UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Utilização de Algorítmos Genéticos na Otimização do Escalonamento de Mensagens Proprietárias do Protocolo SAE J1939 sobre CAN bus.

Marcio Folly de Campos

Itajubá, Maio de 2010

Ficha catalográfica elaborada pela Biblioteca Mauá
Bibliotecária Margareth Ribeiro - CRB_6/1700

C198u
Campos, Márcio Folly de
Utilização de algoritmos genéticos na otimização do escalona_
mento de mensagens proprietárias do Protocolo SAE J1939 sobre
CAN bus / Márcio Folly de Campos. -- Itajubá, (MG) : [s.n.], 2010.
123 p. : il.
Orientador: Profa. Dra. Lúcia Regina Horta Rodrigues Franco.
Dissertação (Mestrado) – Universidade Federal de Itajubá.
1. Escalonamento. 2. Rede CAN. 3. SAE J1939. 4. Algoritmo
genético. I. Franco, Lúcia Regina Horta Rodrigues, orient. II. Universidade Federal de Itajubá.
III. Título

UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

Marcio Folly de Campos

Utilização de Algorítmos Genéticos na Otimização do Escalonamento de Mensagens Proprietárias do Protocolo SAE J1939 sobre CAN bus.

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para a obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Área de Concentração: Automação e Sistemas Elétricos Industriais

Orientador(a): Profa. Dra. Lucia Regina Horta Rodrigues Franco

Maio de 2010

Itajubá - MG

Resumo

A competição entre processos pelos recursos de um processador é muito explorada na ciência da computação, gerando um grande número de algoritmos de escalonamento. A mesma competição ocorre em uma rede embarcada onde as mensagens competem pelo acesso ao barramento.

Este trabalho mostra uma visão sobre as métricas temporais de uma mensagem na rede *Controller Area Network*, CAN, e como o escalonamento de mensagens está relacionado com as técnicas de cálculo temporal aplicadas ao protocolo J1939, publicado pela Sociedade de Engenharia Automotiva, *Society of Automotive Engineering*, SAE. Para tal, foram estudados os modelos matemáticos propostos por diferentes trabalhos. A aplicação destes modelos é feita em um conjunto de mensagens SAE J1939 sobre CAN Bus e os resultados são analisados.

O resultado das análises temporais indicam que em determinados casos a otimização é necessária. A aplicação de algoritmos genéticos é estudada, neste trabalho, com a finalidade de otimização do escalonamento de mensagens SAE J1939. Para isto, é proposta uma correlação entre a teoria de algoritmos genéticos e as características do protocolo, fazendo com que seja possível representar o problema pelo método de otimização escolhido. Este trabalho apresenta também a implementação do algoritmo genético para solucionar o problema detectado na análise temporal. Os resultados das simulações são apresentados e analisados.

Abstract

The competition between processes for a processor resources is hardly explored in computer science, generating a mount of scheduling algorithms with it own features and applications. The same competition occurs in an embedded Controller Area Network, CAN, for a bus access.

This research shows an overview about message temporal metrics over CAN, and how the message scheduling is related to the temporal calculation techniques applied in the J1939 protocol, published by Society of Automotive Engineering, SAE. Thus, the mathematic models from different researches were studied. The application of the related models is done in the SAE J1939 protocol set of messages and the results are analysed.

The results of the temporal analysis indicate that the optimization is often required. The application of Genetic Algorithms on this case, for optimization purposes, is presented. A correlation between the genetic algorithms theory and the protocol characteristics is proposed allowing the problem modelling through the chosen optimization method. The implementation of the genetic algorithm to solve the detected problem during the temporal analysis is presented. The simulation results are presented and analyzed.

Sumário

Lista de Tabelas

Lista de Figuras

Tabela de Abreviaturas

CAN – Controller Area Network

CRC – Cyclic Redundant Check

DLC – Data Length Code

DM – Diagnostic Message

DP – Data Page

DTC – Diagnostic Trouble Code

EDF – Earliest Deadline First

FMI – Failure Mode Identifier

GA – Genetic Algorithm

IMD – Inconsistent Message Duplicate

IMO – Inconsistent Message Omission

ISO – International Standard Organization

LIN – Local Interconnect Network

LLC – Logic Link Control

MAC – Medium Access Control

MAU – Medium Access Unit

MDI – Medium Dependent Interface

NRZ – Non Return to Zero

OC – Occurrence Count

OSI – Open System Interconnect

PDU – Protocol Data unit

PE – PDU Specific

PF – PDU Format

PGN – Parameter Group Number

PLS – Physical Layer Signaling

PMA – Physical Medium Attachment

RTR – Remote Transmit Request

SAE – Society of Automotive Engineering

SPN – Suspect Parameter Number

SRR – Substitute Remote Request

VBA – Visual Basic for Applications

WCRT – Worst Case Response Time

Agradecimentos

Agradeço à minha orientadora Profa. Dra. Lucia Franco, aos professores, pelos quais pude obter conhecimentos valiosos para o desenvolvimento do trabalho, aos colegas de curso, cuja troca de conhecimentos contribuiu de forma decisiva, à minha família, e, à minha esposa Mariana que me suportaram durante esta jornada.

1. INTRODUÇÃO

1.1. Objetivos.

O primeiro objetivo deste trabalho é analisar a rede CAN e o protocolo de comunicação de dados SAE J1939 no que diz respeito ao seu comportamento temporal de forma a identificar casos de atrasos no tempo de resposta, ou latência, de uma ou mais mensagens gerando falta da informação da parte de quem espera recebê-la. Para tal, uma série de teorias relacionadas ao comportamento temporal de redes CAN são analisadas de forma a encontrar o modelo mais adequado ao caso estudado, o protocolo SAE J1939. O segundo objetivo do estudo é definir o melhor modelo para simular o comportamento temporal do barramento sob a ação de erros, seguindo as análises anteriormente descritas. Simular significa implementar em software os modelos matemáticos estudados e calcular de tempo de resposta de um dado conjunto de mensagens. E como objetivo final deste trabalho, propor a implementação de algorítmos genéticos na otimização do escalonamento de um conjunto de mensagens SAE J1939 sobre CAN, dada a condição de falha identificada pela análise temporal, descrita dos dois primeiros objetivos.

1.2. Motivação.

A necessidade crescente de agregar tecnologia a baixo custo exige que as redes embarcadas aceitem cada vez mais informação, sejam cada vez mais rápidas, tenham cada vez mais dispositivos interligados, sem perder sua confiabilidade. Para tal se faz necessário conhecer as características e os limites de uma rede.

Uma série de trabalhos foi publicada com o intuito de modelar o comportamento de uma rede CAN, considerando aspectos temporais e características técnicas deste tipo de rede. Esta teoria vem sendo aperfeiçoada com o tempo e uma série de modelos surgiram. O objeto de estudo, o protocolo SAE J1939, tem características que são levadas em consideração no estudo e

aplicação das teorias de análise de redes CAN até então publicadas. Estas caraacterísticas fazem com que o modelo possa ser adaptado a este protocolo.

Um grande desafio na concepção de um projeto de rede embarcada automotiva é a correta definição e distribuição das mensagens pelos módulos eletrônicos e pelas redes embarcadas que os interligam. Em uma situação onde é necessário aumentar a quantidade de mensagens de um barramento, é importante determinar os limites da rede, em termos de uso do barramento.

1.3. Organização da Dissertação.

Esta dissertação está distribuída da seguinte forma: o capítulo 2 discute o conceito de arquitetura de controle centralizado e distribuído, as vantagens e desvantagens de cada um. Discute também as redes de comunicação de dados, suas características, os tipos de redes existentes e as topologias. O capítulo 3 discute os sistemas de tempo-real, os algorítmos de escalonamento e suas aplicações. O capítulo 4 apresenta a rede CAN, que é o objeto de estudo desta dissertação, e suas características. O capítulo 5 apresenta o Protocolo SAE J1939 e suas características. O capítulo 6 apresenta as teorias propostas nas últimas décadas sobre análise temporal aplicada à rede CAN, de forma a representar teoricamente seu comportamento. No capítulo 7 o protocolo SAE J1939 é analisado em relação às teorias apresentadas no capítulo 6. O capítulo 8 apresenta a teoria de algorítmos genéticos aplicada na solução do problema apresentado neste trabalho. O capítulo 9 trata a criação da ferramenta de simulação do comportamento de uma rede CAN com SAE J1939 baseado nos modelos determinados no capítulo anterior. Neste capítulo são apresentadas as análises dos resultados da simulação com o comportamento do barramento CAN. As conclusões do trabalho são apresentadas no capítulo 10, bem como os próximos passos deste desenvolvimento.

2. REVISÃO BIBLIOGRÁFICA

2.1. Sistemas Distribuídos

O avanço da eletrônica no meio industrial significou, na prática, uma utilização cada vez mais crescente de circuitos eletrônicos com, consequentemente, mais capacidade de controle, mais funções e interações com o ambiente o qual é inserido. Em conjunto ao aumento da responsabilidade dos circuitos eletrônicos está o aumento de sinais eletrônicos lidos ou processados por estes circuitos. Surge, então, os sistemas de controle centralizados, onde todo o processamento é centralizado em um controlador e ocorre sobre parâmetros de controle e sinais de entrada provenientes de uma gama de sensores. A figura 1 representa o conceito de sistema centralizado. De acordo com [32] os principais benefícios são disponibilidade exclusiva dos dados de entrada e a simplicidade das malhas de controle. As principais desvantagens são: grande quantidade de cabeamento necessário para transportar os dados dos sensores da entrada e dos atuadores da saída, e também, a capacidade de expansão limitada ao hardware do controlador eletrônico.

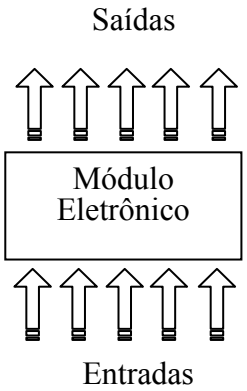

Figura 1 - Sistema Centralizado

Este modelo foi sendo substituído pelo modelo distribuído de controle, onde a responsabilidade do processamento é dividida entre os participantes de uma malha

de controle. Um sensor que antes transformava a variação de uma grandeza física para sinal eletrônico e enviava a um controle central, agora processa este sinal antes de enviar a seu cliente, que pode ser um módulo de controle eletrônico, um atuador, ou até um outro sensor. A figura 2 apresenta a arquitetura de um sistema distribuído.

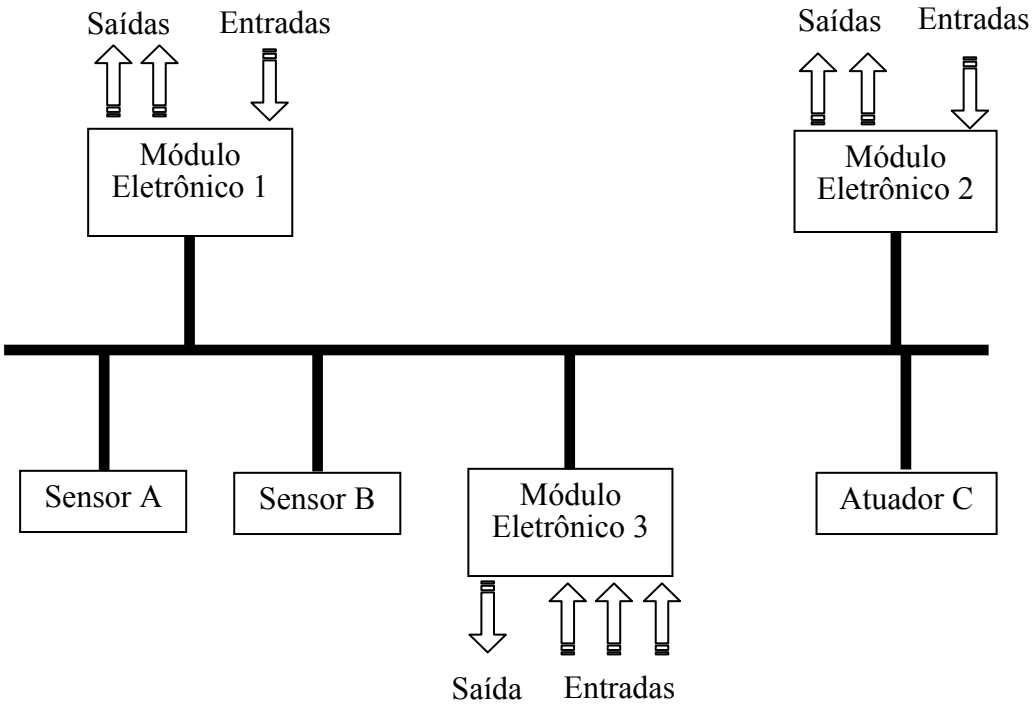

Figura 2 - Sistema Distribuído.

