Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Alocação e Decomposição de Safety Integrity Levels em Sistemas Automotivos Utilizando Meta-heurísticas

Anderson Costa de Souza

JUIZ DE FORA
SETEMBRO, 2024

Alocação e Decomposição de Safety Integrity Levels em Sistemas Automotivos Utilizando Meta-heurísticas

Anderson Costa de Souza

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: André Luiz de Oliveira

JUIZ DE FORA

SETEMBRO, 2024

Alocação e Decomposição de Safety Integrity Levels em Sistemas Automotivos Utilizando Meta-heurísticas

Anderson Costa de Souza

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

André Luiz de Oliveira
Doutor em Ciências da Computação e Matemática Computacional

Gleiph Ghiotto Lima de Menezes
Doutor em Ciências da Computação

Ciro de Barros Barbosa
Doutor em Ciências da Computação

JUIZ DE FORA
17 DE SETEMBRO, 2024

Aos meus pais e irmã.

Resumo

O conceito de *Automotive Safety Integrity Level* (ASIL) é utilizado pela norma ISO 26262 para categorizar o nível de rigor dos requisitos de segurança em sistemas automotivos. Durante o desenvolvimento desses sistemas, os ASILs são iterativamente alocados com o objetivo de mitigar os efeitos de falhas em sistemas, subsistemas e componentes. Quanto maior o rigor do ASIL atribuído a um subsistema ou componente, maior será o custo associado às medidas necessárias para atender aos requisitos de segurança desse ASIL. A ISO 26262 define um método de decomposição de ASILs que permite a elementos redundantes compartilharem a responsabilidade de cumprir um dado ASIL. Dessa forma, encontrar soluções eficientes de alocação de ASILs que otimizem os custos de desenvolvimento e garantam a segurança do sistema tornou-se uma etapa crucial no projeto de sistemas automotivos.

Na literatura, há um conjunto de métodos e ferramentas baseados em meta-heurísticas, como Algoritmos Genéticos, Busca Tabu, Bando de Pinguins, entre outros, empregados para resolver o problema de alocação de ASILs no desenvolvimento de sistemas de segurança crítica. Este estudo propõe um algoritmo baseado em Busca Tabu para resolver o problema de alocação e decomposição de ASILs pelos componentes da arquitetura de sistemas automotivos. A solução proposta foi avaliada em um sistema automotivo de médio porte, e foi realizada uma análise comparativa entre o tempo de execução e os resultados obtidos pelo algoritmo proposto em relação à ferramenta comercial HiP-HOPS que fornece uma solução para este problema. Os resultados demonstraram que o algoritmo proposto produziu a mesma solução de alocação de ASILs que a ferramenta HiP-HOPS. No entanto, em termos de tempo de execução, a ferramenta HiP-HOPS apresentou desempenho superior ao do algoritmo proposto.

Palavras-chave: Automotive Safety Integrity Level (ASIL), Segurança, Sistemas de Segurança Crítica, Sistemas Automotivos, Meta-heurística, Busca Tabu

Abstract

The concept of Automotive Safety Integrity Level (ASIL) is used by the ISO 26262 standard to categorize the level of rigor of safety requirements in automotive systems. During the development of these systems, ASILs are iteratively allocated with the goal of mitigating the effects of failures in systems, subsystems, and components. The greater the rigor of the ASIL assigned to a subsystem or component, the higher the cost associated with the measures necessary to meet the safety requirements of that ASIL. ISO 26262 defines a method of ASIL decomposition that allows redundant elements to share the responsibility of fulfilling a given ASIL. Thus, finding efficient ASIL allocation solutions that optimize development costs and ensure system safety has become a crucial step in the design of automotive systems.

In the literature, there is a set of methods and tools based on metaheuristics, such as Genetic Algorithms, Tabu Search, Penguin Colony, among others, employed to solve the ASIL allocation problem in the development of critical safety systems. This study proposes a Tabu Search-based algorithm to solve the problem of allocating and decomposing ASILs across the components of automotive system architectures. The proposed solution was evaluated on a medium-sized automotive system, and a comparative analysis was conducted between the execution time and the results obtained by the proposed algorithm and the commercial tool HiP-HOPS, which provides a solution for this problem. The results showed that the proposed algorithm produced the same ASIL allocation solution as the HiP-HOPS tool. However, in terms of execution time, the HiP-HOPS tool outperformed the proposed algorithm.

Keywords: Automotive Safety Integrity Level (ASIL), Safety, Critical Safety Systems, Automotive Systems, Metaheuristic, Tabu Search

Agradecimentos

Dedico o presente trabalho, primeiramente, a Deus, fundamento de toda a realidade.

Aos meus pais, Ana Maria da Costa Souza e Marco António de Souza, que não mediram esforços para que essa etapa tão importante da minha vida se concretizasse, concedendo-me apoio emocional e financeiro.

A minha irmã Mariana de Souza Costa por toda preocupação e carinho, cujo sorriso é fonte de muita alegria.

Aos meus professores do Departamento de Computação, que ao longo de todos esses anos me ensinaram atenciosamente tanto sobre minha especialização quanto sobre a vida.

Ao meu orientador Prof. Dr. André Luiz de Oliveira pela amizade, paciência e orientação durante todo esse árduo processo.

« *Ce n'est pas qu'il faut arriver à quelque chose, c'est qu'il faut sortir de là où l'on est.* »

— *Marguerite Duras, L'Amant*

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

ASIL	Automotive Safety Integrity Level
DCC	Departamento de Ciência da Computução
FMEA	Failure Modes & Effects Analysis
FTA	Fault Tree Analysis
HiP-HOPS	Hierarchically Performed Hazard Origin & Propagation Studies
UFJF	Universidade Federal de Juiz de Fora

1 Introdução

O avanço tecnológico contribuiu para o aumento da complexidade de *software* e *hardware*, com a interação cada vez mais profunda entre sistemas computacionais, processos físicos, pessoas e a sociedade. Essa complexidade e interação tornam-se ainda mais críticas ao projetarmos sistemas de segurança. Sistemas críticos de segurança são aqueles cuja falha pode resultar em lesões ou perda de vidas humanas, danos significativos à propriedade (financeira ou material), ou impactos ambientais. Exemplos de sistemas críticos incluem sistemas automotivos, como assistentes de direção autônoma, propulsão, controle aerodinâmico do veículo, além de sistemas de segurança ativa e passiva. A natureza crítica desses sistemas exige a garantia de propriedades de segurança. Segurança, nesse contexto, refere-se à garantia de que o sistema operará sem causar danos catastróficos, mesmo em caso de falhas. Por esse motivo, o desenvolvimento de sistemas críticos de segurança requer aderência a diretrizes estabelecidas por normas como a IEC 61508, para sistemas de automação industrial, e a ISO 26262, que assegura a segurança funcional de sistemas eletrônicos automotivos.

1.1 Contextualização

A ISO 26262 é uma adaptação da norma IEC 61508 para o domínio automotivo, que descreve todas as atividades do ciclo de vida da produção de veículos de passeio compostos por componentes elétricos, eletrônicos e de software. A ISO 26262 prescreve requisitos e processos de garantia de qualidade que devem ser seguidos para atingir o nível de confiança necessário. O standard de segurança para controle de processos industriais IEC 61508 define o conceito de *Safety Integrity Levels* (SIL), que consiste em requisitos de desenvolvimento e verificação de diferentes níveis de rigor alocados para mitigar os efeitos de falhas em sistemas ou componentes, conforme seus níveis de risco, em termos de severidade e probabilidade de ocorrência. Adaptando os SILs, a ISO 26262 oferece uma abordagem baseada em risco específica para o domínio automotivo, determinando as

classes de risco conhecidas como *Automotive Safety Integrity Levels* (ASIL).

A ISO 26262 prescreve um conjunto de atividades de garantia de qualidade, como verificação, validação e testes, que devem ser realizadas para atingir cada nível de confiança. Os ASILs são categorizados em níveis de A a D, do menos ao mais grave, e um quinto nível, denominado QM (*Quality Management*), é designado para riscos nos quais não são necessárias medidas especiais. Cada nível possui um valor numérico associado pela álgebra definida pela ISO 26262: ASIL(QM) = 0, ASIL(A) = 1, ASIL(B) = 2, ASIL(C) = 3, e ASIL(D) = 4. A alocação do ASIL é baseada no comportamento funcional do item avaliado (componente ou subsistema), considerando a gravidade, a probabilidade de exposição e a controlabilidade do impacto causado pela falha do item.

1.2 Motivação e Caracterização do Problema

Quanto maior o ASIL alocado a um objetivo de segurança, mais custosos, tanto financeiramente quanto em termos de esforço, tornam-se os requisitos de segurança designados. Através da decomposição de ASILs, a ISO 26262 permite que um ASIL alocado seja decomposto em componentes que compartilham a responsabilidade por uma falha. Assim, por exemplo, dois componentes responsáveis por um ASIL(D) podem ser decompostos de três formas diferentes: ASIL(QM) e ASIL(D): 0 + 4 = 4, ASIL(C) e ASIL(A): 3 + 1 = 4, e ASIL(B) e ASIL(B): 2 + 2 = 4. Uma heurística de custo é associada a cada ASIL, podendo ser uma heurística linear, exponencial ou experimental. Dessa forma, as diferentes decomposições podem ser comparadas.

O processo de alocação de ASILs para mitigar os efeitos de perigos e sua decomposição em subsistemas e componentes, quando realizado manualmente, demanda tempo e é propenso a erros. A ferramenta HiP-HOPS fornece, de forma automatizada, duas abordagens para alocação de ASILs: uma análise dedutiva de cima para baixo, conhecida como Análise de Árvore de Falhas (FTA, do inglês *Fault Tree Analysis*), e uma análise indutiva de baixo para cima, a Análise de Modos de Falhas e Efeitos (FMEA, do inglês *Failure Modes I& Effects Analysis*). Com a FTA, é possível identificar os conjuntos de corte (*cut sets*), ou seja, as mínimas combinações de eventos que causam uma falha. Cada evento contido em um conjunto de corte deve atender ao ASIL do conjunto como um todo,

ou, mais especificamente, a decomposição dos ASILs dos eventos deve satisfazer o ASIL do conjunto. Com os conjuntos de corte e a heurística de custo, o problema da decomposição de ASILs pode ser tratado computacionalmente.

Existem várias técnicas na literatura que apoiam a decomposição de ASILs alocados para mitigar os efeitos de perigos nos subsistemas e componentes da arquitetura. Esses métodos podem ser classificados em técnicas exatas e meta-heurísticas. Os solvers exatos garantem encontrar a solução ótima, porém com a desvantagem de demandarem um tempo significativo de processamento. Por outro lado, as meta-heurísticas encontram uma solução satisfatória em um tempo aceitável, embora não garantam que a solução encontrada seja a ótima para o problema. Um exemplo de solver exato é o Choco CSP, uma ferramenta de última geração para resolver o Problema da Satisfação de Restrições. Em contrapartida, algoritmos meta-heurísticos, como os da família da Busca Tabu, são frequentemente utilizados para resolver o problema de forma aproximada.

1.3 Justificativa

A alocação e decomposição de ASILs é essencial para o cumprimento da ISO 26262 e para atender aos requisitos de segurança no desenvolvimento de sistemas automotivos, além de reduzir seus custos. No entanto, o problema de alocação de ASILs possui uma natureza combinatória de grande complexidade, o que o torna desafiador. Diversas técnicas, classificadas como exatas ou baseadas em meta-heurísticas, foram desenvolvidas para lidar com esse problema. As técnicas exatas garantem a obtenção da solução ótima, mas podem exigir um tempo de processamento excepcionalmente longo. Em sistemas de menor porte, é vantajoso utilizar uma técnica exata, porém, em sistemas grandes e complexos, novas abordagens se tornam necessárias. As meta-heurísticas, por sua vez, encontram soluções em tempos aceitáveis, mesmo em sistemas de grande porte, mas essas soluções são aproximadas, sem garantia de serem ótimas. Ambas as abordagens possuem vantagens e desvantagens, tornando imprescindível analisar os sistemas em questão para decidir a melhor técnica a ser empregada.

Dada a relevância do problema de alocação e decomposição de ASILs para a Engenharia de Sistemas Embarcados Críticos, bem como os desafios inerentes à obtenção

de uma solução ótima em tempo razoável ao utilizar métodos exatos e meta-heurísticas disponíveis na literatura, este trabalho propõe uma metodologia baseada no método da Busca Tabu para resolver esse problema. O estudo tem como objetivo avaliar a viabilidade dessa abordagem no contexto da alocação e decomposição de ASILs, investigando se as soluções podem ser encontradas em um tempo de processamento inferior ao das abordagens implementadas pela ferramenta comercial HiP-HOPS, ou, na ausência de ganhos de tempo, se a metodologia proposta oferece soluções de custo equivalente ou inferior.

1.4 Hipótese

Este estudo investiga se uma nova implementação do algoritmo de alocação e decomposição de ASILs, utilizando a meta-heurística da família da Busca Tabu, apresenta potencial para gerar soluções mais próximas da ótima, com menor custo e tempo de processamento, em comparação com a solução desenvolvida pela ferramenta HiP-HOPS. A decomposição de ASILs alocados ao nível de sistema visa mitigar os efeitos de eventos perigosos (hazards), os quais possuem potencial de causar danos catastróficos a pessoas, ao meio ambiente ou à propriedade. Esse processo envolve a alocação de requisitos de segurança a subsistemas e componentes que contribuem para a ocorrência dessas falhas, configurando um problema de otimização de natureza min-max: minimizar o custo de alocação dos ASILs enquanto se garante o nível mínimo de segurança necessário.

O problema em questão é uma otimização multinível, onde múltiplos atributos, como custo, segurança, confiabilidade e disponibilidade, podem ser considerados simultaneamente. O objetivo principal da nova implementação baseada em Busca Tabu é verificar se é possível reduzir tanto o tempo de processamento quanto os custos de alocação, ao mesmo tempo em que se mantém a garantia dos requisitos mínimos de segurança.

1.5 Objetivos

Este trabalho tem como objetivo propor uma nova abordagem e apoio ferramental baseada na meta-heurística da Busca Tabu para a resolução do problema de alocação e decomposição de ASILs. A efetividade e a eficácia do método proposto foi avaliada por

meio de um estudo empírico, aplicando à alocação e decomposição de ASILs em um sistema automotivo realista, e os resultados obtidos foram comparados com aqueles gerados pela ferramenta HiP-HOPS. Para alcançar o objetivo principal, os seguintes passos metodológicos foram seguidos:

- Apresentação e descrição do problema de alocação e decomposição de ASILs, explicando sua definição e sua relação com a norma ISO 26262, além da análise das soluções para este problema disponíveis na literatura.
- Projeto da solução utilizando a meta-heurística da Busca Tabu, detalhando seus principais mecanismos e a implementação do algoritmo proposto neste trabalho.
- Utilização da ferramenta HiP-HOPS para a geração automática dos relatórios de análise de árvores de falhas FTA e FMEA no estudo de caso. O arquivo XML gerado pela ferramenta foi utilizado na solução proposta, com suas informações armazenadas em uma estrutura de dados adequada para servir como entrada ao algoritmo desenvolvido.
- Realização de testes e análise comparativa entre os resultados obtidos pela solução proposta e aqueles gerados pela HiP-HOPS, com o intuito de verificar a validade da hipótese levantada neste estudo.

