.120.225	2.238	0,04	6,08
	C. grande	5.123.713	2.901	-0,03	-21,74
	C. médio bal 70	5.120.161	2.241	0,04	5,98
	C. grande bal 70	5.123.713	2.890	-0,03	-21,28
	C. médio bal 50	5.120.065	2.267	0,04	4,89
	C. grande bal 50	5.123.713	2.241	-0,03	5,98
Data mining sem corte	C. médio	5.120.129	2.423	0,04	-1,68
	C. grande	5.120.385	3.458	0,04	-45,11
	C. médio bal 70	5.120.097	2.450	0,04	-2,81
	C. grande bal 70	5.120.385	3.467	0,04	-45,49
	C. médio bal 50	5.120.065	2.432	0,04	-2,04
	C. grande bal 50	5.123.713	2.450	-0,03	-2,81
SVM	C. médio	5.119.937	3.988	0,05	-67,33
	C. grande	5.221.825	4.825	-1,94	-102,48
	C. médio bal 70	5.173.697	1.147	-1,00	51,89
	C. grande bal 70	5.163.777	1.132	-0,81	52,50
	C. médio bal 50	5.173.697	1.423	-1,00	40,31
	C. grande bal 50	5.173.697	1.407	-1,00	40,96

Dos resultados dos *testbenches* com ajuste realizado para o módulo PIACDC, observa-se que as gerações ajustadas por cobertura tiveram resultados muito próximos aos resultados dos *testbench* sem ajuste, provocando pouca redução (algumas vezes, com aumento) no número de casos de teste e tempo de execução em todas as técnicas, como mostrado nas duas últimas colunas da Tabela 29. Pode-se observar tal fato pelos baixos percentuais ou valores negativos. Este efeito é explicado pelo fato de que os eventos que precisam de aceleração (Tabela 30) estão ligados aos componentes DC dos coeficientes da decodificação "INTRA" do MPEG-4. Como os blocos do quadro MPEG-4 são de 8x8 *pixels*, os coeficientes de DC são

operados a cada 64 casos de teste, tendo poucas chances de aceleração, fazendo com que os ganhos sejam baixos.

5.3.2.4 Análise dos resultados

De forma semelhante á análise dos resultados do CPB, visa-se três aspectos: a influência do tamanho do conjunto de treinamento, do balanceamento dos casos de teste e da técnica de aprendizado. Tais aspectos foram avaliados cruzando-se os resultados do treinamento e do *testbench* nas diferentes condições consideradas nos experimentos.

A Tabela 30 mostra a quantidade média de ocasiões em que os eventos de cobertura estão atrasados durante a execução dos *testbenches*, necessitando então de aceleração. Tem-se na primeira coluna os eventos de cobertura, enquanto, na segunda, as categorias às quais eles pertencem são listadas, e, por fim, são apresentadas na terceira coluna, as médias das ocasiões em que os eventos de cobertura correspondentes estavam atrasados. A tabela mostra que os eventos 3 e 5, associados à ocorrência dos valores dos coeficientes DC, apresentam os maiores níveis de atraso, enquanto os eventos 4 e 6 associados aos coeficientes AC nunca estão atrasados. Já os eventos 1 e 2, associados a direção de predição utilizada, mostra que o evento associado à predição no sentido de bloco A apresenta atraso, enquanto o evento da predição do bloco B não está atrasado em nenhuma ocasião.

Tabela 30 - Quantidade média de ocasiões de atraso no *testbench* do PIACDC

Eventos	Categorias	Média de ocasiões em atraso
Ev. 1	5, 17	1.111.727
Ev. 2	6, 18	0
Ev. 3	5, 6	2.163.355
Ev. 4	8	0
Ev. 5	17, 18	2.955.047
Ev. 6	32	0

Dado que as categorias 8 e 32 estão associadas exclusivamente a eventos que nunca requerem aceleração, elas foram desconsideradas na análise dos resultados, já que não têm efeito durante a execução do *testbench*; observa-se que estas categorias possuem as maiores distribuições dentro dos conjuntos de treinamento, como foi apresentado na Tabela 24. A análise dos resultados de treinamento dos

modelos do PIACDC foram realizadas com a avaliação qualitativa apresentada na página 88.

5.3.2.4.1 Comparativo segundo o tamanho do conjunto

A presente seção apresenta através das figuras 33, 34 e 35 o comparativo dos resultados obtidos com os modelos e testbenches construídos para CDV, em esta seção mostra-se a influência do parâmetro tamanho de conjunto de treinamento. As figuras são mais ilustrativas das tendências verificadas ao se modificar os parâmetros de comparação, porém, dados específicos estão tabelados e podem ser encontrados no Apêndice 2 (A2.1.). Estas figuras apresentam em forma de barras azuis os resultados do comparativo qualitativo dos modelos aprendidos e com linhas as diferenças entre as reduções de quantidade de casos de teste (vermelho) e tempo de execução (verde). Estas figuras estão organizadas da esquerda para direita, apresentando-se as seguintes comparações de uso: do conjunto grande ao invés do conjunto médio, do conjunto grande com balanceamento 70% ao invés do conjunto médio com balanceamento a 70% e do conjunto grande com balanceamento 50% ao invés do conjunto médio com balanceamento 50%.

Na Figura 33 as comparações realizadas com os modelos e *testbenches* construídos com *data mining* são apresentadas. As comparações entre os modelos mostram que aqueles obtidos através de conjuntos de treinamento grandes são melhores. Entretanto para os resultados dos testbenches não é observada nenhuma diferença na redução de quantidade de casos de teste, enquanto que, nas comparações da redução do tempo de execução, observam-se maiores diferenças nos casos sem balanceamento e com balanceamento 70%. Nesta análise ficam refletidas as poucas chances de aceleração, que terminam por mascarar o efeito dos *testbenches* com geração ajustada por cobertura, sendo os resultados, então, semelhantes aos obtidos com o *testbench* sem ajuste. Isto faz também com que as melhoras nos modelos aprendidos não sejam observadas nos resultados dos *testbenches*. Já a comparação em termos de ganhos de tempo resulta pouco conclusiva quanto à influência do tamanho conjunto no treinamento; o único que se observa é que os conjuntos menores tendem a fornecer melhores reduções de tempo, indicando a forte influência do processamento do ajuste com respeito ao processamento próprio do *testbench*.

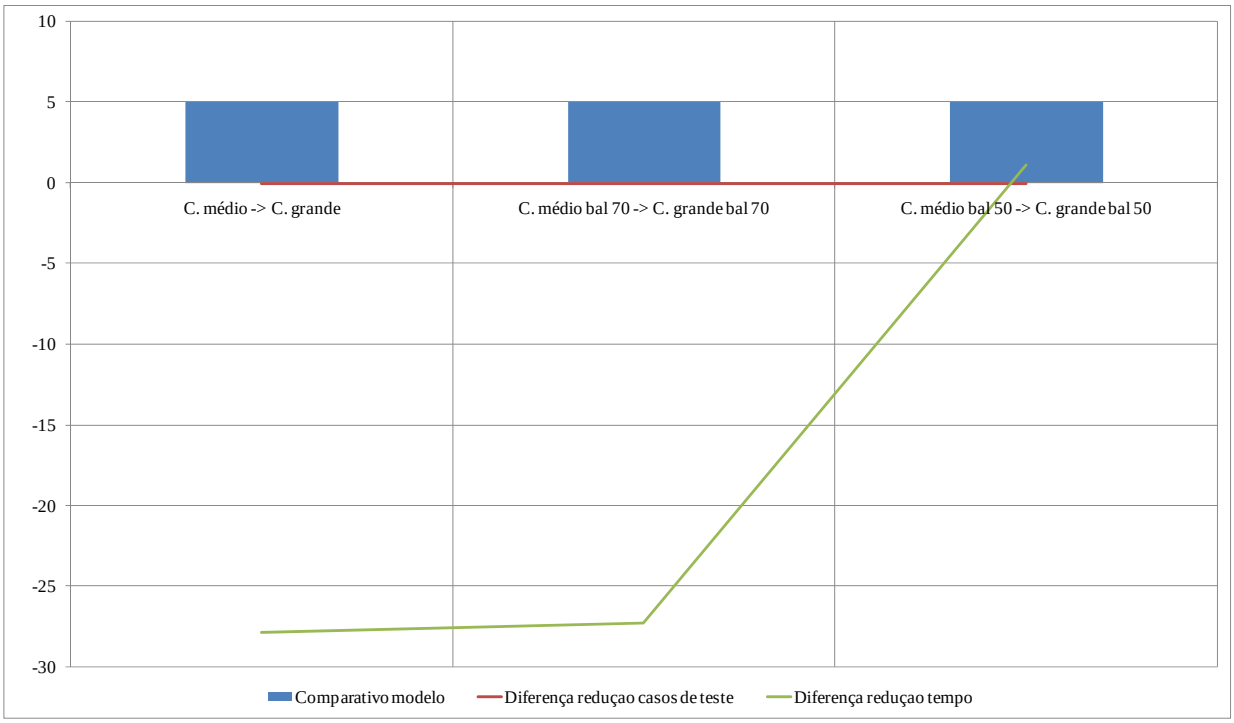

Figura 33 - Comparativo dos resultados do treinamento com *data mining* e do *testbench* entre os conjuntos de tamanho médio e grande, para o PIACDC

A Figura 34 apresenta as comparações feitas entre os resultados dos modelos e *testbenches* construídos com a técnica *data mining* sem corte. Nesta figura se observa uma tendência semelhante à apresentada na Figura 33, onde todos os resultados dos modelos são melhores com os conjuntos grandes; nos resultados dos *testbenches*, enquanto a redução de casos de teste é praticamente zero, as reduções de tempo de execução são melhores para os conjuntos de tamanho médio sem balanceamento e balanceamento 70%. Esta tendência reforça a observação da forte influência do tempo de processamento do modelo de ajuste no tempo total de execução do *testbench* e das poucas chances de aceleração observadas no *testbench*.