De acordo com [32] as principais vantagens de um sistema de controle distribuído são redução do cabeamento, distribuição da responsabilidade do controle, capacidade de modularização. Baseado no princípio de que o controle é distribuído e de que uma informação gerada por um dos dispositivos pode ser útil a uma ou mais malhas de controle, a garantia das condições acima citadas se dá em um sistema de controle distribuído quando a comunicação entre os dispositivos obedece a uma regra comum, o protocolo de comunicação. Suas desvantagens são o aumento da complexidade das malhas de controle e a necessidade da definição de um protocolo de comunicação comum. A figura 2 apresenta um grupo de dispositivos interligados por um meio comum. A este meio

comum de comunicação entre os dispositivos dá-se o nome de Rede de Comunicação de Dados.

2.2. Redes de comunicação de dados.

Rede de Comunicação de dados é o meio comum de comunicação entre dispositivos capazes de se comunicar eletronicamente. Uma rede de comunicação de dados tem uma série de características que determinam sua forma de funcionamento e seu campo de aplicação, como por exemplo, a taxa de transmissão, o meio físico, o tipo de transmissão, e também a codificação dos dados. Os sistemas automotivos absorveram os avanços tecnológicos vindos da eletrônica e da computação na medida em que as indústrias do ramo passaram a implementar controles eletrônicos nos automóveis. No decorrer da evolução, os controles de um automóvel se tornaram complexos, distribuídos e interligados por meio de redes de comunicação de dados embarcadas.

As informações trocadas entre os dispositivos conectados são formatadas em mensagens. Cada mensagem tem uma certa quantidade de bits que estão divididos em dois campos principais: *overhead* , que significa os dados de controle da comunicação, e *payload*, que significa a quantidade de informação que efetivamente se está transmitindo. Cada protocolo de comunicação tem uma formatação característica a sua aplicação.

2.3. Topologias de Rede.

As diversas formas de interligação entre os dispositivos conectados via rede de comunicação de dados são chamadas de topologias de rede. A figura 3, 4 e 5 apresentam as formas mais comuns de topologia de rede [43].

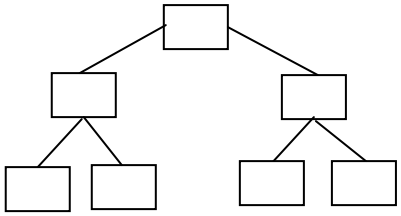

Figura 4 - Topologia em Árvore [43].

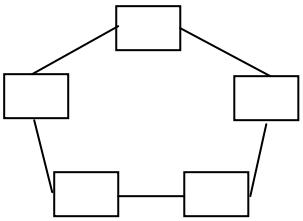

Figura 3 -Topologia em Anel [43].

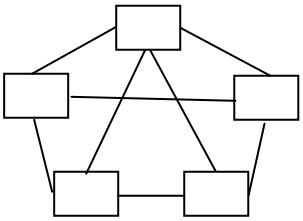

Figura 5 - Topologia Mista [43].

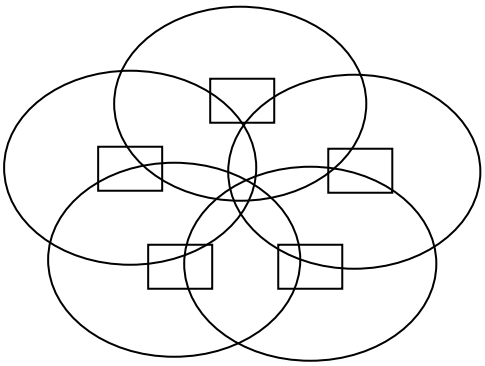

Figura 6 - Topologia Mesh [43].

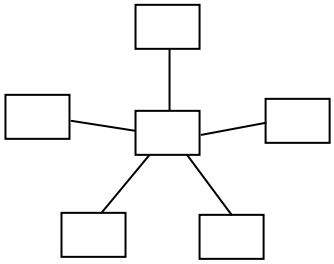

Figura 8 - Topologia em Estrela [43].

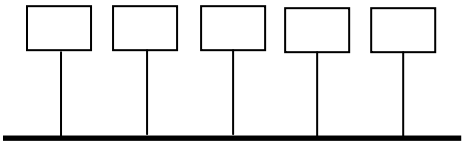

Figura 7 - Topologia em Barramento [43].

Não é objeto de estudo deste trabalho o aprofundamento nas características e aplicações de todas as topologias, porém as figuras 7 e 8 apresentam as topologias mais utilizadas na indústria automobilística. A topologia em estrela, onde um dos módulos eletrônicos faz o papel de mestre é utilizado em redes de baixa velocidades, da ordem de dezenas de kilobits por segundo, como exemplo, a rede LIN [33]. A topologia em barramento é a mais utilizada em sistemas automotivos, por sua flexibilidade, modularidade, além de se poder implementar uma rede multi-mestre, não dependente de um só módulo eletrônico para funcionar.

2.4. Sistemas de Tempo-Real.

Sistemas de tempo real são aqueles onde um determinado *deadline* precisa ser alcançado. O *deadline* de uma mensagem ou processo é o instante de tempo máximo em que uma dada mensagem ou processo precisa chegar ao seu destino ou ser finalizado. De acordo com [30] o *deadline* é o ponto máximo no tempo, ou seja, o limite, na linha do tempo, onde um dado evento deve ocorrer.

Os sistemas de tempo real podem ser divididos em *soft real-time* e *hard real-time*. Em [10] um sistema *hard real-time* é aquele em que é mandatório o antendimento aos requisitos temporais, de outra forma, a segurança do sistema estará comprometida. Em [30], um sistema onde existam processos com *deadlines* críticos, chamados *hard-deadlines*, são ditos como sistemas *hard-real-time*. Um sistema *soft real-time* é aquele que pode conviver com algum atraso no atendimento aos deadlines, para mensagens ou processos não críticos, sem comprometer a segurança do sistema.

2.5. Algorítmos de Escalonamento.

A forma ordenada com que os processos acessam um recurso comum é determinada pelo tipo de escalonamento que se é aplicado. Em [10] os algorítmos *rate monotonic* e *deadline monotonic* são apresentados para aplicação em sistemas hard real-time. *Rate monotonic* é a denominação do algorítmo de escalonamento que utiliza as taxas de repetição, ou período, de cada processo para determinar sua prioridade. *Deadline monotonic* é a denominação do algorítmo de escalonamento onde os deadlines de cada processo são usados como referência para determinar sua prioridade. Outros algorítmos também são utilizados como o dinâmico *earlier deadline first* , onde dinamicamente a tabela de prioridades é atualizada dependendo de quem está mais próximo de seu deadline. Em [15] a comparação entre *deadline monotonic* e *earliest deadline first* foi apresentada, bem como, uma solução mista em aplicação à rede CAN, que será comentada no próximo capítulo.

O objetivo deste trabalho é a analise do protocolo SAE J1939, que foi desenvolvido para CAN em veículos comerciais. Como será visto em detalhes posteriormente, o acesso ao barramento, no caso do protocolo SAE J1939, é determinado pela prioridade de cada mensagem [5]. Esta prioridade é previamente definida pelo valor numérico de seu identificador. Além desta característica, a rede CAN é não-preemptiva, ou seja, uma mensagem não tem sua transmissão interrompida por uma outra mensagem, de maior prioridade, que tenha se tornado apta à transmissão. Esta característica será comentada posteriormente. Portanto a política de escalonamento do protocolo SAE J1939, sobre uma rede CAN, é chamada de *Fixed Priority Non-Preemptive Scheduling*, ou, escalonamento não preemptivo de prioridade fixa [31].

2.6. A rede CAN.

2.6.1. Histórico.

A introdução de uma rede de comunicação de dados entre módulos eletrônicos automotivos teve seu início nos anos 80 com a criação da CAN pela Robert Bosch Gmbh. A motivação foi de otimizar a comunicação entre suas centrais eletrônicas pela multiplexagem de informação no barramento, consequentemente reduzindo cabeamento entre os dispositivos interligados. O trabalho foi publicado primeiramente como CAN Bus 1.0a onde é definida uma rede de baixa velocidade para aplicações não críticas. Posteriormente duas especificações foram publicadas, [1] definindo redes de velocidades mais altas para aplicações de controle e críticas. Este protocolo foi bem aceito pela indústria automotiva devido à sua representatividade em relação às funções e parâmetros sobre os quais pode se aplicar o conceito de controle distribuído.

Nos anos 90 a International Standard Organisation publicou a norma ISO 11898 [2], baseada na documentação Bosch [1], abrindo as portas do conceito CAN para que se tornasse o padrão a ser implantado em aplicações automotivas.

Em [1] a CAN é definida como um protocolo de comunicação serial que suporta eficientemente controles distribuídos em tempo real com um alto nível de segurança.

2.6.2. Modelo OSI e o paralelo CAN.

A ISO definiu, como forma de padrão de comunicação entre dispositivos via redes de comunicação de dados, um modelo, ISO 7498, onde esta comunicação é distribuída em camadas, cada qual com suas atribuições específicas, e com um objetivo comum de organizar priorizar e controlar o envio e recebimento de informações.

A tabela 1 representa o modelo OSI para comunicação via rede.

MODELO OSI	
CAMADAS	***FUNÇÃO***
Aplicação	Interface com o usuário
Apresentação	Formato dos dados e criptografia
Seção	Link remoto
Transporte	Tratamento de erros
Rede	Endereçamento
Enlace	Endereço imediato e forma de transmissão
Física	Características do meio físico

Tabela 1 – Modelo OSI

As informações de controle geradas nas respectivas camadas são acrescentadas aos dados a serem transmitidos em um processo de encapsulamento.

A figura 9 mostra o fluxo de uma mensagem desde a criação do dado que se quer transmitir até a entrega ao cliente final, em um Gerenciador de Emails, por exemplo.

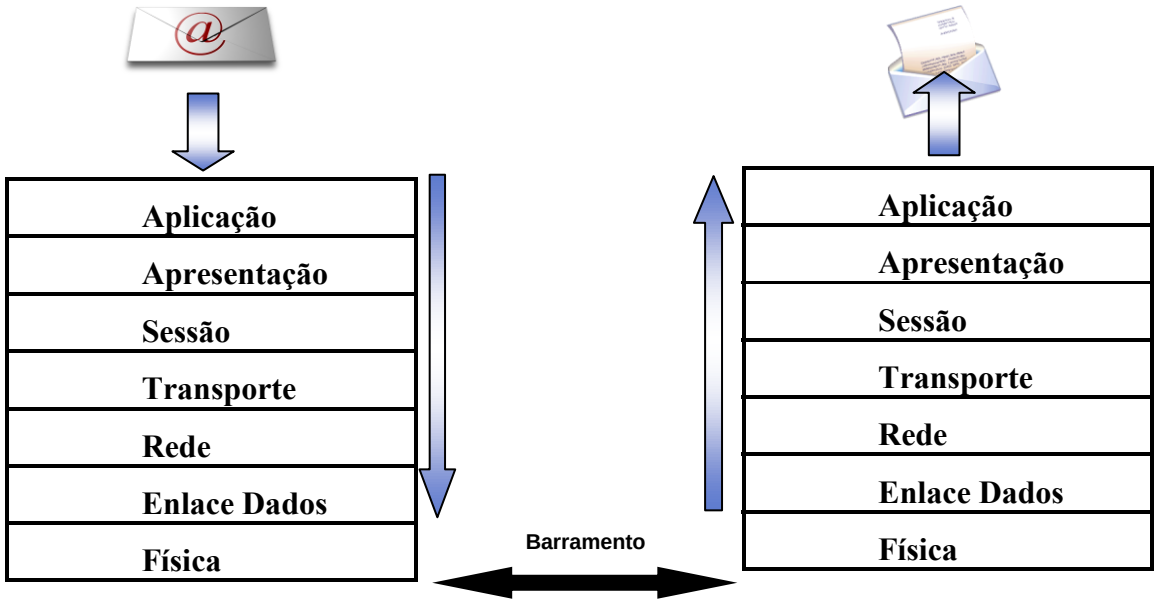

Figura 9 - Fluxo da mensagem

Esta forma de preparação da mensagem foi modificada e adaptada para as aplicações necessárias. A norma ISO 11898 define as camadas física e de enlace de dados , em relação ao modelo ISO, para uma rede CAN de alta velocidade, 125kbps a 1Mbps. As outras camadas do modelo OSI são definidas de acordo com os diferentes protocolos que foram especificados de acordo coma aplicação a que se destina uma rede CAN [32]. O protocolo SAE J1939 define as características da rede de comunicação em relação às camadas intermediárias do modelo OSI, como será apresentado na seção 2.7. A rede CAN foi criada com uma estrutura mais simples onde a mensagem tem menor *overhead* e campo de dados maior. Desta forma o encapsulamento de uma mensagem CAN é otimizado e contém apenas as informações necessárias ao controle da comunicação, além dos dados fisicos. A figura 10 apresenta as camadas de uma rede CAN em comparação com o modelo OSI.