1.6 Organização do Trabalho

Esta monografia está organizada em cinco capítulos. No Capítulo 2, é apresentada a fundamentação teórica, abordando os conceitos necessários para a compreensão das contribuições deste estudo. No Capítulo 3 é descrita a metodologia utilizada para a busca de trabalhos relacionados, seguida de uma revisão e comparação dos resultados encontrados. No Capítulo 4 são detalhadas as principais técnicas e mecanismos aplicados na implementação do algoritmo proposto, baseado na meta-heurística Busca Tabu, incluindo uma descrição do pseudocódigo e do fluxograma da solução proposta. Por fim, no Capítulo 5 são apresentadas as conclusões, as contribuições e as limitações do trabalho, bem como propostas para estudos futuros.

2 Fundamentação Teórica

Neste capítulo, são apresentados os conceitos fundamentais sobre sistemas críticos de segurança e sistemas automotivos. Serão discutidas as características dos sistemas críticos de segurança e a razão pela qual os sistemas automotivos são classificados como tal, além da importância dos padrões de segurança, com foco nos IEC 61508 e ISO 26262. A análise de falhas, incluindo FTA e FMEA, será abordada junto com a ferramenta HiP-HOPS, que automatiza essas análises. Finalmente, será apresentado o problema de alocação e decomposição de ASILs e a meta-heurística da Busca Tabu como uma solução

2.1 Sistemas Críticos de Segurança

Segundo Knight (2002), a grande preocupação de um sistema crítico de segurança é com as consequências de uma falha. Se a falha de um sistema leva a cenários inaceitáveis, então ele é um sistema crítico de segurança. Em suma, sistemas críticos de segurança são sistemas cuja falha pode resultar em lesões ou perda de vida humana, danos significativos à propriedade (patrimônio ou ativo financeiro), ou danos ambientais.

2.2 Sistemas Automotivos

Segundo Mössinger (2010), desde a introdução de sistemas eletrônicos e de *software* em veículos, sua maior fonte de inovação, a indústria se reconfigurou. O *software* em sistemas automotivos aumentou consideravelmente o seu desempenho ao permitir mecanismos mais ágeis e sua segurança ao introduzir sistemas passivos e ativos de segurança. Atualmente veículos possuem várias unidades de controle eletrônico que formam uma rede complexa e interconectada dentro deles. Tais unidades de controle eletrônico são cada vez mais responsáveis por funcionalidade essenciais do veículo como, por exemplo, assistente de direção autônoma, propulsão, controle da aerodinâmica do veículo e o sistemas de segurança ativos e passivos. Pela importância do funcionamento correto dessas funciona-

lidades e a inaceitabilidade de suas falhas, sistemas automotivos são sistemas críticos de segurança.

2.3 Terminologia

Nesta subseção, é apresentado um pequeno glossário com os conceitos fundamentais para a segurança de *software* definidos por (AVIZIENIS et al., 2004). Tais conceitos, são importantes para o entendimento da linguagem utilizada no domínio de segurança de *software* onde o problema de alocação e decomposição de ASILs se encontra.

- Confiança (*Dependability*): capacidade de entregar um serviço que pode ser justificadamente confiável. De outra forma, a confiabilidade de um sistema é a capacidade de evitar falhas de serviço que são mais frequentes e mais graves do que o aceitável;
- Disponibilidade (*Availability*): prontidão para o serviço correto;
- Confiabilidade (*Reliability*): continuidade do serviço correto;
- Segurança (*Safety*): ausência de consequências catastróficas para o(s) usuário(s) e o meio ambiente. Integridade (*Integrity*): ausência de alterações inadequadas do sistema;
- Manutenibilidade (*Maintainability*): capacidade de submeter-se a modificações e reparos;
- Confidencialidade (*Confidentiality*): ausência de divulgação não autorizada de informações
- Proteção (*Security*): uma composição dos atributos de confidencialidade, integridade e disponibilidade, exigindo a existência simultânea de: 1) disponibilidade apenas para ações autorizadas, 2) confidencialidade e 3) integridade com "inadequado"significando "não autorizado".
- Falha (*Failure*): um evento que ocorre quando o serviço entregue se desvia do serviço correto. Um serviço falha porque não está conforme a especificação funcional ou porque esta especificação não descreve adequadamente a função do sistema;

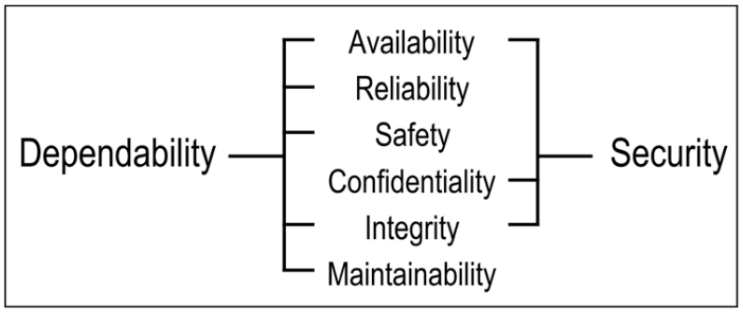

Figura 2.1: Atributos de Confiabilidade e Proteção.

- Defeito (*Fault*): característica do sistema que pode levar a um erro de sistema;
- Erro (Error): parte do estado total do sistema que pode levar à sua subsequente falha de serviço;
- Prevenção de defeitos (*Fault prevention*): significa prevenir a ocorrência ou introdução de defeitos;
- Tolerância a defeitos (*Fault tolerance*): significa evitar falhas de serviço na presença de defeitos;
- Remoção de defeitos (*Fault removal*): significa reduzir o número e a gravidade dos defeitos;
- Previsão de defeitos (*Fault forecasting*): significa estimar o número presente, a incidência futura e as prováveis consequências dos defeitos.

2.3.1 Conceitos da ISO 26262

Nesta subseção, são descritas as principais terminologias e conceitos definidos pela documentação da ISO 26262. Os mesmos são fundamentais para o entendimento da linguagem utilizada na definição do ciclo de vida proposto pela ISO 26262 e suas diretrizes.

- Perigo (*Hazard*): fonte potencial de dano causado pelo mau funcionamento de um item;
- Item (Item): sistema ou conjunto de sistemas para implementar uma função no nível do veículo, ao qual a ISO 26262 é aplicada;

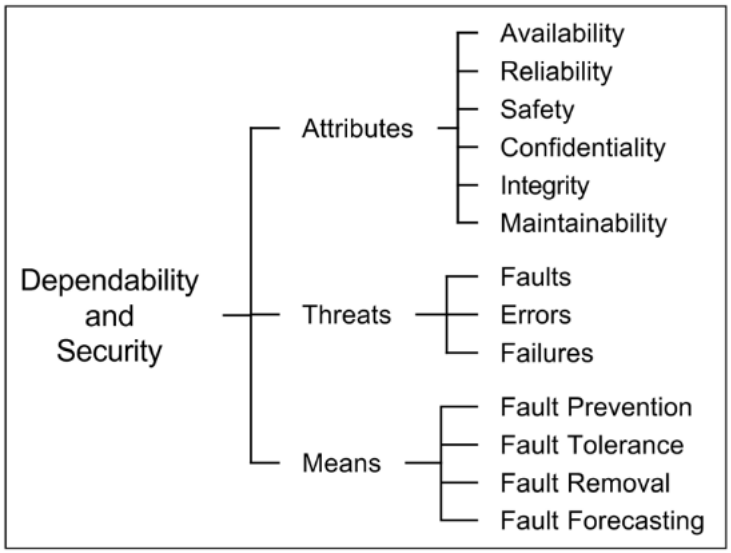

Figura 2.2: Árvore de Confiabilidade e Proteção.

- Elemento (*Element*): sistema ou parte de um sistema, incluindo componentes, *hardware*, *software*, peças de *hardware* e unidades de *software*;

- Risco (*Risk*): combinação da probabilidade de ocorrência do dano e da gravidade desse dano;

- Requisito de segurança funcional (*Functional safety requirement*): especificação do comportamento de segurança independente da implementação, ou medida de segurança independente da implementação, incluindo seus atributos relacionados à segurança;

- Nível de integridade de segurança automotiva (ASIL, do inglês *Automotive Safety Integrity Level*): um dos quatro níveis para especificar os requisitos necessários do item ou elemento da ISO 26262 e medidas de segurança a serem aplicadas para evitar um risco residual não razoável, com D representando o nível mais rigoroso e A o menos rigoroso.

- Decomposição de ASIL (*ASIL Decomposition*): repartição de requisitos de segurança redundantes para os elementos suficientemente independentes, com o objetivo de reduzir o ASIL dos requisitos de segurança redundantes atribuídos aos elementos correspondentes.

2.4 Standards de Segurança

2.4.1 IEC 61508

A Comissão Eletrotécnica Internacional (IEC, do inglês *International Electrotechnical Commission*) é uma organização internacional para a padronização de diretrizes nos campos elétricos e eletrônicos e publicação de *standards* internacionais. Por Gheraibia et al. (2018), o IEC 61508 é um *standard* internacional para a segurança funcional de sistemas elétricos, eletrônicos e eletrônicos programáveis (E/E/PE) relacionados à segurança. Ele estabelece os requisitos para garantir que esses sistemas forneçam uma segurança funcional satisfatória em todo o seu ciclo de desenvolvimento.

2.4.2 ISO 26262

A ISO 26262, (ISO, 2011), é uma adaptação do IEC 61508 para o domínio automotivo que descreve todas as atividades do ciclo de vida da produção de um veículo de passeio compostas por componentes elétricos, eletrônicos e de *software*, fornecendo requisitos e processos apropriados para garantir que os níveis de seguranças desejados sejam atendidos. Como definido pela ISO (2011), o *standard* possui dez partes:

1. Vocabulário (*Vocabulary*): onde são definidos os termos usados em todo o *standard*;
2. Gestão da segurança funcional (*Management of functional safety*): especifica requisitos de gerenciamento de segurança independente de projeto para organizações, incluindo requisitos para implementação de cultura de segurança e gerenciamento de competências. Também define os requisitos específicos do projeto para o gerenciamento das atividades de segurança a serem realizadas durante cada estágio do ciclo de vida da segurança;
3. Fase de conceito (*Concept phase*): especifica os requisitos para definição de item, início do ciclo de vida de segurança, análise de perigo e avaliação de risco e o conceito de segurança funcional;
4. Desenvolvimento de produto ao nível de sistema (*Product development at the system level*): especifica requisitos para desenvolvimento no nível do sistema, incluindo

especificação de requisitos de segurança, projeto, verificação, integração, teste e liberação para produção;

5. Desenvolvimento de produto ao nível de *hardware* (*Product development at the hardware level*): especifica requisitos para desenvolvimento no nível de *hardware*, incluindo especificação de requisitos de segurança, projeto, verificação, integração e teste;

6. Desenvolvimento de produto ao nível de *software* (*Product development at the software level*): Especifica requisitos para desenvolvimento no nível de *software*, incluindo especificação e verificação de requisitos de segurança, testes e integração;

7. Produção e operação (*Production and operation*): especifica os requisitos para produção, operação, serviço e descomissionamento do item;

8. Processos de suporte (*Supporting processes*): especifica requisitos para processos de suporte, como interfaces com desenvolvimentos distribuídos, gerenciamento de mudanças e documentação;

9. Análise orientada a ASIL e análise orientada a segurança (*ASIL-oriented and safety-oriented analysis*): especifica os requisitos para a aplicação da decomposição de ASILs e para a execução da análise de segurança;

10. Diretriz sobre a ISO 26262 (*Guideline on ISO 26262*): fornece uma visão geral do padrão e exemplos de processos-chave para melhor compreensão.

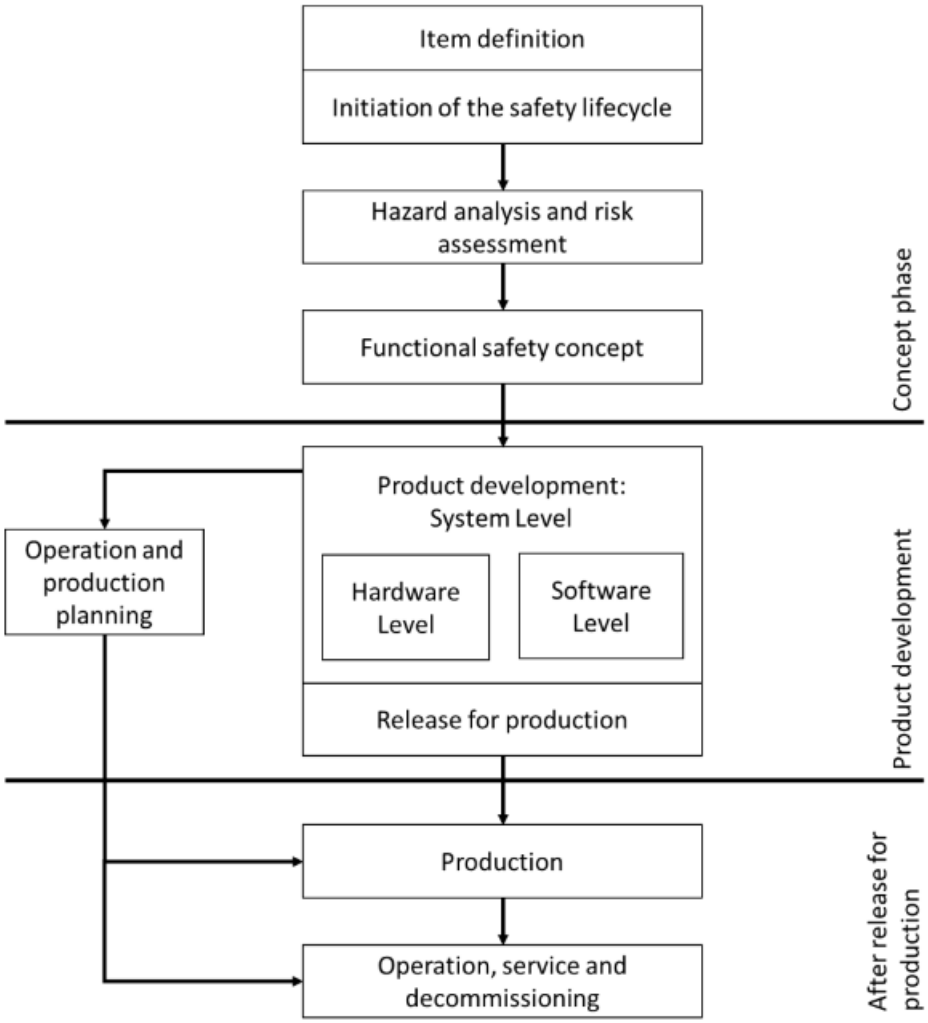

Figura 2.3: Ciclo de Vida de Segurança Simplificado da ISO 26262.