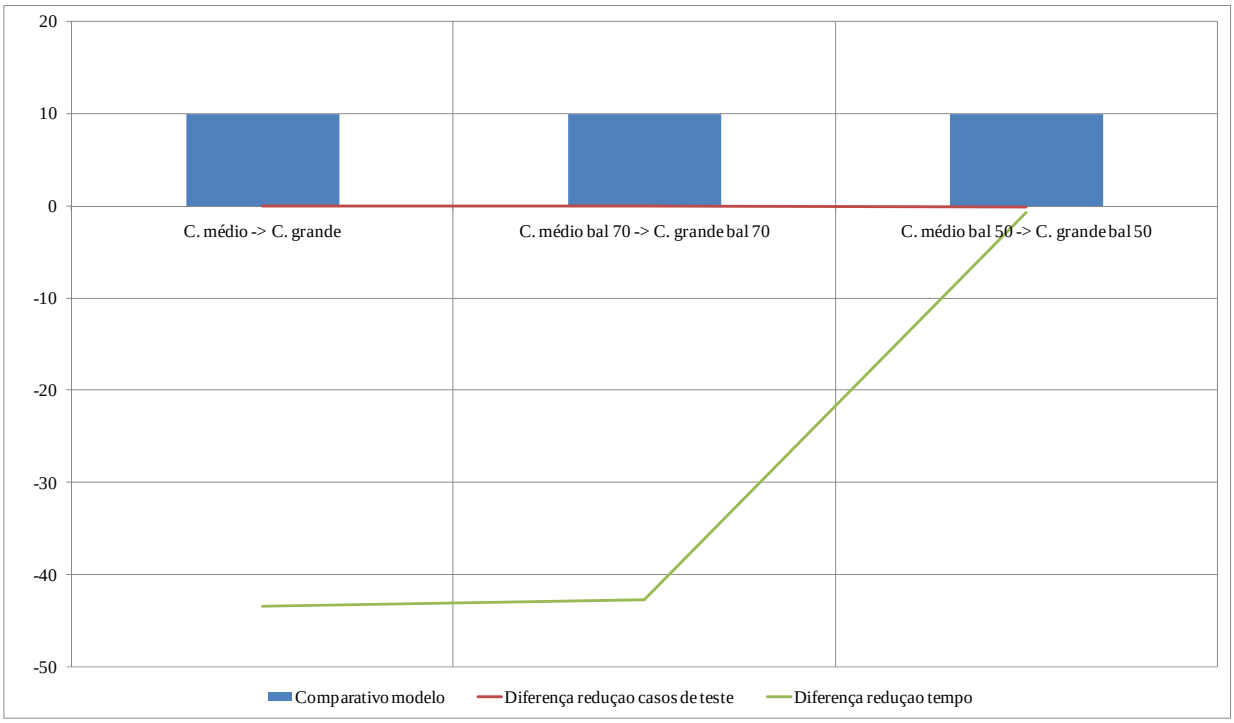

Figura 34 - Comparativo dos resultados do treinamento com SVM e do *testbench* entre os conjuntos de tamanho médio e grande, para o PIACDC

Na Figura 35, observam-se as comparações realizadas com os modelos e testbenches construídos com a técnica SVM. Nesta figura, percebe-se que para todos os conjuntos considerados, os resultados dos modelos mostram que estes são melhores para os conjuntos grandes. Do ponto de vista dos resultados dos *testbenches*, observa-se que para os conjuntos sem balanceamento, é pior se usar o conjunto grande. Para os conjuntos com balanceamento os resultados são praticamente equivalentes tanto para a redução da quantidade de casos de teste quanto para a redução de tempo de execução. Este efeito é devido ao fato de que nos conjuntos sem balanceamento os modelos SVM são consideravelmente maiores que os obtidos com o uso do balanceamento.

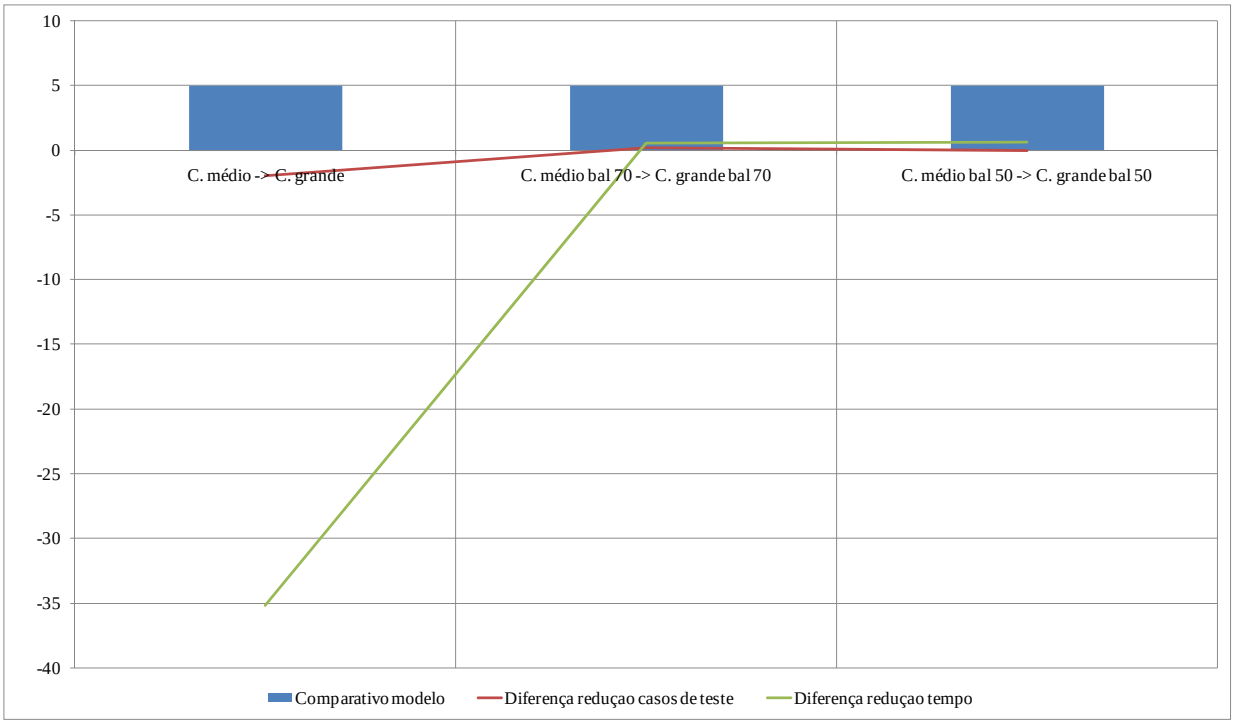

Figura 35 - Comparativo dos resultados do treinamento com SVM e do *testbench* entre os conjuntos de tamanho médio e grande, para o PIACDC

5.3.2.4.2 Comparativo segundo o balanceamento

Na presente seção, o comparativo segundo o nível de balanceamento é apresentado através das figuras 36, 37 e 38, enquanto dados tabelados podem ser acessados no Apêndice 2 (A2.2.). Cada uma destas figuras apresenta através de barras azuis os resultados do comparativo qualitativo dos modelos aprendidos, e através de linhas a diferença entre as reduções de quantidade de casos de teste (linha vermelha) e tempo de execução (linha verde). Estas figuras apresentam, da esquerda para a direita, as seguintes comparações do uso: do conjunto médio com balanceamento 70% ao invés do conjunto médio sem balanceamento, do conjunto grande com balanceamento 70% ao invés do conjunto grande sem balanceamento, do conjunto médio com balanceamento 50% ao invés do conjunto médio com balanceamento 70% e do conjunto grande com balanceamento 50% ao invés do conjunto grande com balanceamento 70%.

A Figura 36 apresenta as comparações dos resultados dos modelos e *testbenches* construídos com a técnica de *data mining*. Nesta figura, pode-se observar que os modelos obtidos apresentam melhorias nos conjuntos médios e são equivalentes nos conjuntos grandes. Com respeito aos resultados dos testbenches se observa

que a diferença entre as reduções dos casos de teste é praticamente zero; isto devido às poucas chances de aceleração apresentadas neste caso de estudo. Entretanto com respeito à redução do tempo de execução, observa-se que a diferença é baixa exceto para o caso do uso do conjunto grande com balanceamento 50% ao invés do conjunto grande com balanceamento a 70%. Entretanto as diferenças de ganho de tempo mostraram-se não conclusivas com respeito os resultados dos modelos aprendidos.