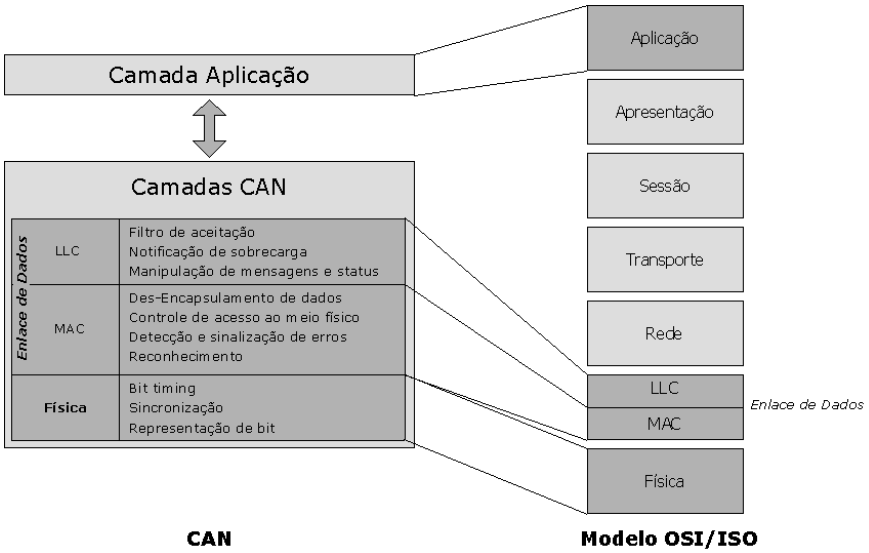

Figura 10 - Camadas de rede CAN x OSI [43]

2.6.3. Camada de Enlace de Dados da rede CAN.

A camada de enlace de dados da rede CAN é dividida em LLC e MAC.

2.6.3.1. LLC – *Logic Link Control.*

A sub-camada LLC é a mais alta da camada de enlace de dados. É responsável pelo filtro de aceitação de mensagens, notificação de overload, gerenciamento da recuperação de mensagens na ocorrência de um erro, e por executar o link entre a camada de aplicação e a sub-camada MAC através de primitivas específicas [2].

Como exemplo, os softwares da camada de aplicação enviam à sub-camada LLC uma primitiva, *L_DATA.request*, contendo as informações de identificador,

tamanho do campo de dados e os dados. A sub-camada LLC ordena estas informações de forma a poder transmití-las à sub-camada MAC.

O frame LLC contém três campos, mostrados na figura 11.

Figura 11 - Frame LLC

Onde:

Identifier field: 11 bits ou 29 bits para frames estendidos. É o identificador da mensagem e detentor de sua prioridade.

DLC field: 4 bits. *Data length code*, é o tamanho do campo de dados, em bytes.

LLC data field: 0 a 64 bits, é o conjunto de dados que se quer transmitir.

Em um processo de recepção a sub-camada LLC recebe da sub-camada MAC as informações acima descritas. O identificador é verificado e, no caso da necessidade de se receber esta mensagem, o frame LLC é enviado à camadas de aplicação via primitiva *L_DATA.indication*.

2.6.3.2. MAC – *Medium Access Control.*

É responsável pelo encapsulamento e desencapsulamento da mensagem, gerenciamento de erros, do acesso ao barramento, além da serialização/deserialização do frame para transmissão/recepção. É a sub-camada mais baixa da camada de enlace de dados.

Em [2] o passo a passo para transmissão e recepção via MAC sublayer é apresentado.

A figura 12 representa o frame MAC.

Figura 12 - Frame MAC

Onde:

SOF – Start of Frame – Bit dominante que determina o início de uma mensagem.

Arbitration field – 11 bits para *standard* frames e 29 bits para *extended* frames. Contém parte das informações vindas do LLC e é o campo que determina a prioridade e consequentemente o acesso ao barramento.

Control field – 4 bits que indicam a quantidade de bytes do campo de dados. Chamado de DLC (*data length code*).

Data field – campo de dados , que pode conter até 64 bits.

CRC field – Cyclic Redundancy Check. 16 bits. Uma das ferramentas de detecção de erros.

ACK field – 2 bits. Campo destinado à confirmação do recebimento de uma mensagem por parte dos nós.

EOF – End of Frame – 6 bits recessivos consecutivos determinando o fim da mensagem.

Por meio de primitivas a sub-camada MAC se comunica com a sub-camada LLC para que seja possível a construção de um frame CAN.

O exemplo iniciado na sub-camada LLC pode ser utilizado na sequência: a sub-camada LLC envia à MAC, via primitiva *MA_DATA.request*, o identificador da mensagem , o DLC (*data length code*) e os dados a serem transmistidos. A camada MAC monta o frame inserindo o identificador no arbiration field juntamente com bits reservados, o DLC no *Control field* e os dados no campo de dados. Além destas informações são inseridas também o SOF o CRC o ACK e o EOF, descritos anteriormente. Finalmente ocorre o processo de serialização, inserção de stuffing bits e transmissão.

Na recepção de uma mensagem, o frame é recebido pela sub-camada MAC e as informações pertinentes (frame LLC) são enviadas à sub-camada LLC via primitiva *MA_DATA.indication*.

2.6.4. Camada física da rede CAN.

De acordo com [1] a camada física é responsável pela codificação e decodificação de bit, geração, envio e recebimento do bit no tempo compatível com a velocidade da rede, e pela sincronização do bit.

Na camada física o trem de bits, ou frame, é codificado de acordo com o método NRZ(*non return to zero*). Isto significa que tanto o nível lógico "1" quanto o "0" são transmitidos em níveis de tensão diferentes de zero, e que , em toda a transmissão de um bit seu nível de tensão permanece constante. Em [2] o nível lógico 1 é chamado de "*recessivo*" e o nível lógico "0" de "*dominante*".

A transmissão é feita em dois canais, CAN-High e CAN-Low, e a recuperação da informação pelos receptores se dá pela diferença entre os canais. A figura 13 mostra a representação física do bit.

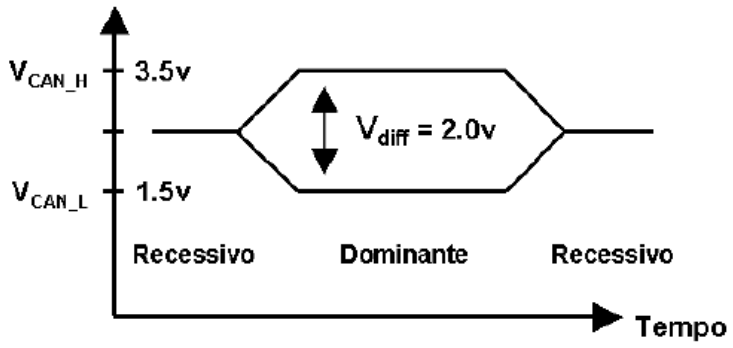

Figura 13 - representação física do bit [32].

O nível de tensão de um bit recessivo é de 2,5[V] para ambos os canais. Os níveis de tensão de um bit dominante são de 3,5[V] no CAN-High e 1,5[V] no CAN-Low.

É importante comentar que um bit dominante sobrescreve um bit recessivo quando da transmissão simultânea de dois bits, um recessivo e outro dominante. Esta característica permite que o acesso ao barramento seja resolvido pela comparação dos sinais de transmissão e recepção de cada nó. Quando um nó transmite um bit recessivo e recebe um bit dominante, significa que existe um outro nó transmitindo com uma prioridade maior. Imediatamente o nó que enviou o bit recessivo interrompe a transmissão e segue apenas recebendo dados do barramento. Este mecanismo permite que o identificador de uma mensagem seja portador de sua prioridade. Na próxima seção o frame CAN será discutido e a questão do acesso ao barramento será comentada novamente.

De uma forma geral o tempo de bit obedece a seguinte equação:

$$T_{bit} = \frac{1}{Tx_{rate}} \tag{1}$$

Onde Tbit é o tempo de bit e Txrate é a taxa de transmissão do barramento.

Um bit CAN é dividido conforme apresentado na figura 14.

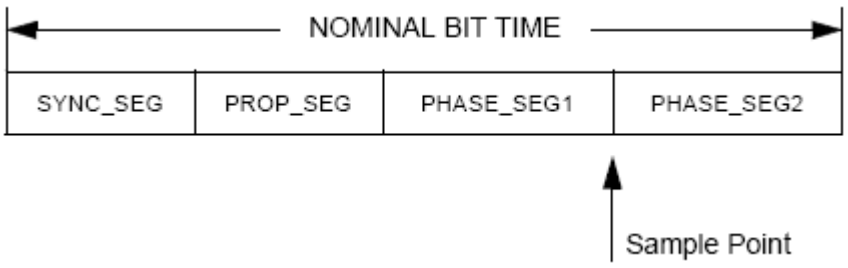

Figura 14 - Divisão temporal de um bit.

Onde SYNC_SEG é o segmento de sincronismo. Nesta janela de tempo acontece o sincronismo dos nós conectados à rede com a transmissão de um dado bit. PROP_SEG é o segmento de propagação. Este segmento compensa os atrasos na propagação do sinal pelo barramento. PHASE_SEG1 e PHASE_SEG2 são segmentos utilizados para compensar os erros de fase e para capturar o nível de tensão do bit, consequentemente, o nível lógico. O ponto exato de captura do nível de tensão do bit fica entre os dois segmentos de fase. O tempo destes dois últimos segmentos pode ser aumentado ou reduzido de acordo com a necessidade de ressincronização. A duração de cada segmento é definida como múltiplo de um *quanta*, que é uma unidade de tempo derivada da frequência do oscilador [1], [2]. A divisão dos segmentos em quantas segue conforme abaixo:

SINC_SEG – 1 quanta.

PROP_SEG – programável de 1 a 8 quantas.

PHASE_SEG1 – programável de 1 a 8 quantas.

PHASE_SEG2 – o tempo máximo de PHASE_SEG1 + tempo de processamento da informação, que é no máximo 2 quantas.

O bit CAN pode ter de 8 a 25 quantas, dependendo da frequência do oscilador e da configuração dos segmentos.

A camada física da rede CAN é definida em [2] como uma conexão entre um dado número de módulos eletrônicos a uma rede de comunicação de dados. É dividida em duas partes:

PLS (*phisical layer signalling*), responsável pela representação, tempo e sincronização do bit.

MAU (*media acess unit*), responsável pelo acoplamento físico do nó ao meio de transmissão. É dividido em PMA (*phisical medium attachment*) e MDI(*medium dependent interface*). O PMA define os circuitos de transmissão e recepção e o MDI define a interface elétrica e mecânica entre a MAU e o meio de transmissão.

A normatização ISO define as características da camada física como níveis de tensão e circuitos de interface com o barramento.

2.6.5. Aplicações CAN.

A rede CAN é hoje aplicada nas mais diversas áreas da indústria. Segue nesta seção alguns segmentos da indústria onde se tem uma rede CAN implementada em escala comercial.

No ramo de processos de fabricação podemos citar a implementação do protocolo Devicenet, implementado em linhas de produção de indústrias de manufatura.

Em relação a bens manufaturados encontra-se redes CAN em geladeiras, satélites, barcos e veículos automotivos em geral.

No que diz respeito ao ramo automobilístico, onde se tem a maior gama de aplicações de redes CAN, a SAE padronizou uma série de protocolos específicos para os diversos segmentos de veículos.

O protocolo SAE J1850 é utilizado por automóveis. A norma ISO 11783 é utilizada por maquinários agrícolas e discutida em [33] e [44].

Existem protocolos que tratam somente de diagnose, ou seja a forma de detecção e informação de uma falha, como o protocolo ISO 9141. O protocolo SAE J1939 é aplicado a veículos comerciais e também a maquinários agrícolas.

O foco deste trabalho é a análise de uma rede CAN implementada com o protocolo SAE J1939.

2.7. O Protocolo SAE J1939

2.7.1. Histórico.

.