2.5 Técnicas de Análise de Falhas

2.5.1 Análise de Árvore de Falhas

A Análise de Árvore de Falhas (FTA, do *inglês Fault Tree Analysis*) como definida por Papadopoulos e Hull (2013), é uma técnica dedutiva feita assumindo que uma falha de sistema ocorre e, de trás para frente, buscam-se as possíveis combinações de eventos que a causou. A falha de sistema se torna a raiz da árvore, ou evento principal (*top event*), e as falhas dos componentes os nós folhas, ou eventos básicos (*basic events*), que se combinam pelas portas lógicas *AND* e *OR*. A FTA pode ser analisada qualitativamente, determinando a quantidade mínima de eventos básicos que causam o evento principal, ou quantitativamente, obtendo a probabilidade de ocorrência do evento principal.

2.5.2 Análise de Modos de Falhas e Efeitos

A Análise de Modos de Falhas e Efeitos (FMEA, do inglês *Failure Modes I& Effects Analysis*), como definida por Papadopoulos e Hull (2013), é uma técnica indutiva, realizada de baixo para cima, onde é proposto um evento e analisado seus efeitos no restante do sistema. O resultado é uma tabela de falhas e seus efeitos no sistema.

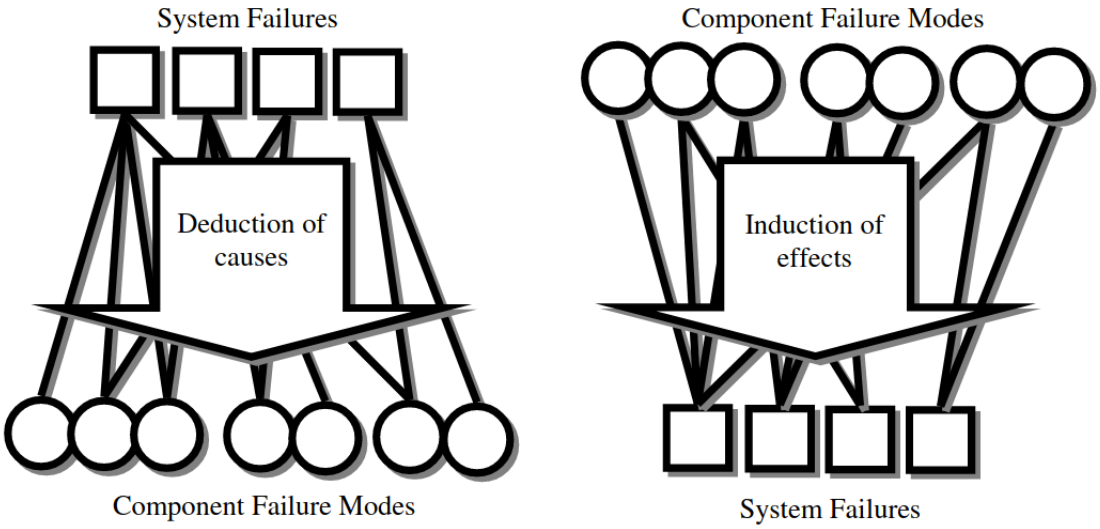

Figura 2.4: Relação Inversa Entre FTA (Esquerda) e FMEA (Direita).

2.6 HiP-HOPS

As técnicas de FTA e FMEA são uteis no processo de análise de falhas de sistemas, porém, com a complexidade enorme que sistemas podem atingir, realizá-las manualmente é trabalhoso e propenso a erros. HiP-HOPS (do inglês *Hierarchically Performed Hazard Origin I& Propagation Studies*), (PAPADOPOULOS; HULL, 2013), é uma ferramenta que utiliza o modelo do projetista para obter informações sobre a confiabilidade do sistema que está sendo modelado. Após anotar o modelo com dados de como cada componente individualmente pode falhar, HiP-HOPS gera várias árvores de falha a partir deles, e dessas árvores pode gerar uma riqueza de informações de confiabilidade apresentadas na forma de uma tabela FMEA.

2.6.1 Análise de Segurança

A análise de segurança (*Safety analysis*) feita pelo HiP-HOPS, segundo Papadopoulos e Hull (2013), acontece em três partes. A primeira fase consiste em anotar o modelo do sistema com os dados de falha necessários para produzir as árvores de falhas e realizar a análise.

Na segunda fase, acontece o processo de síntese da árvore de falhas. Nesse processo, a ferramenta examina o modelo do sistema e seus dados de falha e os combina para criar uma série de árvores de falhas. O processo funciona pegando as falhas das saídas do sistema e trabalhando para trás no modelo para determinar quais componentes causaram essas falhas. Essas falhas são então unidas usando os operadores lógicos apropriados para construir árvores de falhas com as falhas nas saídas do sistema como os principais eventos e as causas-raiz como eventos básicos.

Na terceira e última fase, o HiP-HOPS pega as árvores de falhas recém-construídas e as analisa. O resultado é um FMEA, sendo uma combinação de todas as informações armazenadas nas árvores de falhas e apresentadas na forma de uma tabela que lista os efeitos de cada falha de componente no restante do sistema. Como parte desse processo, mais análises são realizadas nas árvores de falhas, tanto qualitativas (lógicas) quanto quantitativas (numéricas). Isso fornece os conjuntos de corte mínimos da árvore de falhas e a indisponibilidade do evento superior.

2.6.2 Otimização de Projeto

A ferramenta de análise de segurança HiP-HOPS e um algoritmo de otimização multiobjetivo podem, segundo Papadopoulos e Hull (2013), ser combinados para gerar soluções de projeto com compensações não dominadas que atendem aos critérios de confiabilidade. Assim, a ferramenta pode analisar o custo e a taxa de falha de diferentes formas de projetar um mesmo modelo e gerar relatórios para o projetista.

2.7 Alocação e Decomposição de ASILs

2.7.1 Visão geral

A decomposição de ASILs definida pela ISO 26262, permite que componentes e subsistemas heterogêneos e independentes de um sistema que compartilham a responsabilidade por um determinado ASIL não precisem atender a esse ASIL para atingir os requisitos de segurança. Utilizando o Hip-HOPS para gerar FTAs e uma síntese de FMEA, teremos os conjuntos de corte mínimo do sistema, ou seja, o conjunto de eventos mínimos que geram a falha. Assim, por exemplo, se um conjunto de corte mínimo possuir dois elementos C1 e C2 responsáveis por um ASIL (B) = 2, poderemos decompor e alocar para esses elementos as seguintes configurações: C1 = ASIL(QM) e C2 = ASIL(B) e vice-versa, pois 0 + 2 = 2, ou C1 = ASIL (A) e C2 = ASIL(A), pois 1 + 1 = 2. Para determinar qual dessas duas opções é melhor, definimos heurísticas de custo para cada ASIL e assim o problema pode ser tratado computacionalmente.

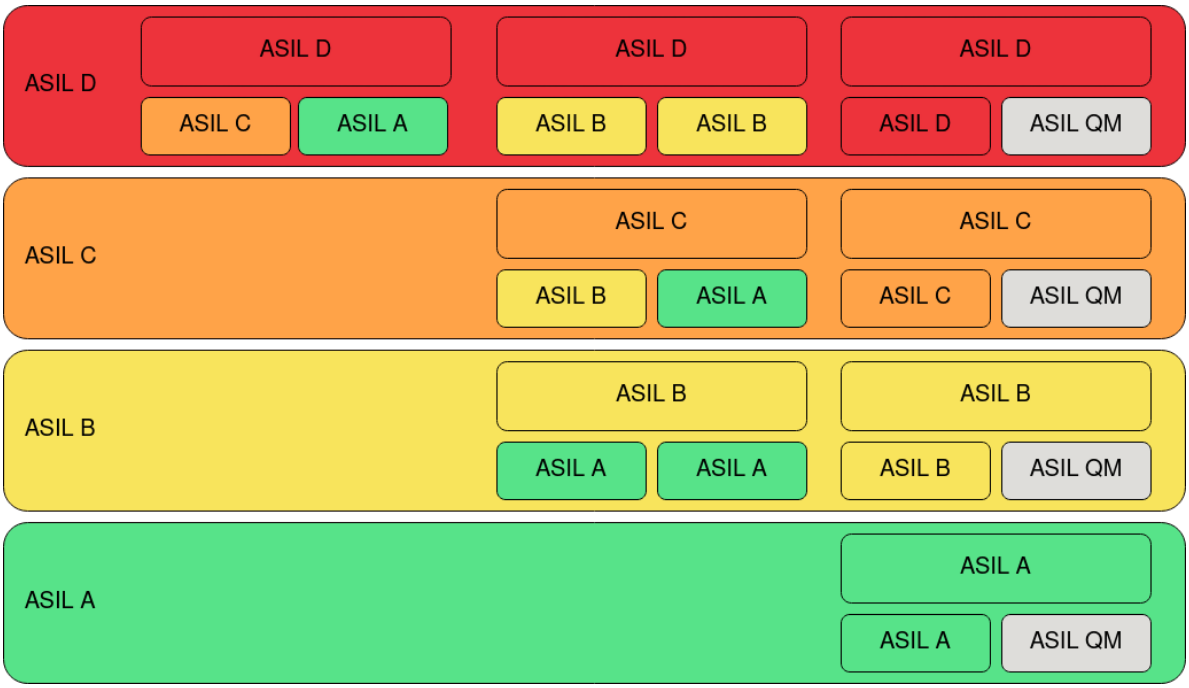

Figura 2.5: Decomposição de ASILs Definida pela (ISO, 2011)

Heurística de custo	QM	A	B	C	D
Linear	0	10	20	30	40
Logarítmico	0	10	100	1000	10000
Experimental-I	0	10	20	40	50
Experimental-II	0	5	30	35	50

Tabela 2.1: Heurística de Custo para ASILs

Como bem definido por Gheraibia et al. (2018), A decomposição e alocação de ASILs visam garantir o cumprimento dos requisitos de segurança durante o ciclo de vida de desenvolvimento do sistema, por um lado, e reduzir o custo de desenvolvimento, por outro. Assim, encontrar uma alocação e decomposição de ASILs apropriadas é uma tarefa crítica e crucial devido ao enorme tamanho do espaço de busca de soluções.

As abordagens da literatura são categorizadas em métodos por *solvers* exatos e métodos por meta-heurísticas. Os métodos por *solvers* exatos possuem a garantia de que a solução encontrada será a melhor possível, porém seu tempo de processamento é grande. Por outro lado, o tempo de processamento dos métodos por meta-heurísticas são consideravelmente mais rápidos, porém não existe garantia de encontrar a melhor solução possível, mas uma solução próxima o suficiente da ótima.

2.7.2 Busca Tabu

A Busca Tabu é uma meta-heurística que utiliza estruturas de memória para guiar uma busca local. Essa memória é uma lista de movimentos proibidos (Tabu) que inibe repetições e previne que a solução fique presa em um ótimo local. O método de Descida mais Íngreme, Subida mais Suave (SDMA, do inglês *Steepest Descent Mildest Ascent*) é uma adaptação feita por Azevedo et al. (2013) do método Subida mais Íngreme, Descida mais Suave (SAMD, do inglês *Steepest Ascent Mildest Descent*) de Hansen e Jaumard (1990), um algoritmo da família da Busca Tabu, que visa minimizar o custo da decomposição de ASILs.

O SDMA segue a direção de descida mais íngreme até que um mínimo local seja alcançado e, em seguida, usa a rota de subida mais suave para escapar dele. O

retorno ao mínimo local é evitado por meio de uma estrutura de memória adaptativa que proíbe movimentos reversos durante um número pré-definido de iterações, p. A busca é interrompida após um número pré-definido de repetições sem melhorar a atual melhor solução.

2.8 Considerações Finais

Neste capítulo, foram apresentados os conceitos e a terminologia relacionados à segurança em sistemas automotivos, destacando a importância dos padrões IEC 61508 e ISO 26262 para garantir a segurança funcional de sistemas críticos de software. Também foram discutidas as técnicas de análise de falhas, como FTA e FMEA, e a utilização da ferramenta HiP-HOPS para automatizar essas análises. Adicionalmente, abordou-se o problema de alocação e decomposição de ASILs e a aplicação da meta-heurística da Busca Tabu como uma solução potencial. A compreensão dessas bases teóricas é fundamental para sustentar a pesquisa. Ao estabelecer esse alicerce teórico, este trabalho busca contribuir para o avanço do conhecimento e enfrentar os desafios contemporâneos na otimização do projeto de sistemas críticos de software.

3 Trabalhos Relacionados

Neste capítulo, são apresentados os trabalhos relacionados às técnicas e métodos para a solução do problema de alocação e decomposição de ASILs. A revisão dos trabalhos foi conduzida seguindo o método de revisão sistemática descrito por Kitchenham (2004), que inclui as etapas de planejamento — definição dos objetivos e questões de pesquisa, escolha das bases de dados e elaboração da string de busca, além dos critérios de inclusão e exclusão — execução da busca, análise e seleção dos estudos primários, e, por fim, relato dos resultados.

O objetivo é oferecer uma visão abrangente dos trabalhos relacionados ao presente estudo, compreendendo como o problema tem evoluído em termos de abordagens meta-heurísticas e métodos exatos. Ao final, todos os trabalhos apresentados são comparados, e uma conclusão sobre o estado da arte do problema estudado e sua relação com o presente trabalho é elaborada.

3.1 Planejamento

Esta revisão aborda três questões, apresentadas a seguir, com suas métricas para quantificar as evidências e apoiar a análise dos resultados:

Questão 1: Quais sãos os métodos propostos ou usados nos últimos anos para resolver o problema de alocação e decomposição de ASILs? **Justificativa:** Obter um panorama abrangente dos métodos existentes para a resolução do problema e quais deles tiveram envolvimento da academia e da indústria em seu projeto. **Métrica:** Conjunto de métodos encontrados na literatura científica.

Questão 2: Como os existentes métodos são caracterizados? **Justificativa:** Entender o contexto, proposta e perspectiva de cada método. **Métrica:** A quantidade de métodos em cada contexto, propósito e perspectiva, além dos relacionamentos entre elas.

Questão 3: Quais são os resultados ao aplicar os existentes métodos no problema de alocação e decomposição de ASILs? **Justificativa:** Avaliar e comparar os resultados

encontrados em cada método. **Métrica:** Características descritas em cada método e os dados de testes disponibilizados.

Para selecionar os trabalhos, foi definida uma *string* de busca com base em duas palavras-chave principais: (i) *constrained optimization* como um termo abrangente da questão do trabalho para obter muitos estudos, e *automotive safety integrity levels* para restringir a busca ao problema da alocação e decomposição de ASILs.

Também são definidos critérios de inclusão (CI) e exclusão (CE) de trabalhos relacionados:

- CI1: O trabalho propõe um método para a alocação e decomposição de ASILs.
- CI2: O trabalho cita a utilização de métodos para a alocação e decomposição de ASILs.
- CE1: O estudo aborda (propõe ou utiliza) um método para a alocação e decomposição de SILs, mas não para o domínio automotivo (ASIL).
- CE2: O estudo aborda (propõe ou utiliza) um método para a alocação e decomposição de ASILs, mas existe outro estudo mais completo relacionado ao mesmo método.
- CE3: O estudo não traz informações detalhadas sobre um método para a alocação e decomposição de ASILs, pois se trata de sumário, descrição de minicurso, palestra, resumo de eventos, entre outros.
- CE4: O estudo foi escrito em outro idioma que não o inglês ou português.