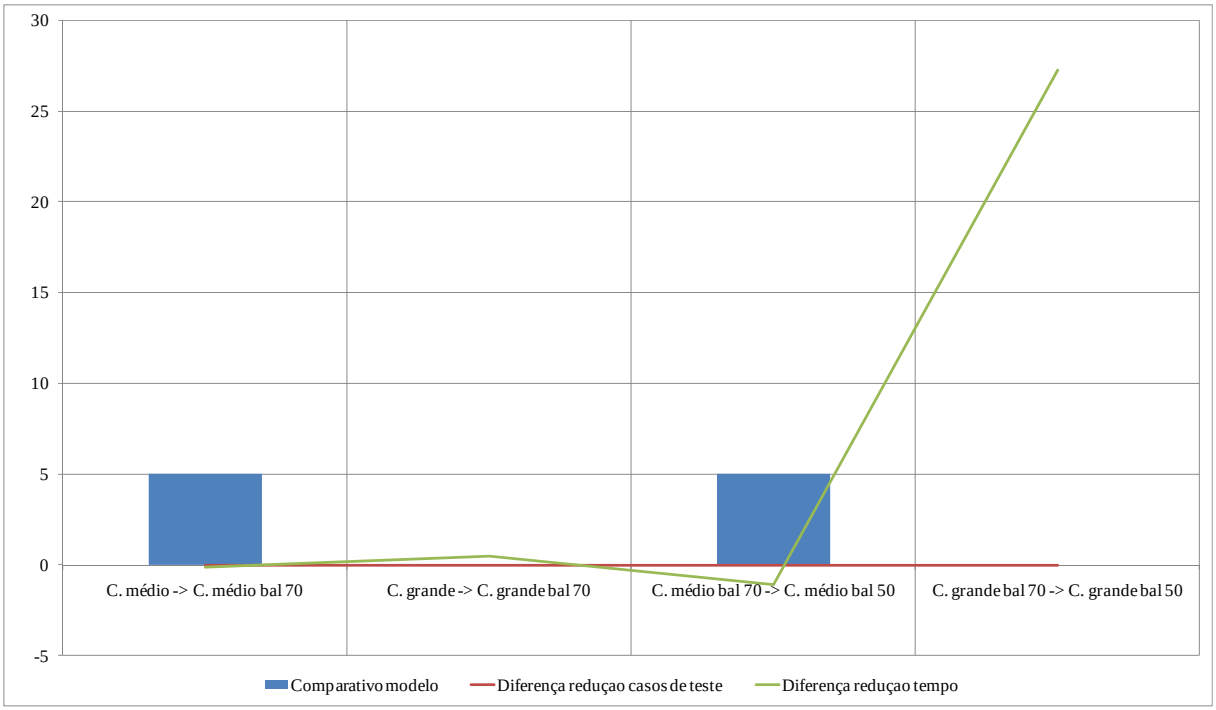

Figura 36 - Comparativo dos resultados do treinamento com *data mining* e do *testbench* entre os conjuntos com diferentes balanceamentos, para o PIACDC

A Figura 37 apresenta as comparações dos resultados dos modelos e *testbenches* construídos com a técnica de *data mining* sem corte. Os resultados apresentados mostram as mesmas tendências observadas na Figura 36; isto deve-se principalmente ao fato de que caso de estudo do PIACDC não apresenta muitas chances de aceleração.

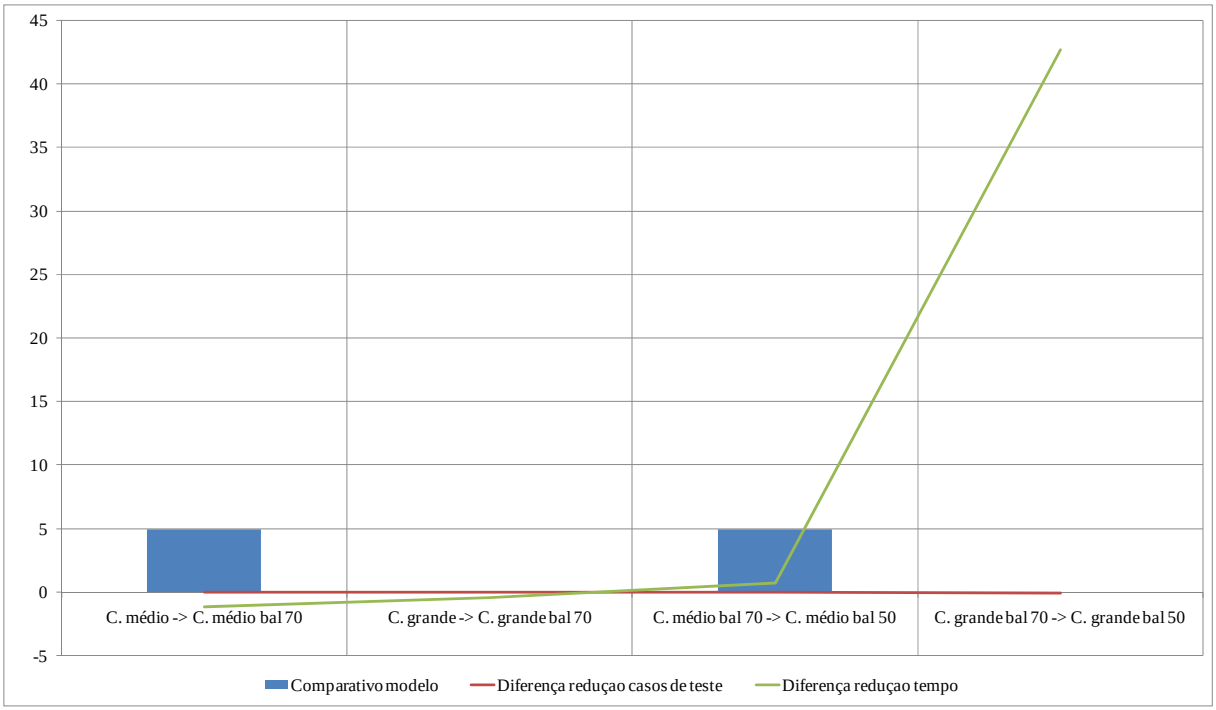

Figura 37 - Comparativo dos resultados do treinamento com *data mining* sem corte e do *testbench* entre os conjuntos com diferentes balanceamentos, para o PIACDC

A Figura 38 apresenta as comparações dos resultados dos modelos e *testbenches* construídos com a técnica de SVM. Nesta figura mostra-se que, para todos os modelos aprendidos, pode-se observar uma melhoria quando o nível de balanceamento é aumentado. Entretanto nos resultados de redução de quantidade de casos de teste, observam-se diferenças ao redor de zero; isto devido ao fato de que no caso de estudo do PIACDC não há muitas chances de aceleração. Com respeito às diferenças dos resultados do ganho de tempo, observa-se que os modelos SVM sem balanceamento tiveram uma maior número de vetores de suporte, precisando, então, de maior tempo de execução; resulta numa diferença positiva nas duas partes da figura quando os modelos com balanceamento de 70% são utilizados ao invés dos modelos sem balanceamento, porém na comparação entre os modelos com balanceamento 70% e 50% não apresentam a mesma tendência, portanto os resultados não são completamente conclusivos.

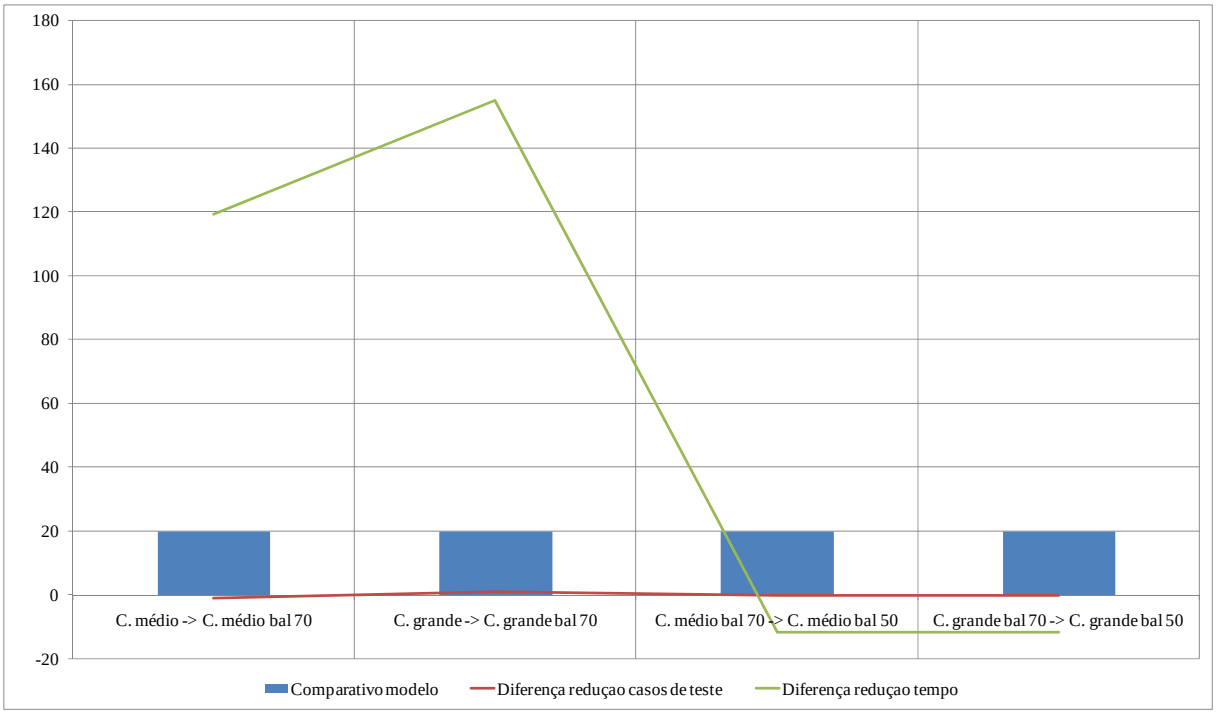

Figura 38 - Comparativo dos resultados do treinamento com SVM e do *testbench* entre os conjuntos com diferentes balanceamentos, para o PIACDC