A SAE definiu e publicou uma série de protocolos baseados na rede CAN, denominados *SAE Recommended Practices* para as diversas aplicações embarcadas[11], [12]. Por meio do sub-comitê de controle e redes de comunicação, *Truck & Bus Control and Communication network*, do comitê de elétrica de caminhões e ônibus da SAE, *Truck & bus Electrical Committee*, a SAE definiu, na categoria de *SAE Recommended Practice*, o protocolo SAE J1939 para aplicações em veículos comerciais [3]. Este padrão é aberto e define, como veremos mais detalhadamente ao longo deste capítulo, parâmetros necessários às tecnologias aplicadas em veículos comerciais. Define também, além de outras

pontos importantes da comunicação em rede, mensagens, identificadores e prioridades, além do agrupamento de parâmetros de comum característica, de forma a otimizar o uso e a transmissão da informação.

O protocolo SAE J1939 foi publicado em oito volumes, cada qual relativo a uma camada de rede, em analogia ao modelo OSI. A numeração dos volumes não é sequencial e segue os números indicados na sequência do nome do protocolo, exemplo: SAE J1939/11 , onde 11 é a numeração do volume. As numerações são 11, 15, 21, 31, 71, 73 e 81. [3], [4], [5], [6], [7], [8], [9].

Nesta seção será apresentada de forma sucinta as principais características de cada volume.

2.7.2. SAE J1939-11/15 – *physical layer*.

A camada física de uma rede CAN é discutida e definida em [1] e [2]. Porém no protocolo SAE J1939 o documento [3] define as características da camada física sendo o meio físico um par trançado com malha de terra.

O meio físico com dois canais, CAN_High e CAN_Low, os níveis de tensão, e os resistores casadores de impedância são definidos conforme [1] e [2]. O máximo número de nós na rede é definido em 30.

Os circuitos de interface e as impedâncias internas são definidos em [3].

O protocolo J1939 determina a velocidade do barramento em 250 [kbps]. Isto define o tempo de bit em 4 [μs]. As divisões temporais de um bit são determinadas conforme [1] e [2], e são como apresentadas na seção 2.6.4.

O documento [4] define como meio de transmissão um par trançado porém sem malha de terra.

Em relação à arquitetura elétrica é definido em [3] e em [4] um comprimento máximo de 40 metros para o barramento principal e 1 metro de comprimento máximo dos ramais para uma velocidade de rede máxima de 1Mbps.

2.7.3. SAE J1939/21 - *Data link layer*

Primeiramente definida por [1] e [2], a camada de enlace de dados recebe, neste documento [5] características exclusivas para o protocolo J1939.

É definido o uso exclusivo de frames estendidos, previamente definidos por [1], ou seja, com o identificador de 29 bits. Esta informação é relevante no que diz respeito à análise temporal que será apresentada adiante.

As figuras 15 e 16 mostram as divisões de um frame standard, onde o identificador é formado por 11 bits, e de um frame estendido, onde o identificador é formado por 29 bits.

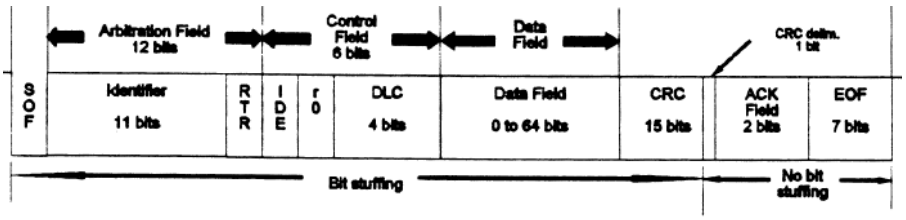

Figura 15 - Standard CAN Frame.

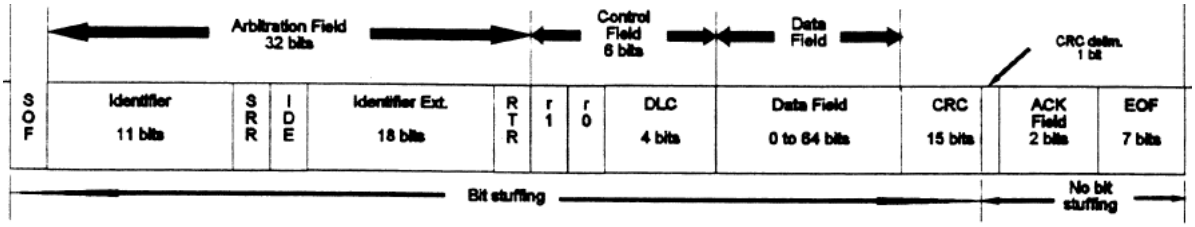

Figura 16 - Extended CAN Frame.

A análise a partir de agora se concentra no frame estendido, pois é o tipo utilizado nas mensagens do protocolo J1939.

O termo *bit stuffing* apresentado nas figuras 15 e 16 significa a região da mensagem que está sujeita ao acréscimo de um bit para cada sequência de cinco bits de mesmo nível lógico. O bit acrescido tem nível lógico inverso ao da sequência. Este artifício evita a perda de sincronismo por parte dos controladores CAN , dado que a primeira parte do bit, o *edge*, é o segmento de sincronismo, e também é uma das formas de detecção de erros.

Os 32 bits do *arbitration field* [2] e os seis bits do *Control Field* mostrado na figura 16 são definidos pelo protocolo J1939 conforme tabela 2.

Arbitration and Control field bits			
Position	***CAN***	***J1939***	***notes***
1	SOF	SOF	Start of frame - dominant
2	ID28	P3	Priority
3	ID27	P2	Priority
4	ID26	P1	Priority
5	ID25	R1	SAE Reserved Bit - dominant
6	ID24	DP	Data page
7	ID23	PF8	PDU Format - part of PGN.
8	ID22	PF7	PDU Format - part of PGN.
9	ID21	PF6	PDU Format - part of PGN.
10	ID20	PF5	PDU Format - part of PGN.
11	ID19	PF4	PDU Format - part of PGN.
12	ID18	PF3	PDU Format - part of PGN.
13	SRR	SRR	Substitute remote request
14	IDE	IDE	Identifier extension bit
15	ID17	PF2	PDU Format - part of PGN.
16	ID16	PF1	PDU Format - part of PGN.
17	ID15	PS8	PDU Specific - PDU Format extension or Destination Address
18	ID14	PS7	PDU Specific - PDU Format extension or Destination Address
19	ID13	PS6	PDU Specific - PDU Format extension or Destination Address
20	ID12	PS5	PDU Specific - PDU Format extension or Destination Address
21	ID11	PS4	PDU Specific - PDU Format extension or Destination Address
22	ID10	PS3	PDU Specific - PDU Format extension or Destination Address
23	ID9	PS2	PDU Specific - PDU Format extension or Destination Address
24	ID8	PS1	PDU Specific - PDU Format extension or Destination Address
25	ID7	SA8	Source Address
26	ID6	SA7	Source Address
27	ID5	SA6	Source Address
28	ID4	SA5	Source Address
29	ID3	SA4	Source Address
30	ID2	SA3	Source Address
31	ID1	SA2	Source Address
32	ID0	SA1	Source Address
33	RTR	RTR	Remote Transmission Request
34	r1	r1	CAN Reserved bits
35	r0	r0	CAN Reserved bits
36	DLC4	DLC4	Data Length Code
37	DLC3	DLC3	Data Length Code
38	DLC2	DLC2	Data Length Code
39	DLC1	DLC1	Data Length Code

Tabela 2 - Sequência de bits do *arbitration field* e do *control field*.

Onde:

P – Prioridade

R – bit reservado SAE

DP – Data page: provisão para duplicar o número de identificadores possíveis.

SRR – Substitui o RTR para frames *standard*. É transmitido “recessivo”.

IDE – Extensão do Identificador

RTR – Solicitação de transmissão remota

PF – *PDU format*: componente do PGN (*Parameter Group Number*). Define o nome da mensagem.

PE – PDU específico: componente do PGN. Caso o PF tenha valor menor do que 240dec, o PE indica o endereço de destino, ou *Destination Adress*. Caso o PF seja maior do que 240dec , o PE indica apenas o complemento do PGN e é chamado de *group extension*.

DLC – *Data Length Code*: tamanho do campo de dados, conforme já definido no capítulo anterior.

O PDU, *Protocol data unit* , é formado pelas informações apresentadas na tabela 1. O PDU é enviado pela sub-camada LLC à sub-camada MAC para que o frame completo seja composto.

O campo de dados de cada mensagem é formado por parâmetros físicos pré-definidos. Cada grupo de parâmetros recebe um número, o PGN . Este número é formado pelo PDU *Format* e o PDU específico.

A definição do PGN de cada mensagem está diretamente ligada à definição de sua prioridade, pois o PGN é parte do identificador, que por sua vez, faz parte do "*arbitration field*" . Isto significa que quanto menor é o valor numérico de um PGN, consequentemente de um identificador, maior é a prioridade da mensagem e mais chance esta mensagem tem de conseguir acesso ao barramento. Na análise temporal que será apresentada posteriormente, será visto que a prioridade de uma mensagem está inversamente relacionada ao tempo que esta pode ficar esperando a vez de acessar o barramento.

Em [5] são definidos endereços específicos para mensagens chamadas proprietárias, ou seja, mensagens formadas por PGNs livres, que não tem parâmetros associados. O projetista de uma arquitetura embarcada pode definir parâmetros específicos para uma tecnologia ou aplicação nova e associá-los a um destes PGNs livres, criando uma mensagem proprietária. Nestes casos a taxa de transmissão da mensagem também é definida no projeto. Esta duas definições, a do PGN, identificador a ser utilizado, e a da taxa de transmissão de uma mensagem proprietária, devem ser feitas com cautela, pois podem comprometer a escalonabilidade da rede. Este assunto será tratado em conjunto com a análise temporal aplicada ao protocolo SAE J1939 posteriormente.

Existem, dentro do identificador SAE J1939 três bits de prioridade. Estes bits podem ser definidos basicamente de duas formas: 03 (011_{bin}) para mensagens de alta prioridade ou 06 (110_{bin}) para mensagens de baixa prioridade. Existem casos

de utilização de outros números. Conforme definido na seção 4.2 do capítulo 4 o acesso ao barramento segue os padrões CAN [1] onde um bit dominante, de nível lógico “0”, sobrescreve um recessivo, de nível lógico “1”, de forma que no momento que um nó envia recessivo e recebe, ou lê, dominante, significa que há algum nó de maior prioridade transmitindo ao mesmo tempo. Esta arbitrariedade se alonga até o fim do *Arbitration Field*, ou seja, duas mensagens podem ter o mesmo valor numérico de prioridade (3 bits de prioridade) e a decisão de quem acessará o barramento será tomada pelo valor do ID. Como o ID é formado por diversas informações e algumas delas têm valores fixos, conforme tabela 1, a informação mais importante no momento de definição de acesso ao barramento, em uma arquitetura SAE J1939, é o PGN.

O campo de dados de uma mensagem CAN pode ter até 64 bits, porém o tamanho de determinados pacotes de dados supera estes 64 bits. O protocolo de transporte, definido em [5], permite a transmissão em multipacotes.

A política de escalonamento para o protocolo SAE J1939 é a não preemptiva de prioridades pré-fixadas [27] [31], ou seja, as prioridades já estão definidas nos campos específicos, e a transmissão não é interrompida para que uma mensagem de maior prioridade tenha acesso ao barramento.

De acordo com [5] as mensagens podem ser de cinco tipos: comando, *request*, *broadcast*, *acnowledgment* e *group functions*.

2.7.4. SAE J1939/31 - *Network Layer*

A camada de rede para o protocolo SAE J1939 é definida em [6].

Neste documento os componentes de rede são citados e definidos como bridges, roteadores, gateway, repeaters, entre outros.

Os parâmetros e regras para uma bridge são definidos, assim como sua função de filtro de uma rede para outra. Mensagens de rede para acesso aos filtros e parametros de rede também são definidas.

Para considerações de arquitetura de rede foi definida uma arquitetura com somente uma rede, sem gateways, de acordo com a tabela de mensagens pré definidas no protocolo SAE J1939. Além destas mensagens, foram consideradas algumas mensagens proprietárias de forma a piorar a questão temporal da análise.

2.7.5. SAE J1939/71 - *Vehicle Application Layer*

As definições que se seguem a partir de agora representam a vocação do protocolo SAE J1939. Até então, para as camadas física e enlace de dados, quase todas as definições seguiram os estudos prévios de [2] e de [1], exceto pelo identificador da mensagem, disponibilizado pela sub-camada LLC à sub-camada MAC em uma transmissão, e vice-versa em uma recepção, que foi particionado conforme mostrado na seção anterior.