3.2 Condução

A revisão foi conduzida em cerca de um mês, adaptando a *string* de busca para cada base de dados e considerando a busca por título, resumo e palavras-chave. Utilizando as questões de revisão e os critérios de inclusão e exclusão obteve-se sete trabalhos, sendo que seis deles foram utilizados para a apresentação e coleta de dados para a revisão realizada no presente trabalho.

3.3 Coleta de Dados

Nesta seção são descritas as contribuições, benefícios e limitações de cada estudo selecionado. Também contém uma tabela comparativa entre os métodos otimização apresentados.

3.3.1 Otimização por Busca Tabu

No trabalho de Azevedo et al. (2013), os autores apresentam uma nova técnica metaheurística que se utiliza da Busca Tabu para realizar buscas eficientes no espaço de solução, baseando-se no HiP-HOPS para obter uma automação escalável da decomposição de ASILs. A técnica é testada usando o estudo de caso de um sistema de frenagem hibrido.

Os autores utilizam o algoritmo Subida mais Íngreme, Descida mais Suave (SAMD, do inglês *Steepest Ascent Mildest Descent*), (HANSEN; JAUMARD, 1990), membro da família de algoritmos da Busca Tabu, com base em sua aplicação para a otimização da confiabilidade de sistemas por Hansen e Lih (1996). Assim, criando o método de Descida mais íngreme, Subida mais suave (SDMA, do inglês *Steepest Descent Mildest Ascent*). Partindo de uma solução viável, o algoritmo busca a descida mais ingrime decrementando o valor do ASIL da falha resultante na maior redução de custo. Se decrementar o valor de um ASIL ferir as regras de decomposição, então um mínimo local foi atingido e uma subida mais suave aumentando o valor do ASIL da falha resultando no menor aumento de custo é buscada. Variáveis de controle são usadas para guardar o número de iterações onde o valor de um ASIL não pode ser decrementado após um movimento de aumento e vice-versa, ou seja, o número de iterações onde o valor de um ASIL não pode ser aumentado depois de um movimento de decremento. Como as variáveis que controlam o movimento de subida são atualizadas mais tardiamente do que as que controlam movimentos de descida, a diversificação da direção de busca é incentivada.

O estudo de caso é modelado pelos autores como apenas uma unidade de frenagem, onde são considerados dois perigos para o sistema de frenagem híbrido: nenhuma frenagem após o comando (H1) e frenagem com valor incorreto (H2). ASILs D e A foram atribuídos a H1 e H2, respectivamente, baseado em suas severidades. Foi determinado que H1 é causado pela omissão de saída de ambos os dispositivos de interrupção e H2 por

um desvio de saída de valor incorreto de pelo menos um deles. Após utilizar o HiP-HOPS para gerar as FTAs do sistema foi obtido 19 MCSs para H1 e 11 MCSs para H2.

Os autores repetiram o algoritmo 10 vezes para cada função de custo com o número máximo de iterações sem melhora determinado para 5000. Para cada heurística de custo foi identificada a priori a melhor solução com o uso de uma técnica exaustiva. Na tabela, Best representa o custo dos ótimos previamente identificados; NBest conta quantas execuções encontraram o ótimo global; Dev mostra o desvio médio quando a solução ótima não foi encontrada; Iter fornece a iteração do algoritmo quando a melhor solução de cada execução foi encontrada e a CPU lista o tempo de processamento (em segundos).

	Best	Nbest	Dev	Iter	CPU
Linear	380	9	2.6%	58	0.001
Logarítmica	11300	10	0%	58	0.001
Experimental	390	9	13.2%	2325	0.02

Tabela 3.1: Resultados dos testes

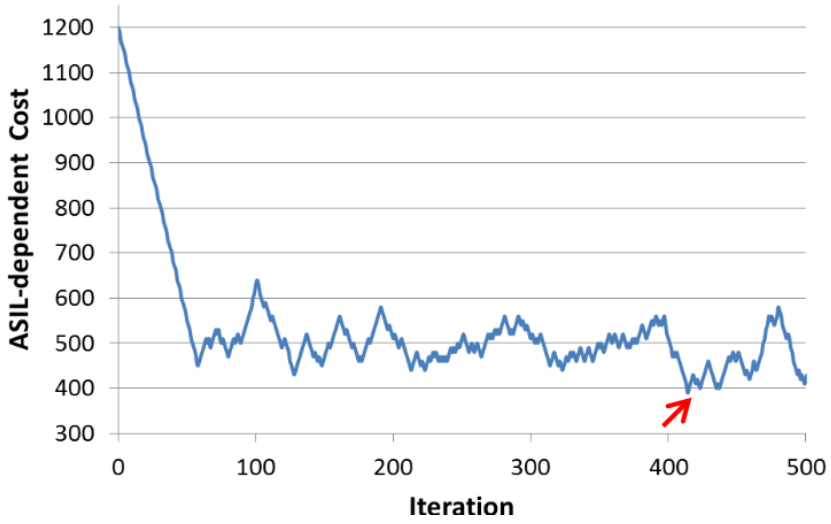

Figura 3.1: Resultado do teste com a heurística de custo experimental.

A técnica utilizada pelos autores é um método de meta-heurística flexível ao lidar com problemas com várias características, que ao ser aplicada em sistemas de larga escala entrega resultados quase ótimos com um tempo mais satisfatório que *solvers* exatos.

Porém, a técnica não garante encontrar a solução ótima para o problema e os autores não confirmaram os resultados em sistemas maiores.

Como a técnica é baseada em Busca Tabu, ela consegue evitar ótimos locais e ter uma maior variedade no espaço de busca. Porém, tem-se a certeza de que a técnica, em seu movimento de descida mais íngreme, irá encontrar um mínimo local e como é uma busca única, dado um cenário ruim, ao encontrar um mínimo local, pode precisar de muitas iterações para realizar os movimentos de subida mais suave e escapá-lo.

3.3.2 Otimização por Caça em Grupo de Pinguins

No trabalho de Gheraibia et al. (2015), os autores apresentam uma nova técnica meta-heurística baseada no comportamento de caça colaborativa de pinguins chamada Algoritmo de otimização da Busca de Pinguins (PeSOA, do inglês *Penguins Search Optimisation Algorithm*). Usando a metáfora de reservas de oxigênio como operador de busca, é possível que pinguins preservem energia e a utilize apenas em áreas do espaço de busca com boas soluções. A técnica é testada usando o estudo de caso de um sistema de frenagem hibrido.

A população de pinguins inicia em posições aleatórias. Cada pinguim mergulha em busca de soluções e consome sua reserva de oxigênio. Enquanto busca, um pinguim pode modificar sua solução inicial usando o valor da reserva de oxigênio. Pinguins em um bom caminho ganham uma reserva de oxigênio maior que os permitem continuar na mesma direção. Já pinguins que descobrem más soluções tem sua reserva de oxigênio diminuída. O processo é repetido até que a taxa de erro seja verificada, ou o número máximo de iterações atingido. Pinguins podem comunicar entre si e compartilhar informações que levam a imigrações entre grupos dado uma função que considera abundância de alimentos para cada grupo. Um pinguim se junta a um grupo com probabilidade proporcional a quantidade de peixes comidos pelo grupo correspondente, aumentando a chance de encontrar boas soluções nas próximas iterações. Uma região explorada por um grupo é abandonada quando todos os membros do grupo emigram.

A adaptação do PeSOA para o problema de alocação de ASILs começa com a geração de classes de prioridade para cada combinação de ASILs. As prioridades são

utilizadas para estimar o tamanho de cada população. O processo continua com a geração aleatória de populações de pinguins com uma solução viável para o problema. A população é divida em diversos grupos para cobrir diversas áreas do espaço de busca e escapar do ótimo local. A divisão é feita com base na frequência de cada modo de falha nos componentes de corte mínimo. Para cada grupo de pinguins é atribuído um conjunto de modos de falha na qual modificarão durante a busca pela melhor alocação através do sistema. Cada pinguim gera de sua posição soluções vizinhas e escolhe aquela com menor custo que satisfaça as restrições de integridade de segurança do sistema. A reserva de oxigênio é atualizada privilegiando os pinguins que encontraram boas soluções e os pinguins são redistribuídos entre os grupos refletindo a melhoria de cada pinguim.

Os autores testaram técnica usando como estudo de caso um sistema de frenagem hibrido. Um total de 6 perigos foi definido para o HBSM3 e ASILs foram atribuídos a cada um deles:

- Sem frenagem nas 4 rodas = ASIL D;
- Sem travagem nas 3 rodas = ASIL D;
- Frente sem frenagem = ASIL D;
- Sem frenagem traseira = ASIL C;
- Sem diagonal de frenagem = ASIL C;
- Frenagem com valor errado 4 rodas = ASIL D.

Um total de 60 falhas de componentes foram identificadas no sistema e a aplicação da Análise de Árvore de Falhas revelou 11.573 conjuntos de cortes. Isso produz um grande espaço de solução potencial na ordem de 10^41 alocações potenciais.

Algorithm	MinCost	GA	TS	PeSOA
Linear	770	15.51	4.44	0.27
Experimental-I	830	14.63	14.70	0.59
Experimental-II	620	3.31	5.72	0.89
Logarítmica	51150	3.04	1.91	0.32

Tabela 3.2: Comparação do PeSOA com o Algoritmo Genético (GA) e Busca Tabu (TS) (média do tempo de CPU em segundos)

Como demonstrado pelos autores, o algoritmo é melhor que as principais meta-heurísticas até então utilizadas, Busca Tabu e Algoritmo Genético, e possui uma convergência mais rápida. Como é uma meta-heurística, PeSOA não possui garantia de encontrar a solução ótima, mas possui resultados satisfatórios para sistemas grandes e complexos. Como existem vários grupos de pinguins e aqueles que estão em direção a soluções melhores são recompensados, os grupos em soluções ruins são dissolvidos em emigrações para grupos em posições melhores do espaço de busca e o problema evita ótimos locais, apresentando uma convergência rápida.

3.3.3 Otimização por Sistemas de Equações Lineares

No trabalho de Dhouibi et al. (2014), os autores propõem uma solução exata para o problema de alocação de ASILs interpretando os componentes de corte mínimo como equações lineares onde uma alocação possível seria a solução da seguinte equação que respeita as restrições de integridade de segurança do sistema:

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \times \begin{bmatrix} ASIL_{F_1} \\ \vdots \\ ASIL_{F_n} \end{bmatrix} = \begin{bmatrix} ASIL_{SR_1} \\ \vdots \\ ASIL_{SR_n} \end{bmatrix}$$

Obtido a matriz aumentada e considerando todas as restrições, ela pode ser resolvida. Várias abordagens podem ser utilizadas para a resolução, porém como há infinitas soluções e a matriz nem sempre é quadrada, muitas delas não são viáveis. Os autores optam então pela Forma escalonada reduzida por linhas (RREF, do inglês *Row Reduced Echelon Form*). O RREF é resolvido computacionalmente utilizando-se do método de eliminação de Gauss-Jordan que permite identificar as variáveis básicas e livres que corresponde às colunas sem entrada inicial. Para encontrar soluções, as variáveis da matriz de equações são definidas em números de 0 a 4 que correspondem à álgebra da ISO 26262 para ASILs.

Para o teste, os autores consideram um sistema com dois requerimentos de segurança (SR1 e SR2) com ASIL D e ASIL C, respectivamente. As funções de perda vão de F1 a F5 onde:

- A perda de F1 = ASIL D;
- A perda de F2, F3 e F4 = ASIL D;
- A perda de F3, F4 e F5 = ASIL C.

Convertidas para as seguintes funções:

$$ASIL(F1) = 4$$
$$ASIL(F2) + ASIL(F3) + ASIL(F4) = 4$$
$$ASIL(F3) + ASIL(F4) + ASIL(F5) = 3$$

Portanto, as possíveis alocações são soluções da seguinte equação:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} ASIL_{F_1} \\ \vdots \\ ASIL_{F_5} \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 3 \end{bmatrix}$$

Aplicado a solução ao exemplo e outros sistemas genéricos, foi possível encontrar todas as soluções possíveis. A tabela a seguir mostra a comparação entre o algoritmo deste trabalho (Algoritmo 1) e o algoritmo proposto por Abele e Librino (2012) (Algoritmo 2).

Exemple size	Nbr possible allocations	Alg 1	Alg 2
5 Functions, 3 MCS	10	0.01	0.1
10 Functions, 19 MCS	1	0.09	356.16
24 Functions, 30 MCS	3	0.04	0.34
48 Functions, 44 MCS	75	0.76	1.4

Tabela 3.3: Tempo de processamento dos testes

Essa abordagem exata, além de garantir a solução ótima para o sistema, encontra todas as possíveis soluções, útil para o analista tomar decisões para o projeto. Porém, para sistema de larga escala ou escaláveis, essa abordagem possui um tempo de processamento que pode ser tanto insatisfatório quanto inviável. Outra vantagem desse método é que a

conversão do problema de alocação e decomposição de ASILs para um sistema de equações lineares é simples e direto, existindo várias técnicas na literatura de Álgebra Linear para resolver tais sistemas.

3.3.4 Otimização por Algoritmo Genético Baseado em Penalidades

No trabalho de Parker et al. (2013), os autores apresentam uma nova técnica para a resolução do problema de alocação de ASILs que utiliza um algoritmo genético baseado em penalidade para explorar eficientemente o espaço de busca e identificar alocações ótimas de ASILs aos componentes do sistema. A técnica é testada usando o estudo de caso de um sistema de frenagem hibrido.

Algoritmos genéticos (GA, do inglês *Genetic Algorithm*) são técnicas meta-heurísticas que imitam o processo biológico de evolução para eficientemente explorar grandes espaços de busca para encontrar soluções ótimas. O algoritmo genético descrito pelos autores é baseado no algoritmo genético baseado em penalidade usado por Coit e Smith (1996). O processo do GA pode ser resumido da seguinte forma: inicializa aleatoriamente uma população de soluções potenciais (neste caso, uma solução é um conjunto de ASILs alocados para cada modo de falha do componente); gerar novas soluções candidatas para a população da próxima geração por meio de cruzamento ou mutação; adicionar os próximos candidatos à população; classificar a população por aptidão; remover os candidatos menos aptos até um certo limite da população; e por fim, continuar gerando novas soluções candidatas até que o limite de geração seja atingido.