5.3.2.4.3 Comparativo segundo a técnica de aprendizado

Esta seção apresenta os resultados o comparativo dos modelos obtidos no treinamento e execução dos *testbenches* do módulo PIACDC variando a técnica de aprendizado utilizada. Figuras são utilizadas para ilustrar o comportamente frente às mudanças de técnicas, enquanto valores numéricos podem ser encontrados nas tabelas do Apêndice 2 (A2.3.). Enquanto a Figura 39 mostra as diferenças quando a técnica de *data mining* sem corte é utilizada ao invés da técnica de *data mining*, a Figura 40 mostra as diferenças quando se compara a utilização da técnica SVM ao invés da técnica de *data mining*. Neste comparativo a técnica de *data mining* foi utilizada como referência porque ela tem mostrado melhor estabilidade nos resultados dos *testbenches*. As duas figuras apresentam em forma de barras azuis as comparações dos modelos aprendidos e em forma de linhas as diferenças dos resultados dos testbenches: na cor vermelha a diferença na redução de quantidade de casos de teste e na cor verde a diferença na redução de tempo de execução. Estas figuras apresentam, da esquerda para a direita, as comparações realizadas com os seguintes conjuntos: conjuntos médios sem balanceamento, conjuntos grandes sem balanceamento, conjuntos médios com balanceamento 70%, conjuntos

grandes com balanceamento 70%, conjuntos médios com balanceamento 50% e conjuntos grandes com balanceamento 50%.

A Figura 39 mostra a que a técnica de *data mining* sem corte apresenta melhores resultados de erro que a técnica de *data mining*, para todas as condições de treinamento. Isto ocorre dado que os modelos com corte utilizados tiram informação das categorias minoritárias necessárias para realimentar a informação de cobertura. Entretanto, as diferenças na quantidade de casos de teste indicam que as duas técnicas são semelhantes nos *testbenches*, e como mostra a Tabela 29 praticamente sem redução, devido as poucas chances de aceleração. Apesar disto, os resultados do ganho de tempo mostraram que os *testbenches* com *data mining* sem corte tiveram resultados piores (aumento do tempo de execução), especialmente para as condições de conjuntos grandes. Uma possível causa para este efeito é o fato de os modelos com *data mining* sem corte apresentarem mais detalhes e, portanto tendo um número maior de ramos que os modelos com *data mining*, requerendo um maior tempo de execução.

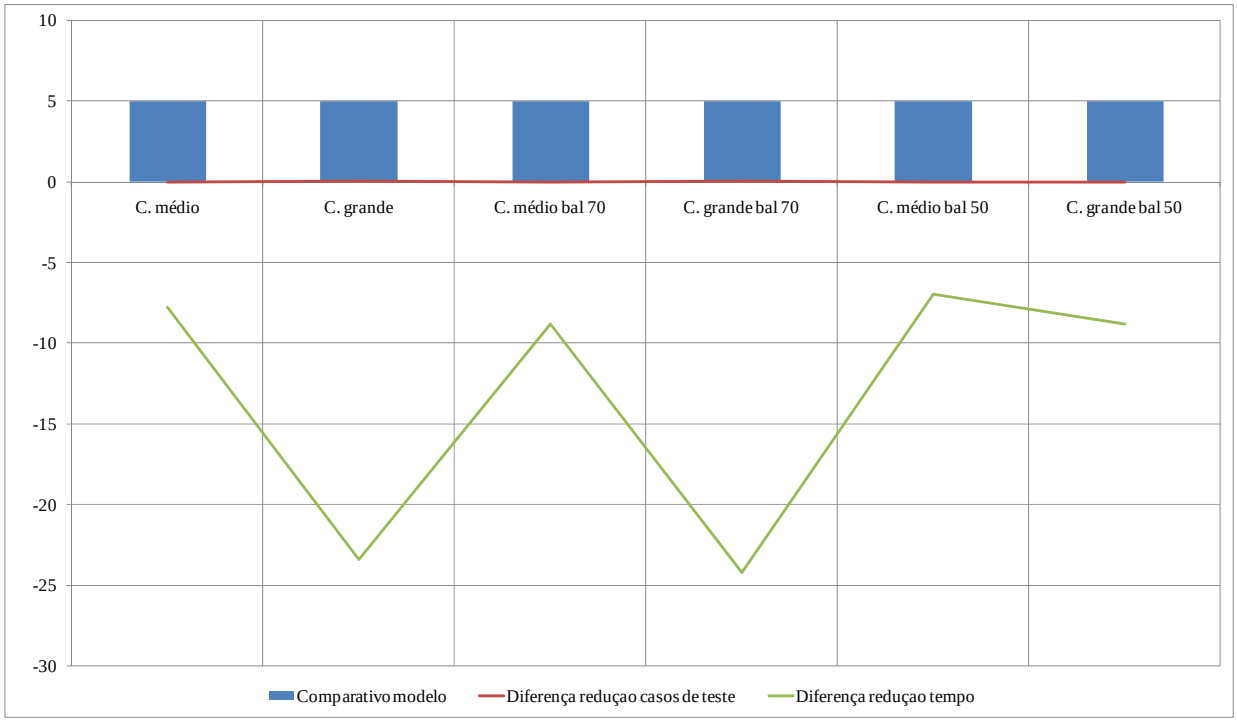

Figura 39 - Comparativo dos resultados do treinamento e do *testbench* para o CBP entre as técnicas de *data mining* sem corte e *data mining*

Nas comparações da Figura 40 observa-se que a técnica de SVM apresenta modelos piores que os gerados na técnica *data mining*, quando o conjunto de

treinamento não é balanceado; por outro lado, quando o balanceamento é utilizado, os modelos são melhores. Isto é devido a que a otimização do SVM é centrada nas categorias majoritárias quando o balanceamento não é utilizado, mostrando piores resultados de treinamento para as categorias minoritárias utilizadas no comparativo. Já quando o balanceamento é utilizado o SVM consegue otimizar também nas categorias minoritárias. Do ponto de vista dos ganhos em número de casos de teste, o SVM mostrou resultados iguais ou piores que os obtidos com *data mining*, porém com diferenças não significativas neste último caso; isto é devido às poucas chances de aceleração mostradas pelo *testbench*. Já em termos de tempo de execução dos *testbenches*, os resultados com SVM mostraram um aumento considerável com respeito à técnica de *data mining*, para as condições sem balanceamento; entretanto, nas condições com balanceamento, reduções ocorreram. Tal efeito é devido ao fato de que os modelos SVM sem balanceamento são significativamente maiores, enquanto ao número de vetores de suporte, que os modelos SVM com balanceamento, requerendo um maior tempo para a execução. Porém quando o balanceamento é aplicado, os modelos SVM têm o tamanho reduzido e são rapidamente executados, oposto ao efeito do balanceamento nas técnicas de *data mining* e *data mining* sem corte.

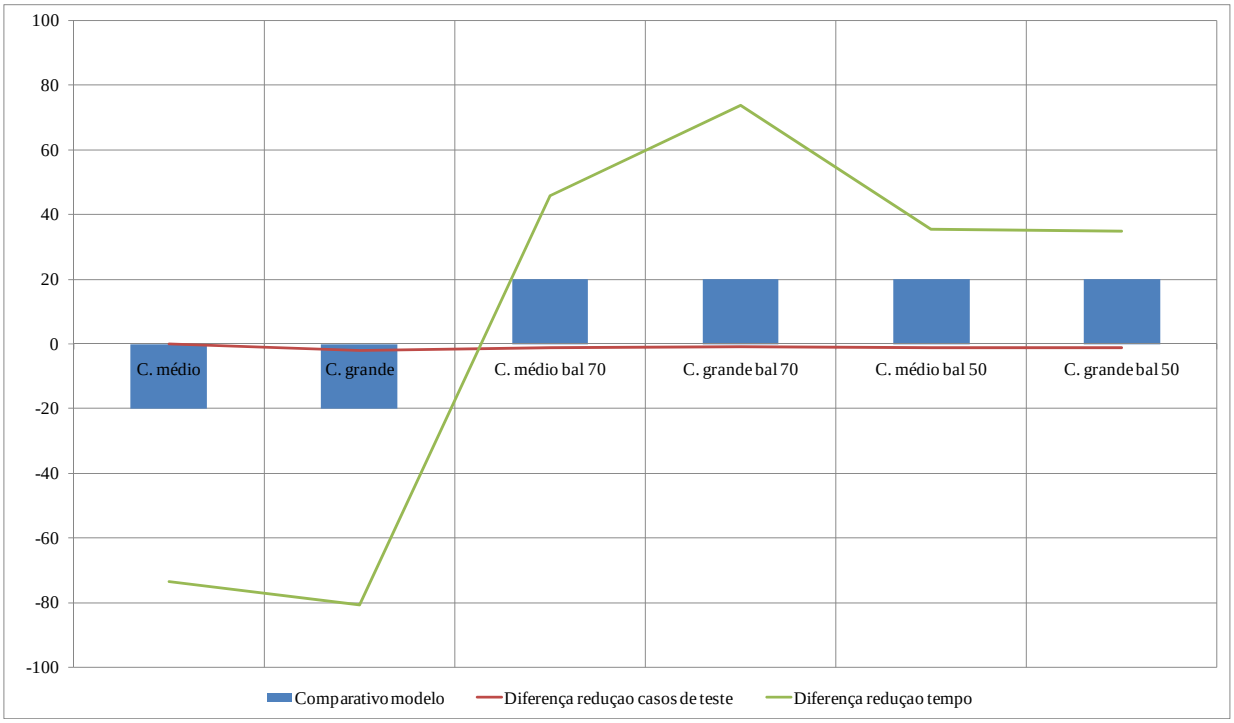

Figura 40 - Comparativo dos resultados do treinamento e do *testbench* para o CBP entre as técnicas de SVM e *data mining*