Em [7] todos os parâmetros representativos para a aplicação de uma rede em veículos comerciais são definidos. Esta definição inclui , para cada parâmetro, a variável física a que diz respeito, a resolução da medição, o range de medição, a escala da medição, o tamanho da palavra que o representará, um número identificador do parâmetro e, principalmente, a que grupo de parâmetros este estará associado. Estas definições serão utilizadas pelos softwares da camada de aplicação ao rodar as rotinas e algorítmos de controle ou qualquer que seja a responsabilidade a que foram destinados.

O número do parâmetro é chamado de SPN, *Suspect Parameter Number* , e é utilizado para diagnose. A próxima seção diz respeito exclusivamente à diagnose.

São definidas também as variantes de cada parâmetro, caso sejam pertinentes. Por exemplo, se um parâmetro representa o status de um interruptor, deverá seguir o seguinte critério: “00” para representar desabilitado, “01” para representar habilitado”, “10” para representar erro e “11” para representar não disponível.

Os parâmetros estão divididos como segue:

61 parâmetros de controle, 21 parâmetros de estado para systems de *drivetrain*, 14 parâmetros de controle para sistemas de *drivetrain*, 5 parâmetros de configuração para sistemas de *drivetrain*, 312 parâmetros de informação, e 108 parâmetros de status de informação. Tem-se definidos pelo protocolo J1939 um total de 521 parâmetros aplicados à veículos comerciais.

Estes parâmetros estão agrupados em PGNs por função ou por taxas de repetição similares e/ou por fazerem parte de um mesmo sistema ou subsistema. O protocolo SAE J1939, por meio de [7], define 145 PGNs seguindo os conceitos técnicos de cada parâmetro e as definições da camada de enlace de dados [5] no que diz respeito ao valor em hexadecimal apropriado à cada PGN.

Como já foi dito anteriormente, a análise técnica de cada parâmetro, consequentemente a definição do valor do PGN e da taxa de transmissão definem a prioridade e as características temporais de rede de uma mensagem.

2.7.6. SAE J1939/73 - Application Layer –Diagnostics

A diagnose para o protocolo SAE J1939 é definida em [8].

A diagnose de um sistema representa a capacidade deste de perceber situações excepcionais, indicar ao controlador do sistema, quando necessário, e tomar ações, também se necessário. Para tal foi definido um número para cada parâmetro do protocolo SAE J1939, o SPN [7]. Este número associado a outras informações definem, em [8], um código de diagnose, o DTC (*Diagnostic Trouble code*). O DTC é utilizado para indicar a ocorrência de falhas nos componentes dos nós conectados à rede CAN. A figura 17 indica o conteúdo de um DTC.

	DTC																													
	Byte 3 8 least significant bits of SPN (bit 8 most significant)								Byte 4 second byte of SPN (bit 8 most significant)								Byte 5 3 most significant bits of SPN and the FMI (bit 8 SPN msb and bit 5 FMI msb)								Byte 6							
	SPN																			FMI					CM	OC					
J1939 Frame Format	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
	1	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0

Figura 17 - Formato de um DTC.

Onde:

SPN – *Suspect Parameter Number*, 19 bits. Número associado a cada parâmetro definido no protocolo J1939. Indica em qual componente ou variável física ocorreu a falha.

FMI – *Failure Mode Identifier*, 5 bits. Número que indica o modo de falha. Em [8] uma série de FMIs são definidos. Porém uma janela de numeração não definida foi deixada para futuras necessidades, por exemplo, de um modo de falha novo.

OC – *Occurence Count*, 7 bits. Número que indica a quantidade de falhas ocorridas.

Os DTCs são transmitidos dentro de mensagens específicas chamadas DM – *Diagnostic Messages*. O protocolo SAE J1939, em [8], define 19 mensagens de diagnose, cada qual com seu conteúdo e propósito.

A diagnose veicular é implementada na camada de aplicação, uma vez que analisa, em seus algorítmos de controle, os dados vindos das camadas inferiores, e caso necessário ou quando solicitado, transmite uma mensagem de diagnose

Durante um projeto automotivo a definição e implementação de uma estratégia de diagnose é feita pelo time de desenvolvimento, como disciplina de validação e certificação do produto. Esta estratégia de diagnose pode vir como estratégia do produto de uma montadora ou vir de legislações aplicadas aos sistemas automotivos, como as portarias do CONAMA (Conselho Nacional do Meio Ambiente), órgão ligado ao Ministério do Meio Ambiente, cujo conteúdo diz respeito à emissões de gases pelos veículos automotores.

A diagnose veicular auxilia a equipe de desenvolvimento de produto, durante a fase de desenvolvimento, e auxilia a assistência técnica das montadoras, por meio das ferramentas de diagnose, durante toda a vida do veículo. Em [8] são definidos também as políticas de acesso às informações de diagnose de forma a garantir que somente as informações permitidas serão acessadas.

.

2.7.7. SAE J1939/81 - Network Management.

Em [9] a configuração de endereços de rede e a identificação de cada ECM (*eletronic control module*) são definidas. No protocolo SAE J1939 os sistemas aplicados à veículos comerciais cujos módulos eletrônicos de controle estão conectados em rede, têm seu endereço de rede definido. Endereços para novos sistemas eletrônicos podem ser definidos, contanto que não coincidam com os já definidos pelo protocolo.

3. ANÁLISE TEMPORAL E ESCALONAMENTO NA REDE CAN

3.1. Modelamento Temporal na Rede CAN.

A especificação da rede CAN define que a política de acesso ao barramento é do tipo Bitwise Arbitration, ou seja , existe colisão mas não é destrutiva e , após colisão de duas informações diferentes, apenas o bit dominante permanece, no caso, o nível lógico “0”. A definição de prioridade pode ser feita on-line ou off-line. Estudos mostram diferentes políticas de escalonamento aplicadas à rede CAN. Em [15] uma comparação entre as diferentes políticas de escalonamento, *deadline monotonic* e EDF (*earliest deadline first*), é apresentada e uma solução mista é proposta. Em aplicações automotivas, baseadas em protocolos pré-definidos como o SAE J1939 e SAE J1587, a definição de prioridade é feita off-line caracterizando um escalonamento fixo, off-line e não dinâmico.

Na rede CAN a mensagem é totalmente transmitida, do primeiro ao último bit do frame [1]. A menos que algum erro seja detectado, a transmissão não é interrompida por mensagens de maior prioridade que estejam prontas para ser transmitidas. Desta forma o escalonamento é não preemptivo. A política de escalonamento de uma rede CAN pode ser definida como escalonamento não preemptivo de prioridade fixa. A contribuição do gerenciamento de erros será analisada em seção à parte.

Desde o fim do século passado teorias foram apresentadas de forma a quantificar e mensurar as questões temporais de uma rede CAN. Suas evoluções são brevemente citadas no decorrer deste próximo parágrafo.

As propostas de solução para o problema da concorrência de processos a um mesmo recurso foram tratadas em muitos trabalhos. As teorias, inicialmente,trataram do escalonamento aplicado a processadores, como em [10] ou em [11]. Esta base foi utilizada por [12] na proposta de análise temporal aplicada às redes CAN baseada no processador CAN Intel 82527. Em [14] uma análise é proposta considerando um modelamento temporal ideal. Em [13] um comportamento temporal não ideal é aplicado aos processadores Intel 82527 e Philips 82C200. Como existem diferenças entre processadores CAN, os

desenvolvedores devem assegurar que a escolha do processador está de acordo com o comportamento esperado para a rede em uma determinada aplicação.

Em um paralelo entre uma rede CAN e um processador, considera-se a mensagem como um processo, e o meio de transmissão como o recurso, o processador. Em um computador os processos lutam pelo recurso processador e em uma rede as mensagens lutam pelo recurso meio de transmissão.

A análise temporal é feita considerando os delays existentes no caminho da informação, desde a solicitação até a entrega, passando pelo tempo de processamento e transmissão. O intuito é de modelar , em um ambiente matemático, uma forma de checar se, para um determinado grupo de mensagens, os deadlines estão garantidos, assumindo, sempre no pior caso, que $Dm \leq Tm$, onde *Dm* é o deadline de uma mensagem m e *Tm* é seu período.

A métrica para se garantir que um deadline seja atendido representa o cálculo do tempo de resposta de uma mensagem. O tempo de resposta de uma mensagem é o tempo decorrente entre a solicitação de uma mensagem, se for aperiódica, até o tempo de recebimento por quem deve recebê-la. No pior caso, temos o principal parâmetro desta análise temporal, o tempo de resposta no pior caso, ou WCRT, *Worst Case Response Time*.

De acordo com [12] e [13] o tempo de resposta de uma mensagem "*m*" no pior caso, dado por *Rm*, contempla três etapas temporais e é representado por:

$$R_m = J_m + W_m + C_m \tag{2}$$

Onde:

Jm é chamado de atraso de processamento e representa o atraso de tempo para processamento da informação física e preparo da mensagem para ser enfileirada para transmissão.

Wm é o tempo de fila e representa o tempo gasto pela mensagem na fila de acesso ao barramento.

Cm é o tempo de transmissão e representa o tempo total de transmissão de todos os bits da mensagem incluindo os *bits stuffing* e o espaço inter-frames.

Em [14], trabalho publicado em sequência a [12], o cálculo de tempo de resposta de uma dada mensagem "m" foi resumido a:

$$R_m = W_m + C_m \tag{3}$$

Onde:

Wm é o tempo gasto pela mensagem na fila de acesso ao barramento.

Cm é o tempo de transmissão.

A supressão do elemento *Jm* indica representa o fato considerado de que não há diferenças entre os tempos de processamento e que este tempo de processamento da informação é sempre menor do que a necessidade de se enfileirar uma mensagem com tal informação [31].

O termo *Cm*, que representa o tempo de transmissão, é contabilizado em [12] como a soma do tempo de transmissão de um frame standard mais a quantidade máxima de *bits stuffing*, considerando 34 bits de overhead sujeitos a *stuffing* e uma largura de 5 bits:

$$C_m = \left(\left\lfloor \frac{34 + 8S_m}{5} \right\rfloor + 47 + 8S_m \right) \tau_{bit} \tag{4}$$

Da equação (4), o número resultante da função *floor* representa o máximo de bits *stuffing* que a mensagem pode receber. A soma (34 + 8 Sm) representa exatamente a quantidade de bits de um frame *standard* sujeitos a *stuffing*, mostrada na figura 15. O número 47 é a quantidade de bits de *overhead* de um frame *standard* , 44, mais três bits relativos ao espaço interframe.

Em [17] e [31] é mostrado que, como o bit acrescentado como *stuffing* bit entra na sequência de 5 bits para determinar onde é necessário o próximo *stuffing* bit, a equação anterior se torna:

$$C_m = \left(\left\lfloor \frac{34 + 8S_m - 1}{4} \right\rfloor + 47 + 8S_m \right) \tau_{bit} \tag{5}$$

Onde:

Sm é o número de bytes do campo de dados.

τbit é o tempo de um bit em milisegundos.

Também em [12] e [14] o conceito de "tempo de fila" foi apresentado aplicado à rede CAN. O termo Wm é dividido em duas partes. A primeira relacionada ao tempo de bloqueio sofrido pela mensagem devido a qualquer outra mensagem de prioridade mais baixa e que começou a ser transmitida em instante imediatamente anterior. A segunda parte está relacionada a todas as outras mensagens de maior prioridade que podem ganhar o acesso ao barramento durante a arbitração.

O tempo de fila de uma mensagem "*m*", dado por *Wm*, é apresentado em [12] como:

$$W_m^{n+1} = B_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{W_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k \qquad (6)$$

Onde:

τ_{bit} é o tempo de um bit em milisegundos.

Hp(m) é o grupo de mensagens com prioridade maior do que a mensagem "*m*".

Jk é o atraso de processamento para cada mensagem *k*.

Tk é a taxa de transmissão de cada mensagem *k*.

Ck é o tempo de transmissão de cada mensagem *k*.

A primeira parte da equação (6), o elemento Bm, representa o tempo máximo de bloqueio por mensagens de menor prioridade, e é apresentado em [12] como:

$$B_m = \max_{k \in lp(m)} C_k \qquad (7)$$

Onde:

lp(m) é o grupo de mensagens de prioridade menor do que a mensagem "*m*".

Ck é o tempo de transmissão de cada mensagem *k*.

A segunda parte da equação (6) representa a contribuição de cada mensagem de mais alta prioridade no atraso de fila de uma dada mensagem.