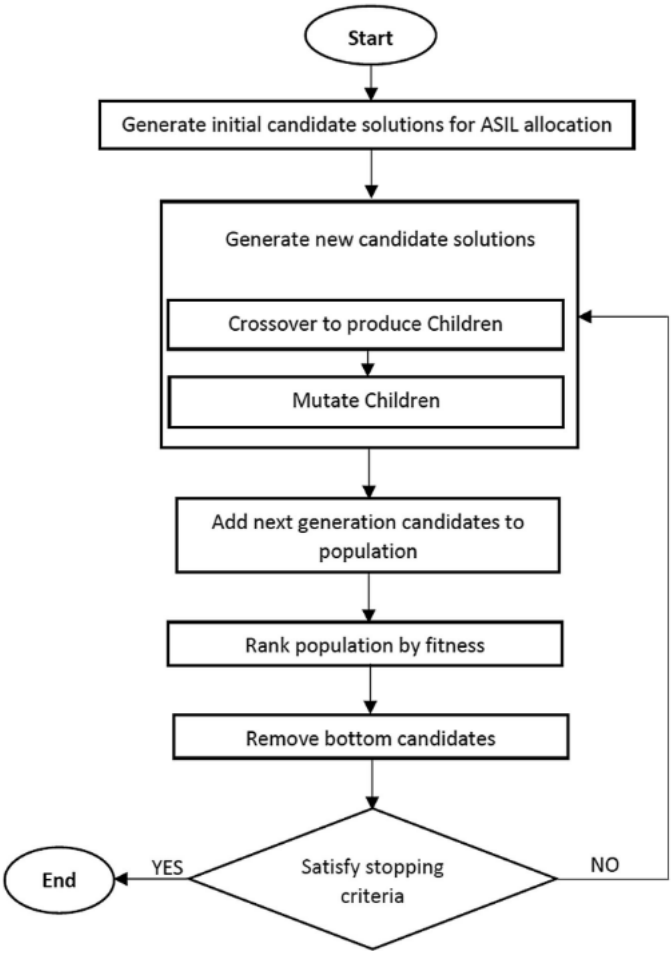

Figura 3.2: Fluxograma do algoritmo genético baseado em penalidade para o problema de alocação de ASILs.

Para o problema de alocação de ASILs, esta abordagem usa uma codificação de comprimento fixo onde cada "caso" pode conter um valor de 0 a 4 (QM a D, respectivamente). A função de aptidão definida é igual à soma dos custos ASILs de cada solução candidata. Soluções iniciais para o problema de alocação de ASILs são geradas aleatoriamente e operadores genéticos são aplicados para criar soluções candidatas, manipulando as existentes selecionadas na fase anterior. Candidatos são classificados de acordo com sua aptidão penalizada e a seleção é feita em quão perto o candidato está de um número escolhido aleatoriamente entre 1 e a raiz quadrada do tamanho da população. As novas soluções geradas são integradas na população e todas a soluções são ranqueadas. Durante o processo de ranqueamento, as soluções candidatas com menor valor de aptidão são favorecidas, a menos que sejam consideradas soluções inviáveis. O método de penalidade

utilizado pelos autores é aplicado para aumentar a penalidade em soluções candidatas inviáveis, ou seja, reduz a aptidão de soluções candidatas que violam as restrições à medida que a busca progride. Um conjunto de conjuntos mínimos de corte (MCSs) da análise da árvore de falhas usando HIP-HOPS é usado para verificar a viabilidade de soluções candidatas.

Para o estudo de caso é considerado um sistema de frenagem hibrido com os seguintes perigos: sem frenagem após comando (H1) e frenagem de valor errado (H2). Existem 24 modos de falha de componentes no sistema, resultando em um comprimento de codificação de 24 atribuições de ASILs possíveis. Isso dá um tamanho de espaço de pesquisa de 5.96×10^{16}. Utilizando o HiP-HOPS para gerar FTAs, temos 19 MCSs para H1 e 11 MCSs para H2. O algoritmo genético foi repetido 10 vezes, cada execução sendo 10000 gerações. O tamanho da população foi de 500, e a taxa de mutação e taxa de cruzamento foram de 0.05 e 0.9, respectivamente.

Cost heuristics	HBSM 1	HBSM 2	HBSM 3
Linear	1.04	4.67	15.51
Experimental-I	1.21	7.32	14.63
Experimental-II	1.07	2.47	3.31
Logarithmic	1.98	2.35	3.04

Tabela 3.4: Tempo médio de execução em segundos para encontrar a solução otimizada usando a abordagem GA baseada em penalidade

A mutação estocástica da população pela solução dos autores permite uma busca eficiente pelo espaço de busca do problema e gera soluções em um tempo satisfatório em sistemas de grande escala e escaláveis. A mutação, cruzamento e as penalidades são características muito importantes para a evolução do algoritmo no espaço viável de soluções, então, carregar uma má mutação pode fazer o algoritmo demorar a conseguir uma boa população. Como o método é uma meta-heurística, não há garantia de encontrar a solução ótima.

3.3.5 Otimização por Colônia de Formigas

No trabalho de Gheraibia, Djafri e Krimou (2018), os autores apresentam uma nova abordagem que usa o algoritmo meta-heurístico otimização por colônia de formigas (ACO, do inglês *Ant Colony Optimization*) inspirado na natureza para resolver o problema de alocação de ASILs e que usa estratégias que reduzem o espaço de busca. O problema foi formulado como um grafo de construção, que as formigas usam para construir possíveis alocações de ASIL. Essa abordagem foi avaliada aplicando-a a um sistema de frenagem híbrido e a um sistema de direção por fio.

O ACO simula comportamentos de forrageamento de formigas: enquanto procuram por comida, as formigas conseguem encontrar o caminho mais curto entre o formigueiro e a fonte de alimento. As formigas operárias depositam feromônios ao longo do caminho até a fonte de alimento. O feromônio é uma comunicação indireta entre as formigas. Essa troca de informações, conhecida como estigmergia, ocorre no nível do solo. O princípio da estigmergia é que o traço deixado no ambiente por uma ação estimula a realização da próxima ação pelo mesmo ou por diferentes indivíduos. Quando uma formiga tem que decidir sobre qual caminho seguir, a escolha está relacionada a dois fatores principais, sendo a concentração de feromônio e a informação heurística sobre o caminho.

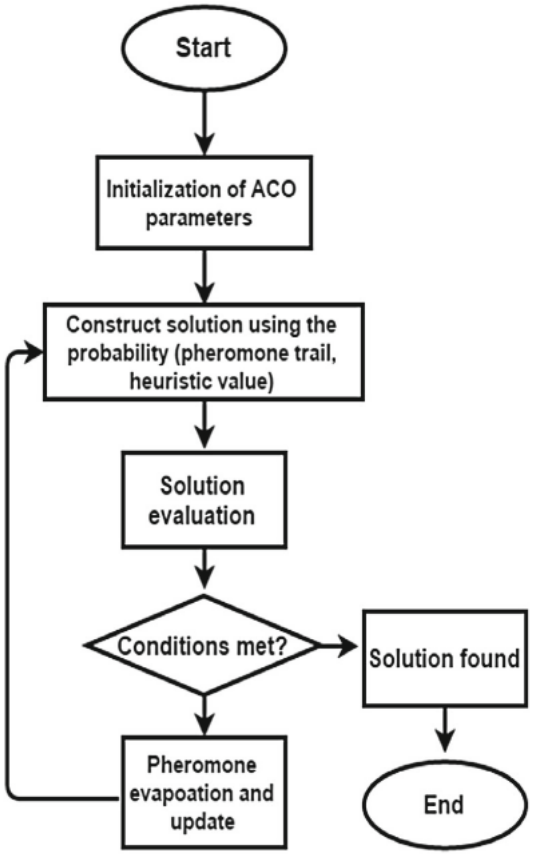

Figura 3.3: Fluxograma do algoritmo de otimização de colônia de formigas.

Antes do processo de otimização começar, ACO utiliza duas técnicas para diminuir o espaço de busca. A primeira estratégia é baseada nos valores da função custo. Após analisar as diferentes combinações de ASILs dos modos de falhas, os autores constataram que conforme a função custo, é possível descartar algumas combinações. A segunda estratégia de redução do espaço de busca é baseada nos conjuntos de corte. Essa abordagem visa reduzir o espaço de busca reduzindo o número de ASILs possíveis para cada modo de falha. O algoritmo de colônia de formigas usa essas estratégias para alcançar a alocação ideal de ASILs em um tempo razoável.

O problema de alocação de ASILs foi formulado como um grafo onde os nós representam os modos de falha e as arestas representam o ASIL alocado ao nó destino dessas arestas. Esta nova representação do problema de alocação de ASILs ajuda as formigas a explorar o espaço de busca eficientemente. Cada formiga explora o grafo e, ao final, atualiza o valor do feromônio para comunicar a outras formigas a qualidade do caminho explorado. Desta forma, as formigas podem tomar decisões informadas enquanto

exploram o espaço de busca, o que por sua vez resulta em uma solução melhorada.

O algoritmo de colônia de formigas para alocação de ASILs foi avaliado usando dois estudos de caso diferentes: o sistema de frenagem híbrido e o sistema de direção por fio. A Tabela 3.5 mostra os resultados para as três variantes do HBSM.

Cost heuristics	HBSM 1	HBSM 2	HBSM 3
Linear	0.003	0.01	0.16
Experimental-I	0.007	0.02	0.35
Experimental-II	0.009	0.03	0.62
Logarithmic	0.008	0.02	0.20

Tabela 3.5: Tempo médio de execução em segundos para encontrar a solução otimizada usando a abordagem baseada em ACO

Como demonstrado pelos autores, o ACO, em termos de tempo de execução, supera todas as abordagens existentes até então. Com a técnica de comunicação indireta que imita os feromônios utilizado pelas formigas e as estratégias que reduzem o espaço de busca, o algoritmo consegue buscar eficientemente pelo espaço de busca e resolver o problema com solução ótima ou quase ótima em um tempo satisfatório. Como toda meta-heurística, não há garantia de encontrar a solução ótima.

3.3.6 Otimização por Programação Linear

No trabalho de Mader et al. (2012), os autores propõem um método que permite a alocação automática de ASILs aos componentes do sistema ao abordar o problema de alocação de ASILs como um problema de programação linear.

Um problema de programação linear visa maximizar ou minimizar um conjunto de equações lineares com variáveis e dados inteiros dado um conjunto de restrições. As restrições usadas na abordagem proposta são identificadas na análise preliminar de riscos, extraídas dos resultados da análise da árvore de falhas e definidas pelo engenheiro de segurança como preferências de segurança.

Essa abordagem usa um framework de modelagem para criar um EAST-ADL (*Electronic Architecture and software Technology-Architecture Description language*), usado

para descrever os componentes do sistema e gerar árvores de análise de falhas cujo conjunto mínimo de corte é utilizando como entrada para o *solver* de programação linear inteira do framework. O framework também possui um *solver* de restrições que permitirá encontrar a solução ótima para o problema de alocação de ASILs. Há dois tipos principais de restrição, a primeira é relacionada com os conjuntos mínimos de corte, onde, para cada conjunto mínimo de corte, uma restrição de violação do objetivo de segurança é definida. A segunda é relacionada com as escolhas e preferências do engenheiro, visando diminuir o custo dos ASILs alocados ao definir um ASIL para um determinado conjunto de corte mínimo.

O *solver* de restrições é então usado para resolver o problema de programação linear inteira para encontrar a alocação ótima de ASILs para os componentes do sistema. Nessa abordagem, os autores criaram um plug-in para uma ferramenta de código aberto chamada Papyrus. Se for adequado, a alocação de ASILs aos componentes do sistema é iniciada e o modelo EAST-ADL é alterado automaticamente conforme a nova solução, caso contrário, o engenheiro de segurança terá que fazer algumas alterações em suas preferências e executar novamente o *solver*.

A principal característica dessa solução é a criação de um *framework* onde os projetistas ou engenheiros de *software* podem importar um modelo do sistema e automaticamente gerar as análises de falhas, também é permitido a criação de restrições customizáveis para o *solver* de programão inteira integrado no *framework* que permitem uma análise mais avançada do projeto desenvolvido. A solução gerada é ótima apenas em relação às restrições definidas pelos projetistas ou engenheiros.

3.3.7 Considerações Finais

Neste capítulo foram apresentadas seis abordagens para a resolução do problema de alocação e decomposição de ASILs. Das abordagens analisadas, foram identificadas técnicas basedas em meta-heurísticas, que fornecem soluções próximas da ótima do problema em um tempo satisfatório e o escalonamento do sistema; mas que, em contrapartida, não garantem a solução ótima. Também, há técnicas exatas de *solvers* que permitem uma grande configuração por parte do usuário e a solução ótima, ou pelo menos ótima em

relação às restrições definidas; porém que possuem um tempo de execução insatisfatório ou inviável para sistemas de grande escala.

Na Tabela 3.6, é apresentado um resumo das principais características dos métodos estudados, sendo eles Busca Tabu (TS), Sistema de Equações Lineares (SEL), Algoritmo de otimização da Busca de Pinguins (PeSOA), Algoritmo Genético baseado em penalidade (GA), Algoritmo de Otimização de Colônia de Formigas (ACO) e Programação Linear (PL):

	TS	SEL	PeSOA	GA	ACO	PL
Tipo	Meta-heurística	Exato	Meta-heurística	Meta-heurística	Meta-heurística	Exato
Solução ótima?	Não	Sim	Não	Não	Não	Sim
Customizável?	Não	Não	Não	Não	Não	Sim
Tempo viável em sistemas complexos?	Sim	Não	Sim	Sim	Sim	Não

Tabela 3.6: Características dos Métodos Estudados

Na Tabela 3.7 é apresentada uma análise comparativa do tempo médio de execução em segundos dos trabalhos meta-heurísticos estudados utilizando a heurística de custo Experimental-II em três variantes do HBSM: Busca Tabu (TS), Algoritmo de otimização da Busca de Pinguins (PeSOA), Algoritmo Genético baseado em penalidade (GA) e Algoritmo de Otimização de Colônia de Formigas (ACO). Os dados foram retirados dos seus respectivos trabalhos ou de Gheraibia et al. (2018).

Técnicas	HBSM 1	HBSM 2	HBSM 3
GA	1.07	2.47	3.31
TS	0.84	0.79	5.72
PeSOA	0.10	0.41	0.89
ACO	0.009	0.03	0.62

Tabela 3.7: Tempo Médio de Execução em Segundos para Encontrar a Solução Otimizada Usando as Abordagens Apresentadas

Nas soluções meta-heurísticas, constatamos a característica de utilizar técnicas para uma busca mais abrangente pelo espaço viável de soluções, evitando mínimos locais

para uma convergência mais rápida em uma boa solução. Como mostrado pela Tabela 3.7, a Busca Tabu foi o método meta-heurístico com o maior tempo de processamento no maior sistema testado (HBSM 3), justamente por não tentar evitar os mínimos locais, mas seguir estratégias para escapá-los depois de já os terem atingindo. Os trabalhos com os métodos de *solver* exatos, não apresentam o tempo de processamento, mas em compensação, os autores evidenciam a customização e flexibilidade possível na utilização de tais métodos.

4 Decomposição de ASILs Usando a Busca Tabu

Este capítulo apresenta a abordagem utilizada para implementar o algoritmo de Busca Tabu no contexto do problema de alocação e decomposição de ASILs em sistemas de segurança crítica. A implementação da Busca Tabu baseia-se no trabalho de Hansen e Jaumard (1990), que aplicou o procedimento *Steepest Ascent Mildest Descent* (SAMD) para explorar o espaço de busca, visando encontrar uma solução de decomposição de ASILs próxima da ótima para uma determinada arquitetura de sistema.

Na Seção 4.1, é apresentada a representação da solução para o problema de alocação e decomposição de ASILs utilizando a Busca Tabu, além do processo de geração da solução inicial. A Seção 4.2 detalha a etapa de exploração do espaço de busca do algoritmo proposto. Em seguida, na Seção 4.3, é descrito o mecanismo de memória, que compreende as listas tabu utilizadas para determinar se um movimento (SAMD ou SAMA) será permitido ou proibido no momento da execução. Na Seção 4.4, o pseudo-código e o fluxograma do algoritmo são apresentados e explicados em detalhe. A Seção 4.5 ilustra o passo a passo do algoritmo por meio de um exemplo ilustrativo. Por fim, na Seção 4.6, um exemplo do *Hybrid Braking System* (HBS) é utilizado para testar a viabilidade do algoritmo proposto em cenários do mundo real.