5.3.2.5 Conclusões da análise

Finalmente, depois de analisar as diferentes variações das condições de treinamento e técnicas de aprendizagem, observa-se que para o módulo PIACDC, o balanceamento e o aumento do tamanho do conjunto de treinamento tiveram um efeito positivo nos resultados da avaliação qualitativa dos erros de FP e FN. Porém, devido ao fato de a aceleração para os *testbenches* estar limitada a um caso de teste de cada sessenta e quatro, tais efeitos positivos não foram observados nos ganhos de casos de teste dos *testbenches*. Com respeito aos ganhos em termos de tempo, nas comparações das condições de balanceamento e tamanho de conjunto de treinamento não foi observada nenhuma tendência clara que permitisse tirar alguma conclusão, além de que este resultado denota a existência de forte componente aleatória associada à geração de estímulos.

Com respeitos às técnicas de aprendizado, observa-se que os modelos associados à técnica de *data mining* sem corte se mostram melhores que para *data mining*, em todas as condições. Os modelos com SVM são unicamente melhores quando o balanceamento de casos de teste é utilizado. Também, pode-se observar que os modelos de *data mining* sem corte têm a tendência de serem mais detalhados e

maiores com um maior número de ramos que os modelos de *data mining*, portanto apresentando tempos de execução maiores dos *testbenches*. Entretanto, os modelos com SVM são maiores, na medida em que têm mais vetores de suporte, quando os conjuntos de treinamento são maiores, fazendo com que, quando o balanceamento não é utilizado, o tempo de execução dos *testbenches* sejam mais longos que para com os modelos de *data mining*. Como o balanceamento reduz consideravelmente o tamanho dos conjuntos de treinamento, os modelos SVM tornam-se mais rápidos, fazendo que os ganhos de tempo ocorram quando comparados com os *testbenches* com *data mining*.

6 CONCLUSÕES

6.1 Conclusões do trabalho

A partir do estudo realizado sobre a aplicação da verificação funcional com geração ajustada por cobertura, em núcleos de comunicação e multimídia, observou-se que características particulares dos domínios de aplicação influenciam fortemente o sucesso do uso das técnicas de ajuste.

O tamanho do espaço de casos de teste foi identificado como um fator importante no sucesso do ajuste. Observou-se que para conjuntos de treinamento de tamanho comparável ao espaço de casos de teste, as técnicas de aprendizado e ajuste conseguem reduzir o tempo e quantidade de casos de teste necessários para alcançar as metas de cobertura. Quando os tamanhos dos conjuntos de treinamento são substancialmente menores que o espaço, ocorreram dificuldades no ajuste com as técnicas de *data mining*, redes Bayesianas e SVM.

As dificuldades encontradas no ajuste da cobertura estiveram relacionadas principalmente à distribuição das categorias associadas aos eventos de cobertura. A má distribuição reflete a tendência de certas categorias estarem atrasadas em termos de cobertura na execução dos *testbenches* sem ajuste; fornecem, assim, baixas quantidades de amostras ao se montar os conjuntos de treinamento. Isto faz com que, durante o treinamento, estas categorias tenham uma representatividade inferior nos modelos aprendidos, dificultando nesta medida, a identificação dos casos de teste necessários para acelerar estas categorias.

As dificuldades acima foram contornadas no presente estudo usando três abordagens: a distribuições nos conjuntos de treinamento foram mudadas através do balanceamento, foram aumentados os tamanhos dos conjuntos de treinamento e foi desabilitado o corte na técnica de *data mining*. Destas atitudes, conclui-se:

- O balanceamento mostrou-se com capacidade de melhorar os modelos aprendidos, fato demonstrado pela redução dos erros de falso positivo e falso negativo nas categorias minoritárias dos conjuntos de treinamento.

- O aumento do tamanho do conjunto de treinamento não foi conclusivo, já que só alguns modelos apresentaram melhorias.

- Já o ajuste na geração realizado com os modelos obtidos da *data mining* sem corte mostrou-se positivo, dado que reduziu os erros de classificação observados na avaliação dos modelos.

A execução dos *testbenches* do módulo STREAMPROC mostrou-se positivo em praticamente todas condições e técnicas, apesar de que com diferenças razoáveis no tempo de execução ou casos de teste. É um indicativo de que quando se tem o espaço de entrada pequeno as técnicas de CDV mostram um grande potencial de sucesso.

A execução dos *testbenches* dos módulos do MPEG-4 com os modelos aprendidos permitiu a observação de que, com o módulo CBP, foi possível melhorar os resultados dos *testbenches* quando os modelos aprendidos foram melhores, isto é, apresentaram menores erros de classificação. No caso do CPB, observou-se que as melhorias nos *testbenches* tiveram uma limitante devido ao fato que alguns dos eventos não podem ser ativados em todos os casos de teste. Este efeito foi observado no comparativo entre as técnicas de *data mining* e *data mining* sem corte, nas quais os melhores resultados das testbenches convergem ao mesmo valor. Os resultados com os *testbenches* do módulo CBP indicam que mesmo casos com espaço de entrada grande, CDV consegue reduções consideráveis de até de 44,49% no número de casos teste requeridos e de 42,65% no tempo de execução dos *testbenches*.

No caso do PIACDC, fica mais evidente a dificuldade de gerar os casos de teste desejado; porém, neste caso particular, os eventos atrasados mostraram estar associados aos coeficientes DC, que são operados cada 64 casos de teste, fazendo que as chances de aceleração fossem poucas. Estas limitantes de aceleração mascararam fortemente os efeitos dos modelos aprendidos nos resultados dos *testbenches*.

O uso de SVM mostrou em geral ser capaz de fornecer bons modelos aprendidos, porém, o esquema de filtragem criou ineficiências graves em várias situações. A SVM foi bem sucedida nos *testbenches* do STREAMPROC mostrando resultados

próximos à técnica de *data mining*, que apresentou os melhores resultados. Porém nos casos do MPEG-4 os resultados da SVM foram ruins, pois as técnicas de filtragem requerem a avaliação de um grande número de casos de teste até conseguir encontrar os casos correspondentes com as categorias minoritárias.

Os *testbenches* ajustados com modelos construídos com redes Bayesianas apresentaram os piores resultados nos *testbenches* com STREAMPROC. No CBP, ao invés de redução, o número de casos de teste e o tempo de execução aumentaram. A razão para isto foi uma limitante no aprendizado que considera as variáveis numéricas contínuas como se fossem simples variáveis discretas. Este efeito é reduzido quando o tamanho do conjunto é comparável com tamanho do espaço, como é o caso do STREAMPROC. Entretanto, quando o tamanho não é da mesma ordem, esta técnica falha em aprender as características das categorias minoritárias. Adicionalmente na inferência inversa, as redes Bayesianas sempre indicam algum caso de teste mesmo que elas não consigam estabelecer a categoria associada, fazendo que o ajuste seja feito sem controle.

Os resultados dos *testbenches* ajustados com modelos com *data mining* mostraram as melhores reduções de tempo e número de casos de teste nos casos de estudo STREAMPROC e CBP, enquanto que no PIACDC, estes valores foram próximos dos resultados das outras técnicas devido às poucas chances de aceleração. A *data mining* apresentou uma rápida inferência inversa que tem a característica de indicar claramente quando não há representatividade no modelo para a categoria a acelerar, permitindo que seja imediatamente aplicado um caso de teste aleatório. Esta característica faz com que o resultado do *testbench* seja no pior dos casos igual ao *testbench* sem ajuste. A variação da *data mining* sem corte mostrou-se vantajosa, uma vez que evita a eliminação de detalhes dos modelos aprendidos, relevantes para o ajuste por cobertura da geração de casos de teste.

6.2 Trabalhos futuros

Do presente trabalho foram identificadas as seguintes propostas de trabalhos futuros:

- A avaliação do CDV em sistemas com comportamentos de natureza estocástica requerem uma nova abordagem de geração de casos de teste ajustada por

cobertura, dado que esta característica representa um problema de controlabilidade, não considerado por nenhum dos trabalhos encontrados na literatura.

- O presente trabalho pode ser estendido a outros domínios de aplicação com características de cobertura e do espaço de casos de teste diferentes às já tratadas.

6.3 Publicações realizadas

Como resultado das atividades realizadas, os seguintes artigos foram apresentados durante o desenvolvimento do doutorado:

- "*Comparing two testbench methods for hierarchical functional verification of a bluetooth baseband adaptor*" apresentado na conferência "*International conference on Hardware/software codesign and system synthesis (CODES+ISSS)*" na cidade de Nova Iorque (Estados unidos) no ano 2005 (17).