A teoria apresentada em [12] foi revisada em [31] devido ao comportamento não preemptivo da rede CAN, e sua influência sobre as várias instâncias de cada mensagem na fila de transmissão. O conceito de "*busy period*", ou período de indisponibilidade, foi introduzido. Este período significa o tempo em que o barramento está sendo ocupado com a transmissão de um conjunto de mensagens de mais alta prioridade do que a mensagem *m*. Seu tamanho para uma mensagem *m*, no pior caso, é dado por:

$$t_m^{n+1} = B_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{t_m^n + J_k}{T_k} \right\rceil C_k \qquad (8)$$

Onde:

t_m é o tamanho do período de indisponibilidade, para uma mensagem *m*.

hp(m) é o conjunto de mensagens de prioridade mais alta do que a mensagem *m*.

O valor inicial para *tm* pode ser *Cm* devido ao problema da mensagem de menor prioridade [31], cujo valor de *Bm* é zero.

De acordo com [31], durante o período de indisponibilidade um conjunto de instâncias *q* da mensagem *m* podem se tornar disponíveis para transmissão. O tempo de fila para cada instância é dado por:

$$W_m^{n+1}(q) = B_m + qC_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{W_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k \qquad (9)$$

Onde:

q é uma dada instância de uma mensagem m.

O número de instâncias varia de 0 a *Qm*-1, onde *Qm* é o número de instâncias de uma mensagem *m* prontas para serem transmitidas antes do fim de um período de indisponibilidade. Em [31] Qm é dado por:

$$Q_m = \frac{\lceil t_m + J_m \rceil}{T_m} \tag{10}$$

Onde:

tm é o tamanho do período de indisponibilidade.

Jm é o atraso de processamento da mensagem m.

Tm é a taxa de transmissão da mensagem m.

Neste caso, o tempo de resposta no pior caso para uma mensagem m é o máximo valor entre todas as instâncias desta mensagem dentro de um dado período de indisponibilidade.

$$R_m = \max_{q=0\ldots Q_m-1} R_m(q) \tag{11}$$

E *Rm(q)* é o tempo de resposta no pior caso para cada instância de uma mensagem m.

$$R_m(q) = J_m + W_m(q) - qT_m + C_m \tag{12}$$

De forma a simplificar o cálculo de *Wm*, é proposto, também em [31], o fator de bloqueio como o máximo valor entre *Bm* e *Cm*. A inclusão do valor de *Cm* como opção para o cálculo de *Bm* se dá pela situação da mensagem de menor prioridade, onde *Bm* é zero, e também devido às instâncias anteriores da mensagem *m,* que agora devem ser consideradas como potencial bloqueio à transmissão de *m*.

$$W_m^{n+1}(q) = \max(B_m + C_m) + \sum_{\forall k \in hp(m)} \left\lceil \frac{W_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k \tag{13}$$

3.2. Modelamento do Erro na Rede CAN

A rede CAN tem cinco formas de detecção de erros. São elas:

Bit error – formato do bit.

CRC – Cyclic Redundancy Check

Ack – bit de acknowledgement.

Format – formato do frame incompatível.

Stuffing bit – a cada 5 bits de mesmo nível lógico, um é acrescentado, de nível lógico diferente.

A sinalização do erro é baseada em *error flags* que podem ser *active* ou *passive*. A informação de que houve ou não um erro é confinada em dois contadores TEC (transmit error counter) e REC (receive error counter). Para cada tipo de erro existe uma regra de confinamento. Cada nó conectado a uma rede pode ser elevado a condição de "error passive", "error active" ou "bus-off", dependendo do estado dos contadores [2].

Na ocorrência de um erro, seja qual for sua origem, a consequência imediata é a transmissão de um error frame, por quem detectou o erro, seguido da retransmissão da mensagem falha. Estas transmissões e retransmissões representam um custo temporal ao barramento. Estudos passados incluíram, no calculo do tempo de resposta de uma mensagem, no pior caso, uma função "erro" , que representa o tempo de utilização do barramento no tratamento de erros.

Em [12] a função *error recovery overhead function* foi apresentada como:

$$E_m(t) = \left(n_{error} + \left\lceil \frac{t}{T_{error}} \right\rceil - 1 \right) \cdot \tau_{bit} + \max_{\forall k \in hp(m)} (C_k) \qquad (14)$$

Onde o termo do primeiro parênteses representa a quantidade de erros que pode acontecer em um período. O termo do segundo parênteses representa, para cada erro, o tempo de um error flag e a transmissão de uma mensagem, no pior

caso, que seria a maior mensagem dentre as de maior prioridade do que a mensagem *m*.

Também em [12], foi acrescentado no cálculo de *Wm* o termo *Em(Wm+Cm)* .

A mesma análise é feita em [14].

O modelo matemático do erro foi modificado em [18]. Primeiramente o número de bits gastos para sinalizar o erro, que antes era considerado 29 bits, e no trabalho referenciado considera-se 31 bits. A função $E_m(t)$ é apresentada como uma combinação de erros vindos de múltiplas fontes, divididos em erros simples e erros de burst (rajada). A função erro, $E_m(t)$, se apresenta agora, para *k* fontes de interferência, como:

$$E_i(t) = E_i^1(t) + E_i^2(t) + E_i^3(t) + \ldots + E_i^k(t) \quad (15)$$

Onde, para cada fonte de interferência *n*:

$$E_i^n(t) = Bu^n(t) \times O_i^b + \max(0, I^n - \tau_{bit}) + \mathrm{Re}^n(t) \times O_i^r + \max(0, I_r^n - \tau_{bit}) \quad (16)$$

Sendo:

$Bu^n(t)$ o número de interferências causadas por erros de burst.

$Re^n(t)$ o número de interferências causadas por erros simples.

I^n o tempo de duração de uma dada interferência causada por uma fonte *n* de erro.

$Max(0,I^n-\tau_{bit})$ representa o atraso gerado por uma interferência caso esta seja maior do que um tempo de bit.

O_i é o overhead de um erro, considerado de mesmo valor para erros de burst e erros simples, e representado pela equação a seguir:

$$O_i = 31 \times \tau_{bit} + \max_{k \in hp(i)} C_k \quad (17)$$

A equação (17) é equivalente ao segundo termo da função *error recovery overhead* proposta em [12], considerando 31 bits de overhead, e é um caso especial do modelo proposto em [18].

Algumas características da rede CAN, no que diz respeito a tratamento de erros, podem gerar situações desagradáveis e aumentar o tempo de inacessibilidade do barramento. É o que foi estudado e proposto em [16], [19], [20] e [21], em relação ao problema do "penúltimo bit". Um erro no penúltimo bit de uma mensagem CAN pode gerar duas situações indesejadas: IMO (Inconsistent Message Omission) e IMD (Inconsistent Message Duplicate). Alguns protocolos foram propostos nestes trabalhos de forma a se encontrar um condição ideal de tratamento de erros e , assim, se ter um sistema tolerante à falha.

Em [17] o conceito de inacessibilidade foi apresentado e incluído na análise de tempo de resposta. A equação para o termo *Wm*, inicialmente proposta em [12], se torna:

$$W_m^{n+1} = B_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{W_m^n + \tau_{bit}}{T_k} \right\rceil C_k + Ina_{bus} + Ina_{transc} \qquad (18)$$

Onde:

Ina_{bus} é o tempo máximo de inacessibilidade do barramento e é dado por:

$$ina_{bus} = n_{bus} \times \left\lceil \frac{W_m + C_m}{T_{bus}} \right\rceil \times t_{ina} \qquad (19)$$

A situação de uma sequencia de erros afetando uma mensagem *m* está coberta pelo termo n_{bus}, que representa o numero de erros seguidos dentro de um tempo T_{bus}.

Ina_{transc} é o tempo máximo de inacessibilidade do transceiver CAN:

$$ina_{transc} = 16 \times t_{ina} \tag{20}$$

Após 16 erros consecutivos o transceiver entra em estado de "error passive".

O termo t_{ina} $=C_{max}+C_{error}+C_{IFS}$, é a somatória dos tempos de inacessibilidade devido a: tempo de transmissão de um error frame (C_{error}), tempo de transmissão do Inter Frame Space (C_{IFS}) e o tempo de retransmissão de uma mensagem, no pior caso.

Além dos modelos acima apresentados, a literatura traz análises probabilísticas de forma a otimizar os modelos. Em [24] a distribuição de Poisson é utilizada para descrever a função Em(t) como uma distribuição probabilística, onde uma árvore de probabilidades indica a probabilidade de uma dada mensagem perder seu deadline. Em [16] funções de probabilidade são apresentadas para se chegar aos números de "Inconsistent Message Omissions" e "Inconsistent Message Duplicates" dentro de um determinado tempo. Os contadores de erros TEC e REC são apresentados em [29] sob uma análise probabilística, onde uma cadeia de Markov é utilizada para determinar a probabilidade de um nó chegar ao estado de bus-off. As funções probabilísticas são utilizadas de forma a estimar a ocorrência de determinado evento dentro de um intervalo de tempo e de um determinado comportamento de rede.

Diversas formas de análise de rede sob condições de erros são propostas na literatura. A análise de erros aplicada ao protocolo CAN será apresentada posteriormente no próximo capítulo.

3.3. Otimização do Barramento.

A otimização da utilização do recurso barramento e consequente redução da probabilidade de erros é explorada em [22], [23] e [25].

Em [22] a redução da quantidade de bits *stuffing* é alcançada com a utilização de uma máscara XOR de forma a reduzir bits seguidos de mesmo valor.

A escolha seletiva das prioridades das mensagens é proposta em [23] e [25] com a intenção , também, de redução de bits *stuffing*. A combinação de técnicas

como mascara XOR, escolha seletiva de prioridade aliadas a cálculos de probabilidades são aplicadas em frames CAN de forma a reduzir o pessimismo dos modelos estudados, onde sempre se considera a quantidade de bits *stuffing* no pior caso.

3.4. Modelamento Temporal do Protocolo SAE J1939.

No protocolo SAE J1939, à exceção das mensagens proprietárias, onde a definição da prioridade fica a cargo do projetista, as prioridades das mensagens são pré-definidas. Esta característica define a política de escalonamento deste protocolo como *Escalonamento Não Preemptivo de Prioridade Fixa*, conforme apresentado no capítulo 2. Portanto, uma mensagem de prioridade alta tem um identificador de valor baixo. Como o processo de acesso ao barramento consiste em uma porta lógica "E", quanto menor o identificador maior sua prioridade de acesso ao barramento. Conforme discutido anteriormente, se mais de uma mensagem está no buffer de saída de um nodo, é necessário garantir que a mensagem de maior prioridade terá sempre o acesso ao barramento.

A análise apresentada na sequência tratará primeiramente do cálculo temporal e posteriormente o gerenciamento dos erros, tendo a finalidade de escolher o melhor modelo para ser aplicado ao protocolo SAE J1939.

Assumindo que o tempo de processamento da informação é sempre menor do que o período de disponibilidade de uma dada mensagem a ser transmitida, e que o exemplo utilizado para aplicação do modelo não contém gateways, o Jitter (J) utilizado é 0,2[ms], ou seja , o termo *Queuing Jitter(J)* da equação (2) será considerado 0,2[ms] no cálculo do *WCRT*. A tratativa de supressão do termo "J" no cálculo do *Worst Case Response Time* foi utilizada em [14]. Porém será utilizado o valor J = 0,2[ms] de acordo com o sugerido em [30].

De acordo com as equações apresentadas no capítulos 5 e 6 o tempo de resposta de uma dada mensagem *m* , na aplicação J1939 pode ser considerado como o apresentado por [31], na equação (11), abaixo:

$$R_m = \max_{q=0\ldots Q_m-1} \left(R_m(q) - q T_m \right) \tag{21}$$

Ou seja, será considerado que há mais de uma instância *q* da mesma mensagem a ser transmitida.

O tempo de resposta para cada instância *q* da mensagem *m* será considerado , neste estudo como a combinação da equação (3) e a equação (12):

$$R_m(q) = W_m(q) + qT_m + C_m \tag{22}$$

Antes da definição do termo *Cm*, tempo de transmissão, é necessário lembrar que o protocolo SAE J1939, apesar de permitir frames standard com ressalvas, define suas regras e toda a estratégia de rede exclusivamente para para frames estendidos, ou seja, frames cujo identificador contém 29 bits [5]. O uso de frames standard pode ser feito apenas como mensagens proprietárias e sob as condições especificadas no protocolo SAE J1939. Como o foco do estudo está no protocolo SAE J1939 será considerado apenas frames estendidos. Baseado nesta premissa, o tempo de transmissão, *Cm*, será representado pela equação (5).