4.1 Representação de uma Solução e Geração da Solução Inicial

Para a implementação do algoritmo da Busca Tabu apresentada nesta seção, uma codificação de números inteiros em tamanho fixo é utilizada para armazenar o valor do ASIL de cada Failure Mode (FM) no sistema. O número em cada posição do vetor varia de 0 a 4 sendo relacionados aos números atribuídos aos ASILs pela álgebra de ASIL (ASIL QM = 0, ASIL A = 1, ASIL B = 2, ASIL C = 3 e ASIL D = 4). Um exemplo é dado na

Figura 4.1.

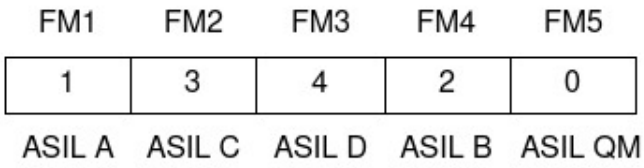

Figura 4.1: Representação da solução de alocação de ASIL.

O custo total de uma solução visitada pela Busca Tabu é dado pela soma dos custos associados com os ASILs alocados nos diferentes FMs. Para o exemplo dado, usando a heurística de custo logarítmica (ASIL QM – custo 0, ASIL A – custo 10, ASIL B – custo 100, ASIL C – custo 1000 e ASIL D – custo 10000), o custo da solução é de 11110 unidades (10 + 1000 + 10000 + 100 + 0).

A Busca Tabu não permite a exploração em espaços de busca inviáveis; e por consequência, demanda uma técnica para gerar uma solução inicial viável. A abordagem concebida por Azevedo (2015) começa por atribuir cada FM com um ASIL aleatório entre 0 e 4. O resultado provavelmente será uma solução inviável. Em seguida, um FM cujo ASIL atribuído não seja igual a 4 e que participe de pelo menos uma restrição de alocação de ASIL que não é satisfeita, é escolhido aleatoriamente para ter seu ASIL incrementado. Essa ação é repetida até que a solução se torne viável.

4.2 Exploração da Vizinhança

O algoritmo sempre inicia com uma solução viável e toda nova alocação é o resultado de uma avaliação do conjunto de possíveis mudanças na solução atual. Mudanças que geram soluções inviáveis são descartadas imediatamente.

A otimização começa buscando o movimento de descida mais íngreme, pois, como toda heurística de custo para ASILs é estritamente crescente, ao reduzir o ASIL de um FM sempre teremos uma redução no custo total do sistema. O movimento de descida mais íngreme é realizado decrementando o ASIL do FM que trará a maior redução no custo do sistema. Considere a solução na iteração t do algoritmo da Busca Tabu apresentado na

Figura 4.2.

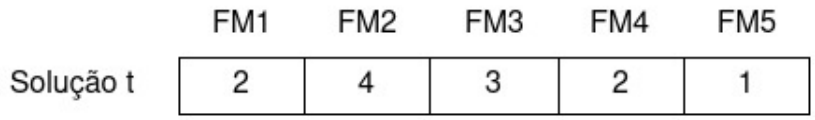

Figura 4.2: Solução ilustrativa da Busca Tabu na iteração t.

Assumindo a heurística de custo logarítmica (ASIL QM – custo 0, ASIL A – custo 10, ASIL B – custo 100, ASIL C – custo 1000 e ASIL D – custo 10000), decrementar o ASIL do FM2 representa tomar a descida mais íngreme:

- FM1 e FM4 (ASIL B para ASIL A) — redução de custo: 100 - 10 = 90
- **FM2 (ASIL D para ASIL C) — redução de custo: 10000 - 1000 = 9000**
- FM3 (ASIL C para ASIL B) — redução de custo: 1000 - 100 = 900

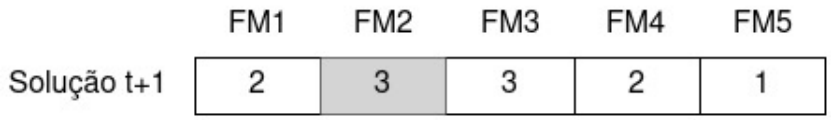

Figura 4.3: Solução ilustrativa da Busca Tabu na iteração $t + 1$.

Na análise de movimentos de descida, pode ocorrer de vários movimentos terem a mesma variação de custo que o movimento de descida mais íngreme. Nesses casos, o algoritmo considera todos esses movimentos e escolhe de forma aleatória qual será usado na próxima iteração.

Suponha agora que a solução da iteração t+1 é um mínimo local, ou seja, nenhum movimento de descida pode ser feito sem violar uma ou mais restrições de alocação de ASIL. Para escapar do mínimo local em potencial, o algoritmo toma o movimento de subida mais suave, ou seja, ele incrementa o ASIL do FM que representa o menor aumento de custo do sistema. Para a iteração t+1, incrementar o ASIL do FM5 é tomar a subida mais suave:

- FM1 e FM4 (ASIL B para ASIL C) — aumento de custo: 1000 - 100 = 900

- FM2 e FM3 (ASIL C para ASIL D) — aumento de custo: 10000 - 1000 = 9000
- **FM5 (ASIL A para ASIL B) — aumento de custo: 100 - 10 = 90**

Portanto, a solução obtida para a iteração $t + 2$ seria a apresentada na Figura 4.4.

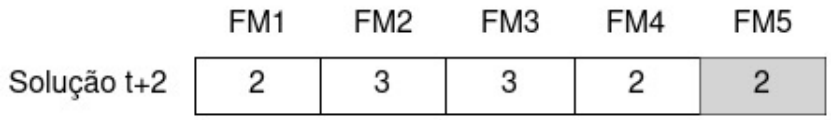

Figura 4.4: Solução ilustrativa da Busca Tabu na iteração $t + 2$.

Na ocasião de vários movimentos terem a mesma variação de custo que o movimento de subida mais suave, o algoritmo considera todos esses movimentos e escolhe de forma aleatória qual será usado na próxima iteração.

4.3 Mecanismo de Memória

Ao escapar do mínimo local pelo movimento de subida mais suave, movimentos reversos são proibidos por p iterações. Assumindo que p é maior que zero, isso implica que na iteração t+3 o ASIL do FM5 não pode ser reduzido. O número de iterações que proíbem o decremento do ASIL de um dado FM i é armazenado na variável f_i. Para aumentar a diversidade, o algoritmo também proíbe incrementar o ASIL de um FM durante q iterações após um movimento de descida. O número de iterações que proíbem o incremento do ASIL de um dado FM i é armazenado na variável f'_i.

Os valores das variáveis p e q são mudadas dinamicamente para reduzir a sensibilidade da seleção do algoritmo. Elas são incrementadas nos períodos $updatePeriod_p$ e $updatePeriod_q$, respectivamente. Também é determinado um $limit_p$ e $limit_q$ que são os valores máximos que p e q podem assumir; quando p e q chegam em seus limites, seus valores são redefinidos para zero.

A abordagem de Hansen e Jaumard (1990) foi estendida para permitir a anulação de uma restrição de um movimento tabu caso a tomada desse movimento signifique 1) a obtenção uma solução melhor do que as encontradas anteriormente ou 2) conseguir

uma solução com o mesmo custo da melhor solução atual, mas que ainda não tenha sido visitada. Essas condições que permitem a desconsideração de movimentos tabus são conhecidas como critérios de aspiração (GENDREAU; POTVIN, 2005).

4.4 Algoritmo de Alocação e Decomposição de ASIL

Para a melhor compreensão do algoritmo da Busca Tabu, a seguir são apresentados um pseudocódigo (Listagem 1) do algoritmo e seu fluxograma (Figura 4.14). Para a execução do algoritmo, é necessário fornecer os seguintes parâmetros de entrada: o número de iterações sem melhora, que atua como critério de parada; os limites das variáveis p e q, que ajustam a sensibilidade na seleção do algoritmo; os conjuntos de corte mínimo (MCS) da arquitetura específica do sistema; e o número de modos de falha (FM) associados a esse sistema. Nas linhas 2 a 7, ocorre a inicialização do algoritmo, começando pela geração da solução inicial na linha 2, seguida pela atribuição dos valores iniciais das variáveis. O laço, que se inicia na linha 8 e termina na linha 23, é condicionado ao número de iterações sem alterações na melhor solução encontrada e contém as principais operações do algoritmo.

Dentro desse laço, as linhas 9 e 10 realizam a decretação das posições da lista tabu de subida que possuem valores diferentes de zero, além de atualizarem as variáveis p e q. Na linha 11, uma estrutura condicional verifica a viabilidade de um movimento de descida, que pode ser permitido pela lista tabu de descida ou por um critério de aspiração. Se viável, a solução atual é ajustada conforme o movimento de descida, a posição correspondente do modo de falha modificado é atualizada na lista tabu de subida, e a melhor solução encontrada é substituída caso a nova solução apresente menor custo.

Caso contrário, é verificada a possibilidade de um movimento de subida. Se permitido pela lista tabu de subida, a solução atual é modificada conforme o movimento de subida, e a posição correspondente do modo de falha modificado é atualizada na lista tabu de descida. O laço se encerra e a melhor solução encontrada é retornada na linha 24 se o critério de parada for atingido, ou seja, quando N iterações sem melhorias no resultado foram completadas.

Na figura 4.5, o fluxograma é lido da seguinte forma: em cada iteração, o algoritmo verifica se há um movimento de descida não-tabu viável ou se o critério de aspiração foi

Algoritmo 1: Busca Tabu.

Entrada: N, número de iterações sem melhora
limit_p, limite para p
limit_q, limite para q
restrictions, conjunto de MCS
numFM, número de FMs
Saída: melhor solução encontrada na minimização.

```
1  início
2      solucao_atual = gera_solucao_inicial();
3      melhor_solucao = solucao_atual;
4      iter = 0;
5      melhor_iter = 0;
6      p = 0;
7      q = 0;
8      para (iter − melhor_iter) < N faça
9          decremente todas as posições da lista tabu de subida que sejam
           diferentes de zero;
10         atualize p e q;
11         se existe um movimento de descida mais íngreme viável ou critério
           de aspiração atingido então
12             solucao_atual = movimento_de_descida(solucao_atual);
13             atualize a posição do FM decrementado na lista tabu de subida
               com q + 1;
14             se custo(solucao_atual) < custo(melhor_solucao) então
15                 melhor_solucao = solucao_atual;
16             fim se
17         senão
18             se existe um movimento de subida mais suave viável então
19                 solucao_atual = movimento_de_subida(solucao_atual);
20                 atualize a posição do FM incrementado na lista tabu de
                   descida com p + 1;
21             fim se
22         fim se
23     fim para
24     retorna melhor_solucao;
25 fim
```

atingido. Se sim, o algoritmo decrementa o ASIL do FM correspondente à descida mais íngreme, recalcula o custo, e atualiza a melhor solução encontrada. Se não houver um movimento de descida viável, o algoritmo verifica se há um movimento de subida não-tabu viável e, se encontrado, incrementa o ASIL do FM correspondente à subida mais suave.

O processo continua até que o critério de parada seja atingido. Se atingido, o algoritmo apresenta a melhor solução e seu custo. Caso contrário, o algoritmo decrementa os valores das listas tabus de descida e subida para todos os FMs. Se a iteração é múltipla de um período de atualização pré-definido, as variáveis p e q são atualizadas. O processo então retorna ao início do loop, continuando até que o critério de parada seja satisfeito.

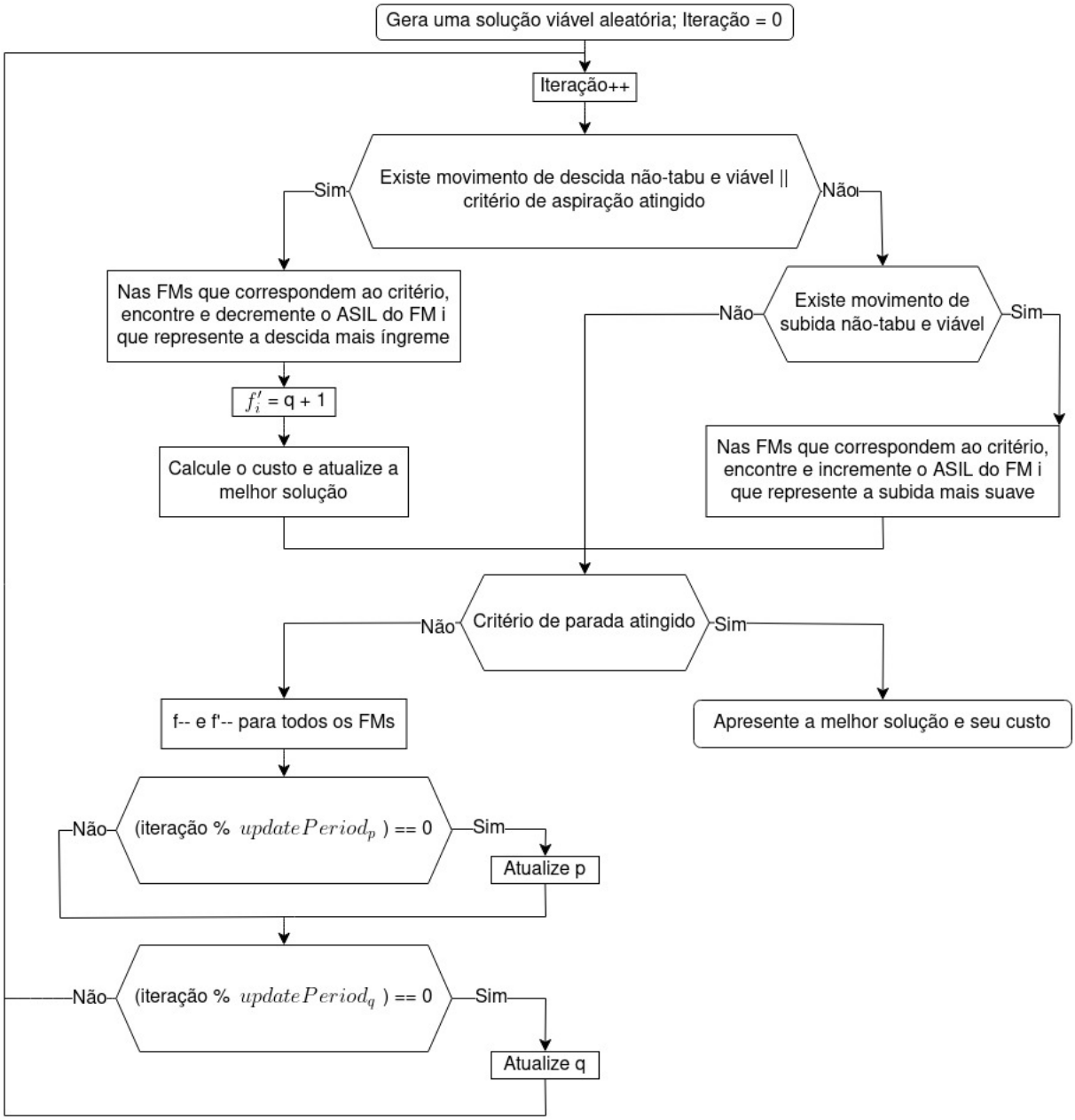

Figura 4.5: Fluxograma para o algoritmo Busca Tabu.