- "*Functional Verification of Communication Systems based on Modular Coverage*" apresentado no evento, "8th IEEE Latin-American Test Workshop (LATW)" na cidade de Cuzco (Peru) no ano 2007 (20).

- "*Domain Oriented Training for Data Mining Coverage Driven Verification*" apresentado na conferencia "Design and Verification Conference (DVcon)" na cidade de San Jose (Estados unidos) no ano 2009 (56).

- "*Support Vector Machine Coverage Driven Verification for Communication Cores*" apresentado na conferência "International Conference on Very Large Scale Integration (VLSI-SoC)" na cidade de Florianopolis no ano 2009 (57).

7 Referências

(1) KEUTZER, K. NEWTON, A. RABAEY, J. SANGIOVANNI-VINCENTELLI, **A. System level design: orthogonalization of concerns and platform-based design.** In: IEEE transactions on computer-aided design of circuits and systems, v.19, n. 12, p. 1523 - 1543, 2000.

(2) IBM Microelectronics. The coreconnect bus architecture. 1999.

(3) ARM. AMBA AHB cycle level interface specification, design methodology and tools. 2003.

(4) OCP-IP. **Open core protocol specification**. Disponível em: http://www.ocpip.org. Acesso em: maio 2009.

(5) WILE, B. GROSS, J. ROESNER, W. **Comprehensive functional verification**. San Francisco. Morgan Kaufmann. 2005. 702p (Systems on silicon series).

(6) SYNOPSYS. **OpenVera language reference manual**, Disponível em: http://www.open-vera.com. Acesso em: maio 2009.

(7) CADENCE. **Testbuilder reference manual**. Disponível em: http://www.testbuilder.net. Acesso em: março 2006.

(8) CADENCE. **Incisive data sheet**. Disponível em http://www.cadence.com/products/fv/enterprise_manager/pages/default.aspx. Acesso em: maio 2009.

(9) PIZIALI, A. Functional verification coverage measurement and analysis. Boston. Kluwer Academic. 2004. 230p.

(10) HEKMATMPOUR, A. COULTER, J. **Coverage-directed management and optimization of random functional verification**. In: IEEE proceedings international test conference, 2003, v 1, pp 148 – 155.

(11) SMITH, J. BARTLEY, M. FOGARTY, T. **Microprocessor design verification by two-phase evolution of variable length tests.** In: IEEE international conference on evolutionary computing, 1997, pp 453 - 458.

(12) CORNO, F. SANCHEZ, E. SONZA, M. **Evolutionary test program for generation: a case of study.** In: IEEE design test of Computers, 2004, pp 102- 109.

(13) WAGNER,I. BERTACCO,V. AUSTIN,T. **Stresstest: an automatic approach to test generation via activity monitors.** In: IEEE design automation conference (DAC), 2005, p 783 – 788.

(14) HSEEH, H. EDER, K. **Test directive generation for functional verification closure using inductive logic programming.** In: IEEE high level design validation and test workshop (HLDVT), 2006 pp 11-18.

(15) FINE, S. ZIV, A. **Coverage directed test generation for functional verification using bayesian networks.** In: IEEE design automation conference (DAC), 2003, pp 286 – 291.

(16) BRAUN, M. ROSENSTIEL, W. SCHUBERT, K. **Comparison of bayesian networks and data mining for coverage directed verification.** In: IEEE high level design verification and test workshop (HLDVT), 2003 pp 91 - 95,

(17) ROMERO, E. STRUM, M. WANG, J. **Comparing two testbench methods for hierarchical functional verification of a bluetooth baseband adaptor.** In: ACM international conference on hardware/software codesign and system synthesis (CODES+ISSS), 2005. pp 327 - 332.

(18) HENNESSY, J. PATTERSON, D. **Computer Architecture: A Quantitative Approach.** Dordrecht: Morgan Kaufmann, 2006. 704p.

(19) HARRIS, D. HARRIS, S. **Digital design and computer architecture**. Dordrecht: Morgan Kaufmann, 2007. 592p.

(20) ROMERO, E. IGUCHI, K. STRUM, M. WANG. **Functional verification of communication systems based on modular coverage**. In:8th IEEE latin-american test workshop (LATW), 2007 pp 37 - 42.

(21) VITULLO, F. SAPONARA, PETRI, E. CASULA, M. FANUCCI, L. MARUCCIA, G. LOCATELLI, R. COPPOLA, M. **A reusable coverage-driven environment for network-on-chip communication in embedded system platforms.** In: Workshop Intelligent solutions in Embedded Systems (WISES), 2009, pp 71 - 77.

(22) SILVEIRA, G. DA SILVA, K. MELCHER E. **Functional verification of an MPEG-4 decoder design using random constrained movie generation.** In: IEEE conference on Integrated circuits and systems design (SBCCI), 2007, pp 360 - 364.

(23) COLIN Y. SONG, C. JUNSHE A. FEI, H. QIFEI, F. **A network based functional verification method of IEEE 1394a PHY core.** In: IEEE Symposium on VLSI (ISVLSI), 2008, pp 245 - 250.

(24) VAPNIK, W. **Statistical learning theory**. Dordrecht: John Wiley, 1998. 736p.

(25) BURGES, C. **A tutorial on support vector machine for pattern recognition.** In: Data mining and knowledge discovery, v.2, n.2, pp 121 - 167, 1998.

(26) YUAN, J. PIXLEY, C. AZIZ, A. **Constraint-based verification**. Dordrecht: Springer. , 2009. 272p.

(27) JERINIC,V. MÜLLER, D. **Safe integration of parameterized IP**. In: integration the VLSI journal, v. 37, n. 4, p 193, 2004.

(28) FOSTER, H. KROLNIK, A. LACEY, D. **Assertion based design**. Dordrecht: Kluwer. 2004. 390p.

(29) SYNOPSYS. **Verification methodology manual**. 2002

(30) SINGH, S. DRUCKER, L. KHAN, N. **Advanced verification techniques for SystemC**. Dordrecht: Kluwer, 2004. 396p.

(31) LAVAGNO, L. SCHEFFER, L. MARTIN G. **EDA for IC System Design, Verification, and Testing**. Dordrecht: CRC, 2006. 608p.

(32) DA SILVA, K. MELCHER, E. ARAUJO, G. **An automatic testbench generation tool for a SystemC functional verification methodology.** In: IEEE conference on Integrated circuits and systems design (SBCCI), 2004, pp 66 - 70.

(33) GRINWALD, R. HAREL, E. ORGAD, M. UR, S. ZIV, A **User defined coverage a tool supported methodology for design verification**. In: IEEE design automation conference (DAC), 1998. pp 158 - 163.

(34) LACHISH, O. **Hole analysis for functional coverage data.** In: IEEE design automation conference (DAC), 2002, pp 807- 812.

(35) ASAF. S. MARCUS, E. ZIV, A. **Defining coverage views to improve functional coverage analysis.** In: IEEE design automation conference (DAC), 2004, p 41 - 44.

(36) GUZEY, O. WANG, L. **Coverage-directed test generation through automatic constraint extraction.** In: IEEE high level design validation and test workshop (HDLVT), 2007, pp 151 - 158.

(37) BRAMER, M. **Principles of data mining.** Dordrecht: Springer, 2007. 344p.

(38) MITCHELL, T. **Machine Learining.** Dordrecht: McGraw Hill, 1997. 414p.

(39) QUINLAN, R. **C4.5: Programs for Machine Learning.** Dordrecht: Morgan Kaufmann, 1992. 302p.

(40) CASTILLO, E. GUTIÉRREZ, J. HADI, A. **Expert systems and probabilistic network models.** Dordrecht: Springer, 1997. 605p.

(41) OCHOA J. **Algoritmos EM para Aprendizagem de Redes Bayesianas a partir de Dados Incompletos.** 2004. 120 p. Dissertação (Mestrado) - Centro de ciências exatas e tecnologia, Universidade Federal de Mato Grosso do Sul, Campo Grande, 2004

(42) MCLACHLAN, G. THRIYAMBAKAM, K. **The EM Algorithm and Extensions.** Dordrecht: Wiley, 2008. 360p.

(43) PLATT, J. **Fast training of support vector machines using sequential minimal optimization.** In: advances in kernel methods - support vector learning, MIT Press, Cambridge, p. 185-208, 1999.

(44) GONZALEZ, J.A.Q. **Uma Metodologia de Projetos para Circuitos com Reconfiguração Dinâmica de Hardware Aplicada a Support Vector Machine**. 2006. 144p Tese (Doutorado), Escola politécnica, Universidade de São Paulo, 2006.

(45) KEERTHI, S.S. et al **Improvements to Platt's Algorithm for SVM Classifier Design**. Neural Computation, 13:637–649, 2001.

(46) FILHO GOMES, J. **Aplicação de técnicas de reconfiguração dinâmica a projeto de máquina de vetor suporte (SVM).** 2010. 93p. Dissertação (Mestrado). Escola politécnica, Universidade de São Paulo, 2010.

(47) BLUETOOTH special interest group. **Bluetooth protocol specification**. Version 1,1. 2001 www.bluetooth.org

(48) BRAZIL-IP **Projeto plataforma fênix**. Disponível em: http://www.brazilip.org.br/fenix/fenix_plataform.asp. Acesso em: maio 2009

(49) XILINX; **Virtex-II Platform FPGAs: Complete Data Sheet.** DS03,1, v. 3.5, November 5, 2007

(50) ISO-IEC. **MPEG-4 Document Standard**, ISO-IEC, 14496, 2000.