O tempo de fila, Wm, sem o acréscimo devido ao tratamento de erros será representado pela equação (09) baseado na definição do tempo de resposta considerando múltiplas instâncias de uma dada mensagem m [31]. O cálculo de cada instância proposto em [31] será usado neste modelo.

3.5. Modelamento de Erros no Protocolo SAE J1939.

O tratamento de situação de erro dentro do protocolo J1939 segue as especificações de [1] sobre o CANBus. O sistema é protegido por cinco mecanismos de checagem de erros: Bits stuffing, código CRC, bit de Acknowledgement, erro de bit e erro de formato.

A análise seguinte compara os modelos de representação de erro apresentados no capítulo 5 e define a melhor combinação entre estes para ser aplicado no protocolo SAE J1939.

O tempo máximo de transmissão, Cm, definido pela equação (5), pode ser adaptado para o SAE J1939. Os 34 bits de overhead sujeitos a stuffing são provenientes de frames standard. Como a discussão está focada em frames estendidos, devemos considerar que 54 dos 64 bits de overhead são sujeitos a

stuffing, conforme mostrado em [31], ou seja, a equação (5), para frames estendidos fica:

$$C_m = \left(\left\lfloor \frac{54 + 8S_m - 1}{4} \right\rfloor + 67 + 8S_m \right) \tau_{bit} \tag{23}$$

E , também como em [31], pode ser simplificado para:

$$C_m = 80 + 10 S_m \tau_{bit} \tag{24}$$

Onde *Sm* é o número de bytes do campo de dados e *Tbit* é o período do bit.

A abordagem apresentada em [12], denotada pela equação (15) e reacessada em [18], pode ser comparada com o conceito apresentado em [17], onde a Inacessibilidade do barramento foi acrescida na conta do Worst Case Response Time, WCRT.

Nota-se que existem diferentes abordagens para modelamento de erros nos trabalhos pesquisados. De forma a simular o comportamento do barramento sob a ação de erros se faz necessário a comparação dos modelos matemáticos previamente propostos sob as mesmas condições, ou seja, sob os mesmos parâmetros de rede e o mesmo conjunto de mensagens.

A componente “erro” do tempo de latência no pior caso de um dado grupo de mensagens CAN foi calculado, para cada mensagem, no pior caso, de acordo com os modelos apresentados em[12], [17] e [18]. Os resultados são apresentados no capítulo 5.

O grupo de mensagens utilizado pertence ao protocolo SAE J1939 e foi aplicado a uma rede de 250 kbps.

O ambiente de simulação utilizado foi desenvolvido durante esta pesquisa, e inicialmente, implementado na forma de um software em VBA. Posteriormente foi utilizado o ambiente VisualStudio para implementar uma ferramenta de simulação em C#. Esta ferramenta será apresentada em detalhes no capítulo 5.

4. ALGORÍTMO GENÉTICO APLICADO AO ESCALONAMENTO DE MENSAGENS SAE J1939.

Após ter sido definido o modelo do comportamento temporal das mensagens do protocolo J1939 no capítulo anterior, são apresentadas neste capítulo considerações relevantes sobre algorítmos genéticos.

Os algorítmos genéticos nasceram na década de 80 como método de busca e vem sendo aplicado em diversas áreas da ciência da computação, da indústria , da medicina ,e da tecnologia da informação.

Este capítulo tem a finalidade de apresentar os conceitos básicos de algorítmos genéticos e sua aplicação na representação do problema em questão.

4.1. Teoria.

A busca por uma solução satisfatória, face a um dado problema, pode ser tratada como a evolução de um conjunto de possíveis soluções para uma que seja a melhor, ou a mais adequada. O problema analisado neste trabalho trata do tempo de resposta de mensagens do protocolo SAE J1939 aplicado a uma rede CAN. Neste caso, em uma condição de não atendimento aos requisitos temporais, tem-se uma quantidade grande de possíveis soluções, que passam pela mudança de prioridades, de taxas de transmissão, de *deadlines*.

De acordo com [37], a evolução é um método de busca utilizado quando se tem um enorme número de possibilidades de soluções. Dada esta definição, tomou-se a decisão de se aplicar algorítmos evolucionários a este problema. Os algorítmos evolucionários são aqueles que se utilizam de conceitos da biologia para refinar um grupo de soluções [36]. Os algorítmos genéticos são um grupo específico dos algorítmos evolucionários, tal como a programação genética, estratégias evolucionárias, programação evolucionária e sistemas de aprendizado. Na definição apresentada por [36], os algorítmos genéticos são diferenciados de outros algorítmos evolucionários pelo fato de que seu espaço de busca G é formado por sequências binárias. Este conceito faz com que seja possível a representação da

solução, que no caso do problema em questão seria uma sequência ótima de mensagens, dado que uma mensagem é, do ponto de vista lógico, uma sequência binária.

O conceito de evolução aplicada à algorítmos se iniciou nos anos 60 porém foi nos anos 70, quando John H. Holland propôs uma adaptação da natureza aos problemas computacionais [46], que o uso dos algorítmos genéticos como método de resolução de problemas foi formalizado e difundido. A partir dos trabalhos de Holland os algorítmos genéticos se tornaram mundialmente conhecidos e populares, sendo utilizados nas mais diversas áreas do conhecimento como medicina, química, economia, redes, processamento de imagens, matemática, design de circuitos elétricos, escalonamento [36].

O paralelo entre natureza e computação começa quando os componentes da genética são analisados. Da biologia temos que o cromossomo de um indivíduo é formado por genes. Nos algorítmos genéticos, um cromossomo é composto por uma sequência de bits, onde cada bit é um gene. Os possíveis valores para cada bit(gene) são chamados alellos, que neste caso podem ser “0” ou “1”, ou também um vetor binário. Cada gene deve representar uma característica do indivíduo. Dependendo do problema um indivíduo pode ser representado por um ou mais cromossomos. Cada cromossomo, ou indivíduo, representa uma possível solução o qual pode ser caracterizado por uma função de avaliação e classificado por uma função *fitness*. Os melhores indivíduos são selecionados para reprodução. O processo de reprodução executa mutações e/ou cruzamentos para que uma nova geração seja criada, avaliada, classificada, selecionada, e desta forma, continuando o processo de evolução. A partir de agora será referenciado o algorítmo genético pela sigla GA, do inglês *Genetic Algorithm*. Uma representação do ciclo de evolução de um GA é mostrada na figura 18 [36].

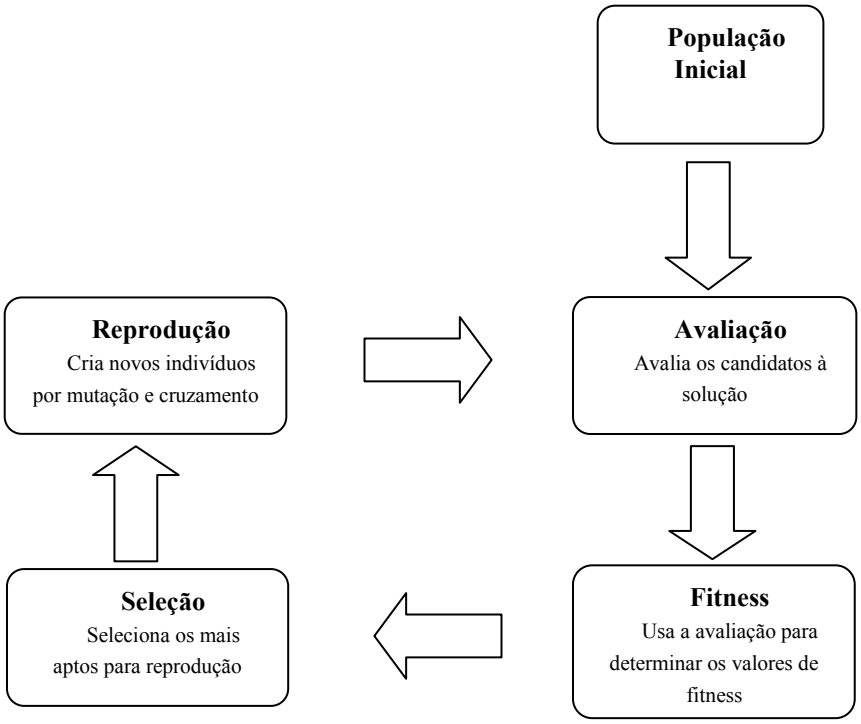

Figura 18 - Ciclo de um GA [36].

4.2. População Inicial

A população inicial é formada por um grupo inicial de possíveis soluções. Existem diferentes métodos para criar a população inicial, desde a utilização de um único indivíduo gerador ou a utilização de um grupo de indivíduos. Pesquisas anteriores aplicando GA em problemas de escalonamento de mensagens em redes de comunicação de dados definiram o indivíduo como sendo uma sequência de transmissão de um grupo de mensagens:[38] e [42]. Foi usado como população inicial, no caso do protocolo J1939, um grupo de sequências de mensagens geradas de um único indivíduo por processo de permutação, onde a diferença entre estas sequências é a posição de cada mensagem na fila, ou seja, sua prioridade. Cada sequência representa um cromossomo, ou indivíduo. Cada mensagem representa

um gene. Cada gene é composto por uma sequência de bits, que representa o conteúdo de cada mensagem.

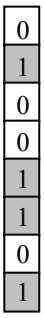

Figura 19 - Cromossomo básico composto por sequência de bits.

M1	01101101010111100100011111010010110010 11
M2	01010111001001001100111010100010100010 1011
M3	00110110101010111111110101000001010101 0101
M4	01111100101010100101010101010111010111 1
M5	00110110101010111111110101000001010101 0101
M6	01110101010101011101010100100100101 00101
M7	01000000101010110101010100010111010111 01
M8	01111111000000000100101010100100101111 1111
M9	00101010100011101001101101010110110110 1000
M10	01101001011010100100101101010101010011 0100

Figura 20 - Representação proposta para aplicação J1939.

4.3. Avaliação e *Fitness*

A função de avaliação, ou função objetivo, tem a finalidade de medir o quão boa é a performance de cada indivíduo de acordo com os parâmetros definidos [38]. A avaliação de um indivíduo depende das características de seu cromossomo. A correta definição da função de avaliação é de grande importância para a eficiência do algorítmo. Quanto melhor for a representatividade da função de avaliação em relação ao problema em questão, mais confiável será a "nota" do indivíduo. Consequentemente as tomadas de decisões em relação à classificação do indivíduo perante a população, em relação à seleção e reprodução também carregarão o nível de confiança adquirido. Neste estudo, o problema caracterizado é relativo à dimensão "tempo". A pergunta que se quer responder é: todas as mensagens de um "*data base*" sairão e chegarão aos destinos dentro do tempo

permitido, ou seja, dado uma configuração de períodos e prioridades para um grupo de mensagens, alguma perderá seu deadline? Portanto a função de avaliação escolhida utiliza toda a teoria apresentada nos capítulos anteriores para representação da qualidade do indivíduo, no que diz respeito ao "tempo". A função de avaliação é composta por duas etapas de cálculo. Em um primeiro momento, o tempo de resposta no pior caso (WCRT – Worst Case Response Time), "R", é calculado para cada mensagem, ou cada gene de um indivíduo, de acordo com o modelo discutido na seção 3. Então, a diferença entre o WCRT (R), e o deadline (D) , de cada mensagem é calculada e chamada de "d", onde d = D – R. O valor de performance para cada indivíduo, chamado de função de avaliação, *fa*, é calculado pelo somatório das diferenças "d" de cada mensagem, ou gene, mais a somatória das recompensas recebidas, menos a somatória das penalidades sofridas, de acordo com as características de cada indivíduo:

$$d_m = \left(D_m - R_m \right) \tag{25}$$

e,

$$fa_{g(m)} = \sum_{\forall m \in g(m)} \left(d_m \right) + \sum Rc - \sum P \tag{26}$$

Onde "d" é a performance de cada gene, "g(m)" é o grupo de genes de um cromossomo, ou seja, grupo de mensagens m, de um dado indivíduo, e "fag(m)" é a função de avaliação para cada indivíduo. Rc representa as recompensas e P representa as penalidades geradas a partir das características dos indivíduos. Se uma mensagem perde seu deadline o valor "d" se torna negativo e diminui o valor de avaliação do indivíduo. Somente a somatória de "d" não é suficiente para que a função de avaliação possa representar as diferenças de uma grande quantidade de indivíduos. Foram criadas então, penalidades e recompensas na função de avaliação de forma a aumentar a diferença entre as soluções boas e ruins. Estes acréscimos e decréscimos no valor da função avaliação dependem das características do indivíduo e são descritos na seção 4.8.