4.5 Execução em um exemplo ilustrativo

Nesta seção, é apresentado um exemplo ilustrativo da aplicação do algoritmo proposto para apoiar a alocação e decomposição de ASILs. Foi considerado um sistema com duas *fault trees* interconectadas e seus respectivos *safety requirements* SR1 e SR2, classificados como ASIL D e ASIL C, respectivamente, como entrada para o algoritmo. Os FMs F1 a

F5 implementam os *safety requirements*. A figura 4.6 descreve como a perda dessas FMs podem levar a violação dos *safety requirements*. SR1 e SR2 podem ser decompostos sobre os elementos nos quais a falha podem levar a perda do requisito. Por exemplo, SR1 pode ser decomposto sobre F2, F3 e F4. Para encontrar uma solução que minimize o custo de alocação e decomposição de ASILs, é utilizada a abordagem apresentada nessa seção.

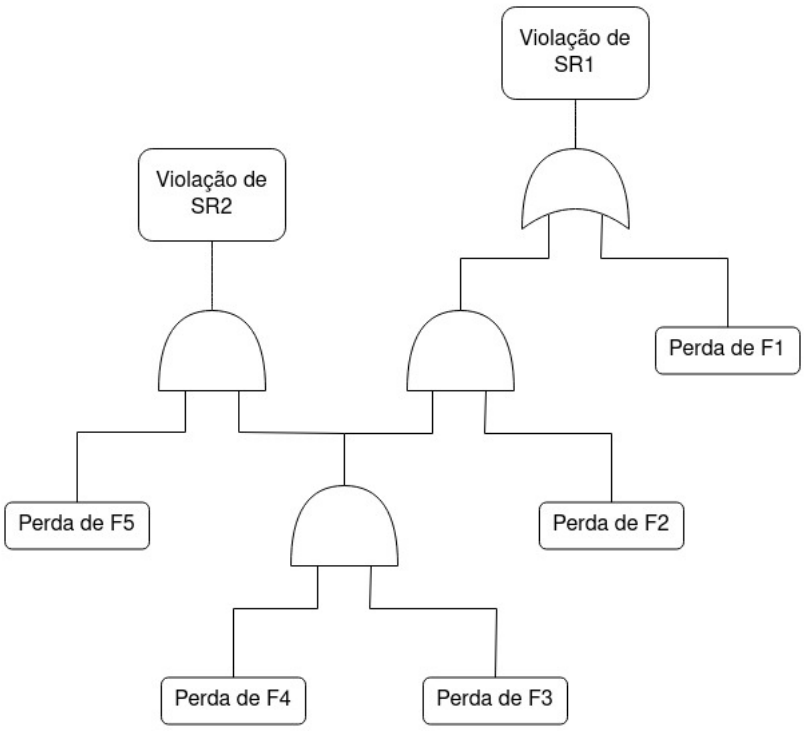

Figura 4.6: FTA do sistema ilustrativo.

Pelo MCS do sistema da Figura 4.6 podemos verificar a seguinte relação entre os ASILs alocados para as FMs e os ASILs dos SR:

$$ASIL(F1) \geq 4 \tag{4.1}$$

$$ASIL(F2) + ASIL(F3) + ASIL(F4) \geq 4 \tag{4.2}$$

$$ASIL(F3) + ASIL(F4) + ASIL(F5) \geq 3 \tag{4.3}$$

A entrada do algoritmo da Busca Tabu é da seguinte forma: N = 5000, $limit_p$ = 4, $limit_q$ = 2, restrictions = MCS da FTA da Figura 4.6 e num_fms = 5. A heurística de custo adotada é a Experimental-I (ASIL QM – custo 0, ASIL A – custo 10, ASIL B –

custo 20, ASIL C – custo 40 e ASIL D – custo 50).

A Figura 4.7 mostra a solução inicial aleatória e viável obtida seguindo a abordagem da subseção 4.4.

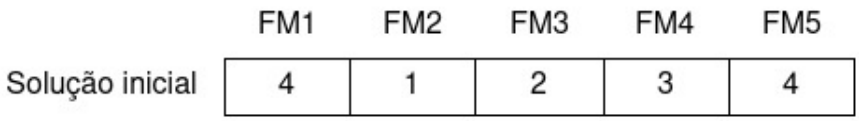

Figura 4.7: Solução inicial do exemplo ilustrativo.

Podemos verificar que a solução encontrada satisfaz as equações (4.1), (4.2) e (4.3) e é de fato viável. O custo da solução é de 170 unidades (50 + 10 + 20 + 40 + 50) e por enquanto é nossa melhor solução. Na primeira iteração, decrementar o ASIL do FM4 representa tomar a descida mais íngreme:

- FM1 e FM5 (ASIL D para ASIL C) — redução de custo: 50 - 40 = 10
- FM2 (ASIL A para ASIL QM) — redução de custo: 10 - 0 = 10
- FM3 (ASIL B para ASIL A) — redução de custo: 20 - 10 = 10
- **FM4 (ASIL C para ASIL B) — redução de custo: 40 - 20 = 20**

Portanto, a solução obtida para a iteração 2 é representada na Figura 4.8.

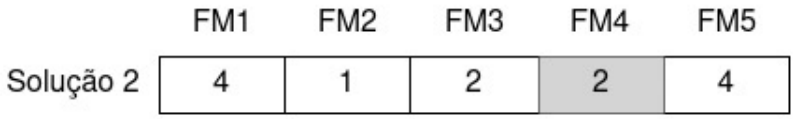

Figura 4.8: Solução para a iteração 2 do exemplo ilustrativo.

A lista tabu é atualizada: FM4 não pode fazer movimentos de subida por 1 iteração. O custo da solução é de 150 unidades (50 + 10 + 20 + 20 + 50) e, portanto, a melhor solução é atualizada. Na iteração 2, todos os movimentos de descida possuem a mesma variação de redução de custo. A escolha do ASIL a ser decrementado será tomada aleatoriamente considerando os movimentos viáveis. Decrementar o ASIL do FM1 irá resultar em uma violação da restrição de alocação (4.1), portanto a escolha fica entre os FMs restantes. Para a iteração 3, escolheu-se aleatoriamente decrementar o ASIL do FM5 para representar tomar a descida mais íngreme:

- FM1 e **FM5 (ASIL D para ASIL C) — redução de custo: 50 - 40 = 10**
- FM2 (ASIL A para ASIL QM) — redução de custo: 10 - 0 = 10
- FM3 e FM4 (ASIL B para ASIL A) — redução de custo: 20 - 10 = 10

Portanto, a solução obtida para a iteração 3 é representada na Figura 4.9.

	FM1	FM2	FM3	FM4	FM5
Solução 3	4	1	2	2	3

Figura 4.9: Solução para a iteração 3 do exemplo ilustrativo.

A lista tabu é atualizada: FM5 não pode fazer movimentos de subida por 1 iteração e a restrição de subida de FM4 foi decrementada, o que a anula. O custo da solução é de 140 unidades (50 + 10 + 20 + 20 + 40) e, portanto, a melhor solução é atualizada. Para a iteração 4, decrementar o ASIL do FM5 representa tomar a descida mais íngreme:

- FM1 (ASIL D para ASIL C) — redução de custo: 50 - 40 = 10
- FM2 (ASIL A para ASIL QM) — redução de custo: 10 - 0 = 10
- FM3 e FM4 (ASIL B para ASIL A) — redução de custo: 20 - 10 = 10
- **FM5 (ASIL C para ASIL B) — redução de custo: 40 - 20 = 20**

Portanto, a solução obtida para a iteração 4 é representada na Figura 4.10.

	FM1	FM2	FM3	FM4	FM5
Solução 4	4	1	2	2	2

Figura 4.10: Solução para a iteração 4 do exemplo ilustrativo.

A lista tabu é atualizada: FM5 não pode fazer movimentos de subida por 1 iteração. O custo da solução é de 120 unidades (50 + 10 + 20 + 20 + 20) e, portanto, a melhor solução é atualizada. Na iteração 4, todos os movimentos de descida possuem a mesma variação de redução de custo. A escolha do ASIL a ser decrementado será tomada aleatoriamente considerando os movimentos viáveis. Decrementar o ASIL do FM1 irá

resultar em uma violação da restrição de alocação (4.1), portanto a escolha fica entre os FMs restantes. Para a iteração 5, escolheu-se aleatoriamente decrementar o ASIL do FM3 para representar tomar a descida mais íngreme:

- FM1 (ASIL D para ASIL C) — redução de custo: 50 - 40 = 10
- FM2 (ASIL A para ASIL QM) — redução de custo: 10 - 0 = 10
- **FM3**, FM4 e FM5 **(ASIL B para ASIL A) — redução de custo: 20 - 10 = 10**

Portanto, a solução obtida para a iteração 5 é representada na Figura 4.11.

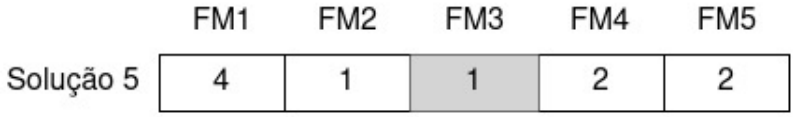

Figura 4.11: Solução para a iteração 5 do exemplo ilustrativo.

A lista tabu é atualizada: FM3 não pode fazer movimentos de subida por 2 iterações e a restrição de subida de FM5 foi decrementada, o que a anula. O custo da solução é de 110 unidades (50 + 10 + 10 + 20 + 20) e, portanto, a melhor solução é atualizada. Na iteração 5, todos os movimentos de descida possuem a mesma variação de redução de custo. A escolha do ASIL a ser decrementado será tomada aleatoriamente considerando os movimentos viáveis. Decrementar o ASIL do FM1 irá resultar em uma violação da restrição de alocação (1), portanto a escolha fica entre os FMs restantes. Para a iteração 6, escolheu-se aleatoriamente decrementar o ASIL do FM5 para representar tomar a descida mais íngreme:

- FM1 (ASIL D para ASIL C) — redução de custo: 50 - 40 = 10
- FM2 e FM3 (ASIL A para ASIL QM) — redução de custo: 10 - 0 = 10
- FM4 e **FM5 (ASIL B para ASIL A) — redução de custo: 20 - 10 = 10**

Portanto, a solução obtida para a iteração 6 é representada na Figura 4.12.

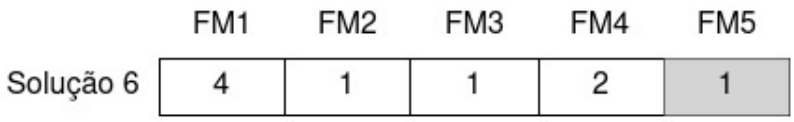

Figura 4.12: Solução para a iteração 6 do exemplo ilustrativo.

A lista tabu é atualizada: FM5 não pode fazer movimentos de subida por 2 iterações e a restrição de subida de FM3 foi decrementada, o que a deixa restringida por 1 iteração. O custo da solução é de 100 unidades (50 + 10 + 10 + 20 + 10) e, portanto, a melhor solução é atualizada. Na iteração 6, todos os movimentos de descida possuem a mesma variação de redução de custo. A escolha do ASIL a ser decrementado será tomada aleatoriamente considerando os movimentos viáveis. Decrementar o ASIL do FM1 irá resultar em uma violação da restrição de alocação (1), portanto a escolha fica entre os FMs restantes. Para a iteração 7, escolheu-se aleatoriamente decrementar o ASIL do FM5 para representar tomar a descida mais íngreme:

- FM1 (ASIL D para ASIL C) — redução de custo: 50 - 40 = 10
- FM2, FM3 e **FM5 (ASIL A para ASIL QM) — redução de custo: 10 - 0 = 10**
- FM4 (ASIL B para ASIL A) — redução de custo: 20 - 10 = 10

Portanto, a solução obtida para a iteração 7 é representada na Figura 4.13.

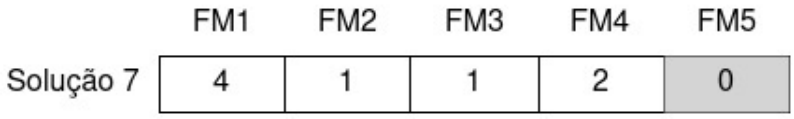

Figura 4.13: Solução para a iteração 7 do exemplo ilustrativo.

A lista tabu é atualizada: FM5 não pode fazer movimentos de subida por 2 iterações e a restrição de subida de FM3 foi decrementada, o que a anula. O custo da solução é de 90 unidades (50 + 10 + 10 + 20 + 0) e, portanto, a melhor solução é atualizada. O algoritmo continuará tentando fazer movimentos até o critério de parada ser atingido, porém, a solução encontrada na iteração 6, ou seja, a solução da Figura 4.13 é a melhor solução encontrada para esse exemplo.

4.6 Teste em um Problema do Mundo Real

Este estudo de caso explora o modelo de um sistema *brake-by-wire* destinado a veículos elétricos, nos quais cada roda é equipada com seu próprio motor elétrico. A frenagem é garantida através de uma abordagem híbrida, que integra a atuação dos motores elétricos com freios eletromecânicos.

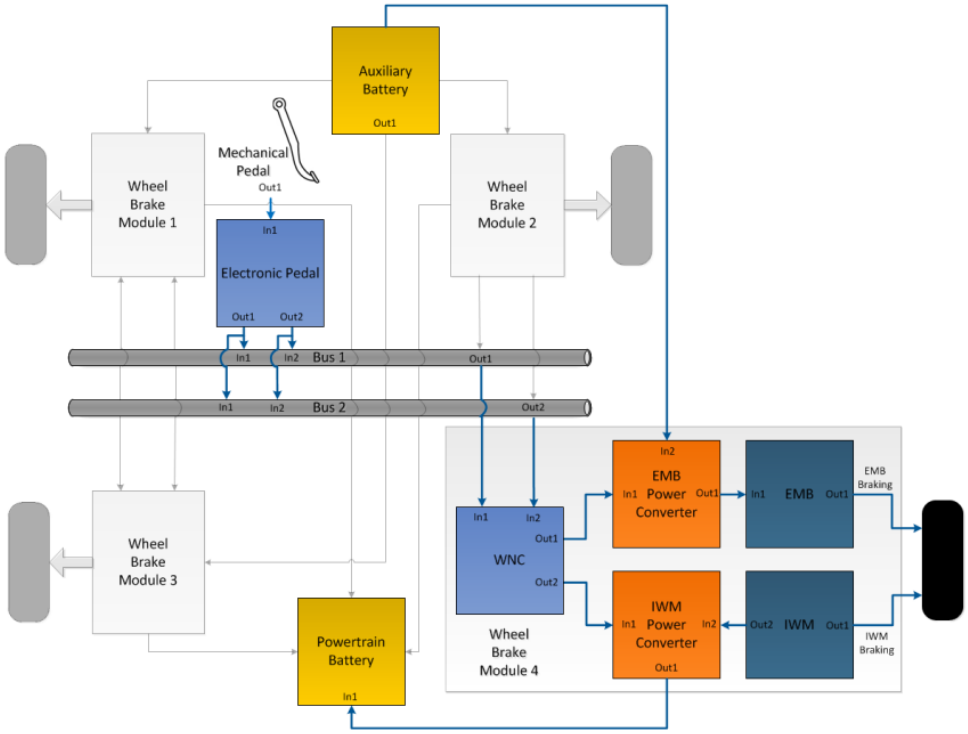

Figura 4.14: Visão geral do HBS.