(51) RICHARDSON, I. **MPEG-4 and H.264 Video Compression: Video Coding for Next Generation Multimedia.** Dordrecht: Wiley, 2003. 320p.

(52) WAIKATO University. **WEKA 3 Data Mining Software in Java**. Disponível em: http://www.cs.waikato.ac.nz/ml/weka/ Acesso em: Janeiro 2010.

(53) DECISION systems laboratory. **Genie & smile wiki documentation**. Disponível em: http://genie.sis.pitt.edu/. Acesso em: Maio 2009.

(54) ACOSTA, R. **MP-SMO: um algoritmo para a implementação VLSI do treinamento de máquinas de vetores de suporte.** 2008. 112p. Dissertação (mestrado). Escola Politécnica, Universidade de São Paulo, São Paulo. 2008.

(55) CHAUDHURI, S. MOTWANI, R. NARASAYYA, V. **Random sampling for histogram construction: how much is enough?.** In: ACM international conference on management of data, 1998, pp 436 - 447.

(56) ROMERO, E. CASTRO, C. STRUM, M. WANG, J. **Domain Oriented Training for Data Mining Coverage Driven Verification.** In: Design and verification conference (DVCON), 2009, pp 64-71.

(57) ROMERO, E. ACOSTA, R. STRUM, M. WANG, J. **Support vector machine coverage driven verification for communication cores.** In: International conference on Very Large Scale Integration (VLSI-SoC), 2009, pp 33 - 37.

APÊNDICE 1- Tabelas de resultados do caso de estudo CBP

A1.1. Comparativo segundo o tamanho do conjunto

O comparativo entre os resultados do treinamento e da execução dos *testbenches* segundo o tamanho de treinamento visa mostrar o efeito de aumentar a quantidade de informação utilizada durante o aprendizado. As comparações são feitas a partir das Tabelas 1, 2, 3 e 4, com resultados com as técnicas de *data mining*, *data mining* sem corte, redes Bayesianas e SVM, respectivamente. As tabelas estão organizadas da seguinte forma: na primeira coluna são apresentados os modelos obtidos com a variedade de conjunto de treinamento de tamanho grande, nos diversos casos de balanceamento de categorias, que são comparados com os modelos obtidos com a variedade de conjunto de treinamento de tamanho médio, apresentados na segunda coluna; a terceira coluna apresenta a comparação qualitativa dos resultados do treinamento, no que se refere aos erros de FP e FN, do modelo da coluna 1 em relação ao da coluna 2; a quarta coluna apresenta a redução verificada no número de casos de teste simulados; e a quinta coluna apresenta a porcentagem de redução no tempo de execução.

Tabela 1 - Comparativo dos resultados do treinamento com *data mining* e do *testbench* entre os conjuntos de tamanho médio e grande, para o CBP

Condição de treinamento 1	**Condição de treinamento 2**	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. grande	C. médio	igual	0,00	-0,87
C. grande bal 70	C. médio bal 70	melhor	10,07	10,96
C. grande bal 50	C. médio bal 50	melhor	0,81	-0,33

Tabela 2 - Comparativo dos resultados do treinamento com *data mining* sem corte e do *testbench* entre os conjuntos de tamanho médio e grande, para o CBP

Condição de treinamento 1	**Condição de treinamento 2**	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. grande	C. médio	melhor	8,24	5,38
C. grande bal 70	C. médio bal 70	melhor	3,85	3,21
C. grande bal 50	C. médio bal 50	melhor	3,38	3,02

Tabela 3 - Comparativo dos resultados do treinamento com redes Bayesianas e do *testbench* entre os conjuntos de tamanho médio e grande, para o CBP

Condição de treinamento 1	Condição de treinamento 2	Diferença de erro	Diferença *Testbench* (%)	
			No. casos de teste	Tempo
C. grande	C. médio	igual	-20,85	-26,60
C. grande bal 70	C. médio bal 70	melhor	-4,98	-6,02
C. grande bal 50	C. médio bal 50	igual	-11,53	-14,22

Tabela 4 - Comparativo dos resultados do treinamento com SVM e do *testbench* entre os conjuntos de tamanho médio e grande, para o CBP

Condição de treinamento 1	Condição de treinamento 2	Diferença de erro	Diferença *Testbench* (%)	
			No. casos de teste	Tempo
C. grande	C. médio	melhor	7,54	-147,10
C. grande bal 70	C. médio bal 70	pior	-76,39	-463,07
C. grande bal 50	C. médio bal 50	pior	-92,50	-551,41

A1.2. Comparativo segundo o balanceamento

O segundo aspecto de comparação é o efeito de balanceamento dos casos de treinamento, tanto nos resultados dos modelos aprendidos quanto na redução dos números de casos de teste e tempo observados na execução do *testbench*. Este comparativo visa mostrar o efeito da redução da quantidade de casos de teste nas categorias determinadas majoritárias, em especial quando em número muito maior que com outras categorias minoritárias.

Para esta análise, as Tabelas de 5 a 9 apresentam, respectivamente, os comparativos com os modelos obtidos com as técnicas de *data mining*, *data mining* sem corte, redes Bayesianas e SVM. Estas tabelas estão organizadas da seguinte forma: A primeira coluna apresenta os modelos obtidos com uma variedade de conjunto de treinamento com uma certa condição de balanceamento de casos de teste, para os diversos tamanhos de conjunto, que são comparados com os modelos obtidos com outra variedade de conjunto de treinamento com balanceamento menor, apresentados na segunda coluna; a terceira coluna apresenta a comparação qualitativa dos resultados do treinamento, no que se refere aos erros de FP e FN, do modelo da coluna 1 em relação ao da coluna 2; a quarta coluna apresenta a redução verificada no número de casos de teste simulados; e a quinta coluna apresenta a porcentagem de redução no tempo de execução.

Tabela 5 - Comparativo dos resultados do treinamento com *data mining* e do *testbench* entre os conjuntos com diferentes balanceamentos, para o CBP

Condição de treinamento 1	**Condição de treinamento 2**	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. médio bal 70	C. médio	melhor	5,81	4,75
C. grande bal 70	C. grande	melhor	15,89	16,58
C. médio bal 50	C. médio bal 70	melhor	37,87	38,30
C. grande bal 50	C. grande bal 70	melhor	28,60	27,01

Tabela 6 - Comparativo dos resultados do treinamento com *data mining* sem corte e do *testbench* entre os conjuntos com diferentes balanceamentos, para o CBP

Condição de treinamento 1	**Condição de treinamento 2**	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. médio bal 70	C. médio	melhor	16,84	17,23
C. grande bal 70	C. grande	melhor	12,45	15,06
C. médio bal 50	C. médio bal 70	melhor	10,65	9,90
C. grande bal 50	C. grande bal 70	melhor	10,18	9,71

Tabela 7 - Comparativo dos resultados do treinamento com redes Bayesianas e do *testbench* entre os conjuntos com diferentes balanceamentos, para o CBP

Condição de treinamento 1	**Condição de treinamento 2**	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. médio bal 70	C. médio	melhor	16,98	19,35
C. grande bal 70	C. grande	melhor	32,85	39,92
C. médio bal 50	C. médio bal 70	igual	29,66	40,12
C. grande bal 50	C. grande bal 70	igual	23,11	31,93

Tabela 8 - Comparativo dos resultados do treinamento com SVM e do *testbench* entre os conjuntos com diferentes balanceamentos, para o CBP

Condição de treinamento 1	**Condição de treinamento 2**	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. médio bal 70	C. médio	melhor	-107,67	4038,15
C. grande bal 70	C. grande	melhor	-191,60	3722,18
C. médio bal 50	C. médio bal 70	melhor	18,15	102,96
C. grande bal 50	C. grande bal 70	melhor	2,04	14,61

A1.3. Comparativo segundo a técnica de aprendizado

A análise dos resultados do caso de estudo do CBP considerou um último aspecto importante que é a dependência dos resultados com respeito á técnica de aprendizado utilizada. Tendo a técnica de *data mining* como referência de comparação os modelos aprendidos e os resultados dos *testbenches* foram comparados da seguinte forma: as Tabelas 9, 10 e 11 apresentam os resultados comparativos, respectivamente, das técnicas de *data mining* sem corte redes Bayesianas e SVM. A técnica de *data mining* foi utilizada como referência porque apresentou uma alta estabilidade em todos os casos de estudo, nos resultados de treinamento e do *testbench*.

Estas tabelas estão organizadas da seguinte forma: a primeira coluna apresenta as condições de treinamento da técnica comparada, em termos de tamanho do conjunto de treinamento e balanceamento dos casos de teste; a segunda coluna apresenta as condições de treinamento de *data mining*; a terceira coluna apresenta resultado da análise qualitativa no que refere-se aos erros de FP e FN, do modelo da coluna 1 em relação ao da coluna 2; a quarta coluna apresenta a redução verificada no número de casos de teste simulados; e a quinta coluna apresenta a porcentagem de redução no tempo de execução.