Foram considerados seis *hazards* para o *hybrid braking system*: *Incorrect Value Braking* (H1) atribuído com ASIL D, *Loss of Braking Rear Wheels* (H2) atribuído com ASIL C, *Loss of Braking Front Wheels* (H3) atribuído com ASIL D, *Loss of Braking Diagonal Wheels* (H4) atribuído com ASIL C, *Loss of Braking 3 out of 4 Wheels* (H5) atribuído com ASIL D e *Loss of Braking All Wheels* (H6) atribuído com ASIL D. Um total de 60 *failure modes* foram identificados, dando um tamanho total de espaço de busca de $5^{60} (\approx 8.67 \times 10^{41})$. Ao aplicar a FTA qualitativa com HiP-HOPS, obtiveram-se informações sobre a propagação de falhas no sistema, expressas na forma de *Minimal Cut Sets* (MCSs). Os MCSs representam a combinação mínima de eventos que resulta em uma falha a nível de sistema; eles podem conter uma única falha que causa diretamente

o risco, ou múltiplas falhas que, em conjunto, o provocam. Para o *hybrid braking system*, foram identificados os seguintes MCSs:

- 1302 MCSs para o H1: 4 pontos de falha única, 2 pontos de falha dupla, 1296 pontos de falha quádrupla;
- 103 MCSs para o H2: 1 ponto de falha única, 3 pontos de falha dupla, 18 pontos de falha tripla e 81 pontos de falha quádrupla;
- 103 MCSs para o H3: 1 ponto de falha única, 3 pontos de falha dupla, 18 pontos de falha tripla e 81 pontos de falha quádrupla;
- 202 MCSs para o H4: 1 ponto de falha única, 3 pontos de falha dupla, 36 pontos de falha tripla e 162 pontos de falha quádrupla;
- 3136 MCSs para o H5: 1 ponto de falha única, 3 pontos de falha dupla, 216 pontos de falha quádrupla e 2916 pontos de falha sêxtupla;
- 6727 MCSs para o H6: 1 ponto de falha única, 3 pontos de falha dupla, 162 pontos de falha quíntupla e 6561 pontos de falha óctupla.

Portanto, o sistema automotivo estudado possui 60 *failure modes* e 11573 *Minimal Cut Sets*. Os testes são feitos tanto com o algoritmo proposto quanto com a ferramenta HiP-HOPS com as seguintes configurações: a heurística de custo adotada é a Experimental-I (ASIL QM – custo 0, ASIL A – custo 10, ASIL B – custo 20, ASIL C – custo 40 e ASIL D – custo 50), 10 execuções, máximo de 5000 iterações sem melhorias e os limites para p e q atribuídos como 2 e 1, respectivamente. Para cada execução uma solução inicial viável foi gerada seguindo a estratégia apresentada na seção 4.1. Todos os testes foram realizados em uma máquina equipada com um processador Intel i3-6006U com clock de 2.00GHz e 8GB de ram.

Os resultados da aplicação do algoritmo de Busca Tabu proposto e da ferramenta HiP-HOPS ao *hybrid braking system* estão apresentados na Tabela 4.1. Para a heurística de custo escolhida, a melhor solução foi identificada previamente utilizando uma técnica exaustiva. “Best” representa o custo da solução ótima previamente identificada; “NBest” indica quantas execuções encontraram o ótimo global; “Dev” mostra o desvio médio quando a solução ótima não foi encontrada; “Iter” refere-se à iteração do

algoritmo onde a melhor solução foi encontrada; “ECPU” é o tempo de execução (em segundos) decorrido quando a melhor solução foi alcançada; “EMCPU” é a média de tempo de execução (em segundos) decorrido quando a melhor solução foi alcançada em todas as 10 execuções; “CPU” é o tempo de execução (em segundos) da melhor solução e “MCPU” é a média de tempo de execução (em segundos) de todas as 10 execuções.

	Best	NBest	Dev	Iter	ECPU	EMCPU	CPU	MCPU
Algoritmo Proposto	830	1	1.45%	179	82.62	84.95	6635.70	4715.73
HiP-HOPS	830	3	0.64%	114	3.44	110.52	154.23	414.38

Tabela 4.1: Resultados dos testes.

Os resultados do algoritmo proposto destacam-se pela capacidade de alcançar a mesma melhor solução (”NBest”) que o HiP-HOPS, evidenciando sua eficácia na obtenção de soluções de alta qualidade. O algoritmo também demonstra um desempenho consistente, com um desvio percentual (”Dev”) de 1,45%, que, embora ligeiramente superior ao do HiP-HOPS, ainda é relativamente baixo, indicando que o algoritmo se aproxima bem da solução ótima. Apesar de o algoritmo proposto exigir mais tempo para encontrar a melhor solução em uma execução específica (”ECPU”), ele apresenta um tempo médio para encontrar a melhor solução nas 10 execuções (”EMCPU”) inferior ao do HiP-HOPS. Essa média reduzida de tempo sugere que o algoritmo proposto possui uma taxa de convergência tão boa quanto ou até melhor que a ferramenta HiP-HOPS.

Apesar de suas boas características, o algoritmo proposto apresenta algumas limitações significativas. O tempo total de execução para encontrar a melhor solução (”CPU”) é substancialmente mais elevado (6635,70) em comparação ao HiP-HOPS (154,23), sugerindo uma eficiência computacional inferior e maior demanda de recursos de processamento. Essa diferença torna-se ainda mais evidente no tempo médio de processamento das 10 execuções (”MCPU”), em que o HiP-HOPS apresenta uma performance aproximadamente dez vezes superior ao algoritmo proposto. Além disso, o HiP-HOPS encontrou a solução ótima em três execuções, enquanto o algoritmo proposto o fez apenas em uma, o que pode indicar que o HiP-HOPS tem uma melhor capacidade de evitar mínimos locais. Essas discrepâncias podem ser atribuídas à necessidade de refatoração e otimização do al-

goritmo proposto, enquanto o HiP-HOPS parece incorporar mecanismos mais sofisticados para lidar com o problema.

Com base nos resultados obtidos, é possível aceitar a hipótese nula de que "HiP-HOPS e o algoritmo proposto não apresentam diferença significativa em relação à qualidade das soluções". Ambos os algoritmos demonstraram a capacidade de alcançar soluções de alta qualidade, próximas do ótimo, mantendo uma convergência e consistência similares. No entanto, a hipótese alternativa, que afirma que "HiP-HOPS possui melhor desempenho", foi confirmada. O tempo total de execução da melhor execução ("CPU") e o tempo médio de execução das 10 execuções ("MCPU") do algoritmo proposto foram significativamente maiores do que os do HiP-HOPS. Isso indica que o algoritmo proposto demanda mais recursos computacionais. Portanto, embora os dois algoritmos sejam comparáveis em termos de qualidade das soluções e taxa de convergência, o HiP-HOPS se mostrou mais eficiente no uso de recursos computacionais.

5 Conclusões

5.1 Contribuições

O presente trabalho desenvolveu uma solução para o problema de alocação e decomposição de ASILs, baseada na meta-heurística da Busca Tabu. O algoritmo proposto foi apresentado tanto em nível teórico quanto prático, incluindo testes realizados em um sistema automotivo real. Os resultados demonstraram que o algoritmo é capaz de gerar soluções que minimizam os custos ao mesmo tempo que garantem os requisitos mínimos de segurança, alcançando resultados equivalentes aos da ferramenta comercial HiP-HOPS. No entanto, em termos de execução e processamento, o HiP-HOPS apresentou desempenho superior ao do algoritmo proposto.

5.2 Benefícios

A alocação e decomposição de ASILs é um problema combinatorial cuja solução manual se mostra impraticável, tornando essencial o conhecimento e a aplicação de ferramentas e métodos meta-heurísticos e exatos para resolver o problema de forma automática.

Métodos baseados em populações, que avaliam múltiplas soluções ao longo de várias iterações, podem exigir um esforço computacional significativo, especialmente para problemas complexos. Em contraste, a implementação da Busca Tabu desenvolvida neste trabalho é uma meta-heurística mais simplificada, que se baseia em uma única solução ajustada a cada iteração por meio de movimentos que exploram soluções vizinhas.

A combinação de uma estratégia de busca local com mecanismos de memória, que evitam a permanência em ótimos locais, demonstrou resultados notáveis. O algoritmo proposto mostrou-se eficaz na obtenção de soluções de alta qualidade, alcançando resultados comparáveis ao HiP-HOPS.

Embora exija mais tempo de execução, o algoritmo demonstra consistência e capacidade de convergir de forma eficiente, sugerindo uma performance competitiva na

obtenção de soluções que minimizam os custos de alocação, ao mesmo tempo que atendem aos requisitos mínimos de segurança.

5.3 Limitações

Embora o algoritmo apresente resultados positivos, ele possui limitações importantes relacionadas à sua eficiência. O tempo de execução significativamente maior e a menor frequência com que encontra a melhor solução indicam uma demanda maior por recursos e uma menor capacidade de evitar armadilhas durante o processamento. Essas limitações podem ser atribuídas à necessidade de melhorias no algoritmo, enquanto o método da ferramenta comercial HiP-HOPS parece utilizar abordagens mais avançadas para lidar com o problema de forma eficiente.

5.4 Trabalhos futuros

Com base nos estudos realizados, os próximos objetivos incluem o desenvolvimento de um algoritmo utilizando o modelo matemático da Suavização Hiperbólica. Será necessário, também, realizar uma refatoração e otimização mais detalhada do algoritmo desenvolvido neste trabalho para avaliar de maneira mais satisfatória seu desempenho e precisão em instâncias reais do problema, comparando-o com a ferramenta HiP-HOPS. Além disso, planeja-se uma análise comparativa entre a Suavização Hiperbólica e a Busca Tabu, considerando a possível integração da Suavização Hiperbólica à ferramenta HiP-HOPS e sua inclusão em revisões do estado da arte sobre alocação e decomposição de ASILs.

Bibliografia

ABELE, A.; LIBRINO, R. Model-based analysis & engineering of novel architectures for dependable electric vehicles report type deliverable d 3 . 2 . 1 report name analysis and synthesis concepts supporting engineering scenarios. In: . [S.l.: s.n.], 2012.

AVIZIENIS, A.; LAPRIE, J.-C.; RANDELL, B.; LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, Institute of Electrical and Electronics Engineers (IEEE), v. 1, n. 1, p. 11–33, jan 2004.

AZEVEDO, L. da S. *Scalable Allocation of Safety Integrity Levels in Automotive Systems*. University of Hull, 2015. Disponível em: ⟨https://books.google.com.br/books?id=18LevgEACAAJ⟩.

AZEVEDO, L. S.; PARKER, D.; WALKER, M.; PAPADOPOULOS, Y.; ARAÙJO, R. E. Automatic Decomposition of Safety Integrity Levels: Optimization by Tabu Search. In: ROY, M. (Ed.). *SAFECOMP 2013 - Workshop CARS (2nd Workshop on Critical Automotive applications : Robustness & Safety) of the 32nd International Conference on Computer Safety, Reliability and Security*. Toulouse, France: [s.n.], 2013. p. NA. Disponível em: ⟨https://hal.archives-ouvertes.fr/hal-00848213⟩.

COIT, D. W.; SMITH, A. E. Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE trans. reliab.*, Institute of Electrical and Electronics Engineers (IEEE), v. 45, n. 2, p. 254–260, 266, jun. 1996.

DHOUIBI, M. S.; SAINTIS, L.; BARREAU, M.; PERQUIS, J.-M. Automatic Decomposition and Allocation of Safety Integrity Level Using System of Linear Equations. In: *PESARO 2014, The Fourth International Conference on Performance, Safety and Robustness in Complex Systems and Applications*. Nice, France: IARIA, 2014. p. 1–5. Disponível em: ⟨https://hal.univ-angers.fr/hal-03287182⟩.

GENDREAU, M.; POTVIN, J.-Y. Tabu search. In: _____. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Boston, MA: Springer US, 2005. p. 165–186. ISBN 978-0-387-28356-2. Disponível em: ⟨https://doi.org/10.1007/0-387-28356-0_6⟩.

GHERAIBIA, Y.; DJAFRI, K.; KRIMOU, H. Ant colony algorithm for automotive safety integrity level allocation. *Appl. Intell.*, Springer Nature, v. 48, n. 3, p. 555–569, mar. 2018.

GHERAIBIA, Y.; KABIR, S.; DJAFRI, K.; KRIMOU, H. An overview of the approaches for automotive safety integrity levels allocation. *Journal of Failure Analysis and Prevention*, Springer Science and Business Media LLC, v. 18, n. 3, p. 707–720, apr 2018.

GHERAIBIA, Y.; MOUSSAOUI, A.; AZEVEDO, L. S.; PARKER, D.; PAPADOPOULOS, Y.; WALKER, M. Can aquatic flightless birds allocate automotive safety requirements? In: *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*. [S.l.]: IEEE, 2015.

HANSEN, P.; JAUMARD, B. Algorithms for the maximum satisfiability problem. *Computing*, Springer Science and Business Media LLC, v. 44, n. 4, p. 279–303, dec 1990.

HANSEN, P.; LIH, K.-W. Heuristic reliability optimization by tabu search. *Ann. Oper. Res.*, Springer Science and Business Media LLC, v. 63, n. 2, p. 321–336, abr. 1996.

ISO. *Road vehicles – Functional safety*. [S.l.]: ISO, Geneva, Switzerland, 2011.

KITCHENHAM, B. Procedures for performing systematic reviews. *Keele, UK, Keele Univ.*, v. 33, 08 2004.

KNIGHT, J. C. Safety critical systems. In: *Proceedings of the 24th international conference on Software engineering - ICSE '02*. [S.l.]: ACM Press, 2002.

MADER, R.; ARMENGAUD, E.; LEITNER, A.; STEGER, C. Automatic and optimal allocation of safety integrity levels. In: *2012 Proceedings Annual Reliability and Maintainability Symposium*. [S.l.: s.n.], 2012. p. 1–6.

MöSSINGER, J. Software in automotive systems. *IEEE Software*, Institute of Electrical and Electronics Engineers (IEEE), v. 27, n. 2, p. 92–94, mar 2010.

PAPADOPOULOS, Y.; HULL, U. of. *HiP-HOPS: Automated Fault Tree, FMEA and Optimisation Tool – User Manual*. 2013. Disponível em: ⟨https://hip-hops.co.uk/manual/HiP-HOPS_Manual.pdf⟩.

PARKER, D.; WALKER, M.; AZEVEDO, L. S.; PAPADOPOULOS, Y.; ARAÚJO, R. E. Automatic decomposition and allocation of safety integrity levels using a penalty-based genetic algorithm. In: *Recent Trends in Applied Artificial Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, (Lecture notes in computer science). p. 449–459.