Tabela 9 - Comparativo dos resultados do treinamento e do *testbench* para o CBP entre as técnicas de *data mining* sem corte e *data mining*

***Data mining* sem corte**	***Data mining***	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. médio	C. médio	melhor	13.62	12.57
C. grande	C. grande	melhor	21.86	18.82
C. médio bal 70	C. médio bal 70	melhor	24.64	25.06
C. grande bal 70	C. grande bal 70	melhor	18.42	17.30
C. médio bal 50	C. médio bal 50	melhor	-2.57	-3.34
C. grande bal 50	C. grande bal 50	melhor	0.00	0.00

Tabela 10 - Comparativo dos resultados do treinamento e do *testbench* para o CBP entre as técnicas de redes Bayesianas e *data mining*

Redes Bayesianas	***Data mining***	**Diferença erro**	**Diferença *Testbench***	
			Casos de teste	**Tempo**
C. médio	C. médio	igual	-147,17	-210,21
C. grande	C. grande	igual	-168,02	-235,93
C. médio bal 70	C. médio bal 70	pior	-136,00	-195,61
C. grande bal 70	C. grande bal 70	pior	-151,06	-212,59
C. médio bal 50	C. médio bal 50	pior	-144,21	-193,79
C. grande bal 50	C. grande bal 50	pior	-149,87	-169,70

Tabela 11 - Comparativo dos resultados do treinamento e do *testbench* para o CBP entre as técnicas de SVM e *data mining*

SVM	***Data mining***	**Diferença erro**	**Diferença *Testbench***	
			Casos de teste	**Tempo**
C. médio	C. médio	melhor	-24,10	-4501,38
C. grande	C. grande	melhor	-16,56	-4647,61
C. médio bal 70	C. médio bal 70	melhor	-137,58	-467,98
C. grande bal 70	C. grande bal 70	melhor	-224,05	-942,01
C. médio bal 50	C. médio bal 50	melhor	-157,31	-403,32
C. grande bal 50	C. grande bal 50	melhor	-250,62	-954,41

APÊNDICE 2- Tabelas de resultados do caso de estudo PIACDC

A2.1. Comparativo segundo o tamanho do conjunto

Para a análise da influência do tamanho do conjunto de treinamento, são apresentadas, nesta seção, as seguintes tabelas: as Tabelas 1, 2 e 3 apresentam uma comparação entre os resultados das variedades de conjunto de treinamento grande e médio para a técnica de *data mining*, *data mining* sem corte e SVM. Por sua vez a informação nestas tabelas estão organizadas da seguinte forma: a primeira coluna indica a variedade de conjunto de treinamento grande; a segunda coluna indica a variedade de conjunto de treinamento médio; a terceira coluna apresenta o resultado da avaliação qualitativa das condições indicadas nas colunas 1 em relação aos da coluna 2, no que se refere aos erros de FP e FN; a quarta coluna apresenta a redução verificada no número de casos de teste simulados entre as duas condições; e a quinta coluna apresenta a porcentagem de redução no tempo de execução.

Tabela 1 - Comparativo dos resultados do treinamento com *data mining* e do *testbench* entre os conjuntos de tamanho médio e grande, para o PIACDC

Condição de treinamento 1	**Condição de treinamento 2**	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. grande	C. médio	melhor	-0,07	-27,82
C. grande bal 70	C. médio bal 70	melhor	-0,07	-27,26
C. grande bal 50	C. médio bal 50	melhor	-0,07	1,09

Tabela 2 - Comparativo dos resultados do treinamento com *data mining* sem corte e do *testbench* entre os conjuntos de tamanho médio e grande, para o PIACDC

Condição de treinamento 1	**Condição de treinamento 2**	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. grande	C. médio	melhor	0,00	-43,43
C. grande bal 70	C. médio bal 70	melhor	-0,01	-42,68
C. grande bal 50	C. médio bal 50	melhor	-0,07	-0,78

Tabela 3 - Comparativo dos resultados do treinamento com SVM e do *testbench* entre os conjuntos de tamanho médio e grande, para o PIACDC

Condição 1	**Condição 2**	**Diferença erro**	**Diferença *Testbench***	
			Casos de teste	**Tempo**
C. grande	C. médio	melhor	-1,99	-35,14
C. grande bal 70	C. médio bal 70	melhor	0,19	0,61
C. grande bal 50	C. médio bal 50	melhor	0,00	0,65

A2.1. Comparativo segundo o balanceamento

A presente seção apresenta a análise feita a partir do cruzamento dos resultados do treinamento e os resultados dos *testbenches* com as técnicas de *data mining*, *data mining* sem corte e SVM visando verificar a influência do balanceamento dos casos de teste. Interessa verificar o efeito da redução da quantidade de casos de teste em determinadas categorias (majoritárias), quando em número muito maior que com outras categorias minoritárias. O comparativo é apresentado da seguinte forma: as Tabelas 4, 5 e 6, mostram resultados, respectivamente, para as técnicas de *data mining*, *data mining* sem corte e SVM. As três tabelas estão organizadas da seguinte forma: a primeira coluna apresenta os modelos obtidos com uma variedade de conjunto de treinamento com uma certa condição de balanceamento de casos de teste, para os diversos tamanhos de conjunto, que são comparados com os modelos obtidos com outra variedade de conjunto de treinamento com balanceamento menor, apresentados na segunda coluna; a terceira coluna apresenta a comparação qualitativa entre os resultados do treinamento da coluna 1 com respeito às condições da coluna 2, no que se refere aos erros de FP e FN; a quarta coluna apresenta a redução verificada no número de casos de teste simulados nos *testbench*es; e , a quinta coluna apresenta a porcentagem de redução no tempo de execução.

Tabela 4 - Comparativo dos resultados do treinamento com *data mining* e do *testbench* entre os conjuntos com diferentes balanceamentos, para o PIACDC

Condição de treinamento 1	**Condição de treinamento 2**	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. médio bal 70	C. médio	melhor	0,00	-0,10
C. grande bal 70	C. grande	igual	0,00	0,46
C. médio bal 50	C. médio bal 70	melhor	0,00	-1,09
C. grande bal 50	C. grande bal 70	igual	0,00	27,26

Tabela 5 - Comparativo dos resultados do treinamento com *data mining* sem corte e do *testbench* entre os conjuntos com diferentes balanceamentos, para o PIACDC

Condição de treinamento 1	**Condição de treinamento 2**	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. médio bal 70	C. médio	melhor	0,00	-1,13
C. grande bal 70	C. grande	igual	0,00	-0,38
C. médio bal 50	C. médio bal 70	melhor	0,00	0,78
C. grande bal 50	C. grande bal 70	igual	-0,06	42,68

Tabela 6 - Comparativo dos resultados do treinamento com SVM e do *testbench* entre os conjuntos com diferentes balanceamentos, para o PIACDC

Condição de treinamento 1	**Condição de treinamento 2**	**Diferença de erro**	**Diferença *Testbench* (%)**	
			No. casos de teste	**Tempo**
C. médio bal 70	C. médio	melhor	-1,05	119,22
C. grande bal 70	C. grande	melhor	1,13	154,97
C. médio bal 50	C. médio bal 70	melhor	0,00	-11,58
C. grande bal 50	C. grande bal 70	melhor	-0,19	-11,54

A2.3. Comparativo segundo a técnica de aprendizado

Esta seção apresenta os resultados comparativos de treinamento e execução dos *testbenches* do módulo PIACDC segundo a técnica de aprendizado utilizada. Enquanto a Tabela mostra as diferenças quando a técnica de *data mining* sem corte é utilizada ao invés da técnica de *data mining*, a Tabela mostra as diferenças quando se compara a utilização da técnica SVM ao invés da técnica de *data mining*. Neste comparativo a *data mining* foi utilizada como referência porque ele tem mostrado melhor estabilidade nos resultados dos *testbenches*. As duas tabelas estão organizadas da seguinte forma: a primeira coluna apresenta as condições de treinamento usadas para a técnica comparada seja *data* mining sem corte ou SVM, em termos de tamanho do conjunto de treinamento e balanceamento dos casos de teste; a segunda coluna apresenta as condições usadas para a técnica de *data mining*; a terceira coluna apresenta o resultado a comparação qualitativa entres os modelos aprendidos, no que refere-se aos erros de FP e FN, do modelo da coluna 1 em relação ao da coluna 2; na quarta coluna, é apresentada a redução verificada no número de casos de teste simulados; e, a quinta coluna apresenta a porcentagem de redução no tempo de execução.

Tabela 7 - Comparativo dos resultados do treinamento e do *testbench* para o CBP entre as técnicas de *data mining* sem corte e *data mining*

***Data mining* sem corte**	***Data mining***	**Diferença erro**	**Diferença *Testbench***	
			Casos de teste	**Tempo**
C. médio	C. médio	melhor	0,00	-7,76
C. grande	C. grande	melhor	0,06	-23,37
C. médio bal 70	C. médio bal 70	melhor	0,00	-8,79
C. grande bal 70	C. grande bal 70	melhor	0,06	-24,21
C. médio bal 50	C. médio bal 50	melhor	0,00	-6,92
C. grande bal 50	C. grande bal 50	melhor	0,00	-8,79

Tabela 8 - Comparativo dos resultados do treinamento e do *testbench* para o CBP entre as técnicas de SVM e *data mining*

SVM	***Data mining***	**Diferença erro**	**Diferença *Testbench***	
			Casos de teste	**Tempo**
C. médio	C. médio	pior	0,01	-73,42
C. grande	C. grande	pior	-1,92	-80,74
C. médio bal 70	C. médio bal 70	melhor	-1,05	45,91
C. grande bal 70	C. grande bal 70	melhor	-0,78	73,77
C. médio bal 50	C. médio bal 50	melhor	-1,05	35,42
C. grande bal 50	C. grande bal 50	melhor	-0,98	34,98