A função fitness, ou função de aptidão, é definida com base nos outros membros de uma geração [38] , [39]. Em linhas gerais indica como está cada indivíduo em

relação a toda a população, ou seja, classifica o indivíduo em relação à população. Existem diferentes métodos de implementação da função aptidão. Em [35] foi utilizada a função de aptidão de acordo com [38], representando o quão longe um indivíduo está da média da população:

$$fit_{g(m)} = \frac{fe_m}{fe_{avg}} \tag{27}$$

Em [36] outros métodos de se obter a função de aptidão são apresentados, como o Ranking de Pareto, onde a aptidão do indivíduo é determinada pela sua posição em um diagrama de Pareto obtido da distribuição dos valores de avaliação de uma população; o método da Competição, onde o indivíduo compete n vezes contra k outros indivíduos e a quantidade de vezes que perde ou ganha leva a seu valor de aptidão; a Soma Ponderada, onde são determinados pesos para os valores da função de avaliação, de modo que a soma ponderada dos valores da função de avaliação gera o valor de aptidão. Neste trabalho a função aptidão foi aplicada, assim como em [35] e em [38]. Após determinar o valor de aptidão, foi determinado um ranking para cada indivíduo.

4.4. Seleção

O processo de seleção é responsável por segregar os melhores cromossomos e descartar os piores, mantendo o processo de evolução. Os melhores cromossomos são aqueles que alcançam os maiores valores de fitness. Existem diferentes métodos seleção, dentre eles, seleção proporcional à aptidão, seleção truncada, competição, seleção ordenada, seleção por ranking e outros. Uma boa visão destes métodos está em [36]. Foi implementado, neste trabalho, o método de seleção truncada, onde foi considerado o ranking calculado na função de aptidão. Foi considerado 2/3 como valor de corte, ou seja, os 2/3 melhores indivíduos serão selecionados para reprodução. Uma boa parte da população, 66%, foi considerada nesta etapa, pois selecionar apenas os melhores pode gerar uma convergência precoce da solução para máximos intermediários.

4.5. Reprodução

O processo de reprodução é responsável por criar descendentes dos indivíduos selecionados de forma que uma nova geração seja criada. Os principais operadores no processo de reprodução são:

MUTAÇÃO: este operador muda geneticamente as características de um ou mais genes. Com este mecanismo a diversidade genética está garantida [38], e o espaço de busca é ampliado. De acordo com [39] o uso da mutação evita uma convergência prematura para uma solução boa, mas não ótima. A mutação é aplicada de acordo com um parâmetro randômico chamado taxa de mutação que será discutido posteriormente. As figuras 21 e 22 mostram exemplos de mutação e um único ponto e mutação em diversos pontos do cromossomo.

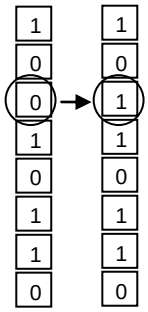

Figura 21 - Mutação em único gene.

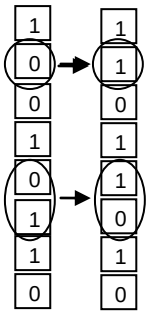

Figura 22 - Mutação em múltiplos genes.

PERMUTAÇÃO: este operador troca o valor genético de dois genes de um mesmo cromossomo, de forma aleatória, e de acordo com a taxa de permutação. A figura 23 exemplifica um cromossomo cujos genes sofreram permutação.

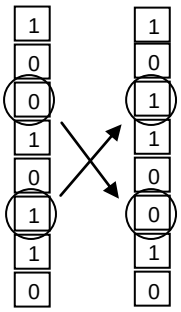

Figura 23 - Dois genes em permutação.

Neste estudo o operador mutação não foi utilizado pois geraria um indivíduo com genes repetidos, o que não seria a melhor solução. De acordo com [36] a permutação é útil na solução de problemas que envolvem a busca por uma sequência ótima de itens, como o escalonamento de mensagens estudado em [38]. De acordo com [37] a permutação pode representar problemas de escalonamento indicando a sequência de ocorrência de tarefas ou eventos, ou no caso deste trabalho, mensagens. Portanto o operador aplicado à este caso de otimização é a permutação. O processo de permutação no protocolo J1939 é mostrado na figura 24:

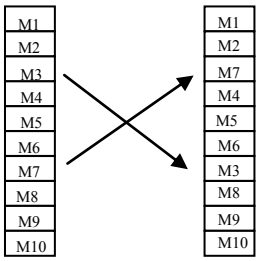

Figura 24 - Cromossomo J1939 em permutação.

A implementação da permutação no problema de escalonamento off-line de mensagens CAN teve, neste trabalho, sua peculiaridade. O conceito de "*permutação seletiva*" é apresentado neste momento como um método onde a permutação ocorre somente se a posição do gene escolhido for desfavorável à solução do problema. O fator *posição* é levado em consideração pelo fato de que a prioridade de uma dada mensagem define exatamente sua posição na sequência de escalonamento. A métrica para decisão sobre o quão favorável ou não é a posição

do gene é feita sobre o valor de *deadline*, e também utilizando a idéia do método de escalonamento *deadline monotonic*. A permutação seletiva segue o algoritmo abaixo:

```
Random x;
Msg x
Msg x+1
If ( DeadlineMsg x > DeadlineMsg x+1)
{
    Permutação(Msg x, Msg x+1);
}
end if
```

CRUZAMENTO: este operador reproduz o comportamento real da natureza recombinando os genes de dois cromossomos a partir de um ponto chamado ponto de cruzamento. O ponto de cruzamento é determinado aleatoriamente. O cruzamento pode ser feito com um ou dois pontos de corte, como mostrado nas figuras 25 e 26. Os descendentes de um processo de cruzamento têm características dos dois cromossomos pais. De acordo com [38] este operador é considerado como principal em um algorítmo genérico. A quantidade de indivíduos que participam deste processo é determinada pela taxa de cruzamento, que, por ser considerado operador principal deve ser maior do que a taxa de mutação.

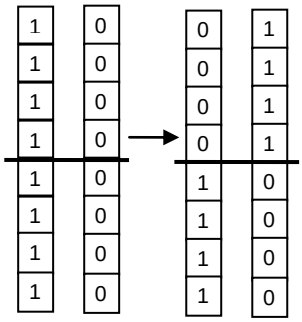

Figura 25 - Cruzamento em único ponto.

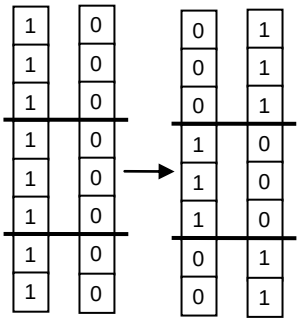

Figura 26 - Cruzamento em múltiplos pontos.

O operador cruzamento, aplicado ao protocolo J1939, pode gerar descendentes com mensagens duplicadas, de forma que após cada processo de reprodução os novos indivíduos são avaliados também em relação a esta característica. Uma das penalidades descritas na seção *avaliação e fitness* diz respeito exatamente a esta característica. A penalidade na nota de avaliação é acrescida em casos de genes (mensagens) duplicados. Desta forma, sua nota de avaliação não será uma das melhores, caracterizando que indivíduos com mensagens duplicadas não são solução para o problema e não poderão continuar o processo de evolução. A figura 27 mostra o cruzamento aplicado a dois cromossomos formados por mensagens SAE J1939.

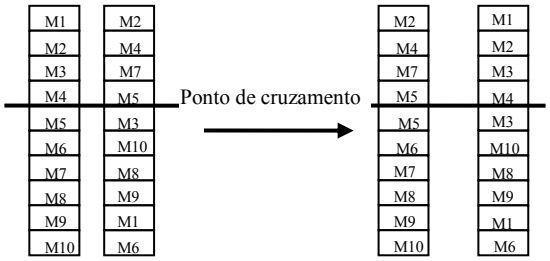

Figura 27 - Cruzamento em um cromossomo J1939.

4.6. Substituição

Existem estratégias definidas para substituição de uma geração por outra nova, recém criada. As estratégias mais utilizadas são:

Algoritmo genético simples, *Simple GA*: este algorítmo não usa a sobreposição de antigos indivíduos na geração nova. Isto significa que todos os indivíduos são substituídos pelos seus descendentes. Neste caso o algorítmo tente a convergir mais rapidamente, pois as melhores características dos melhores indivíduos são mantidas [41]. A correta taxa de cruzamento pode levar o algorítmo a criar

descendentes melhores do que os pais pela transmissão de material genético (Wall, 1996).

Algorítmo Genético de Estado Estacionário, *Steady State GA*: este algorítmo substitui a atual população sobrepondo alguns de seus atuais indivíduos. Isto significa que para cada geração uma parte dos indivíduos é substituída pelos seus descendentes. Neste caso, a convergência pode ser prematura e para uma solução boa, porém não ótima [41]. A taxa de cruzamento e de mutação pode evitar este comportamento.

4.7. Parâmetros de Algorítmos Genéticos

Os operadores de um algorítmo genético são aplicados com base em parâmetros estatísticos. Em [38] é apresentado um resumo sobre quatro parâmetros importantes, que serão utilizados nesta otimização. Estes parâmetros são utilizados no refinamento do algorítmo genético.

População: o número de indivíduos de uma população afeta a performance do algorítmo. Como descrito em [38], se a população é pequena a eficiência do algorítmo pode ser pequena, pois a cobertura ao espaço de busca, em cada geração, é pequena. Se a população é grande o espaço de busca é mais representativo, porém, demanda mais recursos computacionais.

Taxa de mutação/permutação: esta taxa determina o número de indivíduos de uma dada população que serão submetidos ao processo de mutação ou permutação. O processo de mutação/permutação permite uma maior distribuição das soluções no espaço de busca, entretanto, se a taxa for alta, a busca se torna excessivamente aleatória [38] [39].

Taxa de cruzamento: a taxa de cruzamento determina a probabilidade de um dado cromossomo ser submetido ao processo de cruzamento. De acordo com [38], quando maior a taxa, mais indivíduos com valores altos de fitness podem ser

perdidos em cada geração. Quando menor a taxa, maior o tempo dispendido para se chegar à soluções satisfatórias [39].

Número de gerações: este parâmetro define o número de iterações a ser executadas pelo algorítmo genético para se chegar à melhor solução [36]. Se o número de gerações for pequeno uma solução satisfatória pode não ser encontrada. Quanto maior o número de gerações, maior o tempo de execução.

4.8. Otimização aplicada ao protocolo J1939 – resumo.

No capítulo 3 o tempo de resposta de uma mensagem, no pior caso, foi apresentado sob as características do protocolo J1939, baseado na teoria descrita nos capítulos 2 e 3. Da mesma forma, baseado na teoria de algorítmos genéticos, algumas definições foram feitas em cima do conceito de rede CAN e protocolo J1939 levando-se em conta a característica do problema em questão.

Definição 1 – cada gene de um cromossomo é representado por uma mensagem da rede CAN.

Definição 2 - O alello representa a prioridade de cada mensagem, ou seja, a posição de cada mensagem na sequência de transmissão.

Definição 3 - O cromossomo foi representado por uma sequência de mensagens.

Definição 4 - A população foi representada por um grupo de sequência de mensagens, isto é, um grupo de chamados "*databases*".

Definição 5 - Os operadores utilizados são: cruzamento e permutação.

Definição 6 - A permutação foi chamada "*permutação seletiva*" e tem a característica de trocar valores de alellos de genes adjacentes se um gene de mais alta prioridade estiver posicionado depois do gene avaliado, na ordem de prioridade.

Definição 7 - O processo de cruzamento seguiu a implementação padrão com a utilização de um único ponto de troca para evitar um grande número de descendentes com genes duplicados.

Definição 8 - A função avaliação calcula o somatório do tempo entre o *WCRT* e o deadline de cada mensagem, sendo que, quanto maior o número, maior o fitness.

Definição 9 - As penalidades ou recompenses foram distribuídas conforme segue:

1 – foi determinada uma penalidade alta para cada perda de deadline.

2 - foi determinada uma penalidade alta para indivíduos com genes duplicados.

3 - foi determinada uma penalidade , de valor baixo, para cada gene que é precedido, na ordem de transmissão, por um gene com deadline maior.

4 - foi determinada uma recompensa, de valor alto, para cada gene precedido, na ordem de transmissão, por um gene com deadline menor.

Definição 10 - A função de aptidão foi implementada como o padrão: o valor do indivíduo em relação à média de sua geração.

Definição 11 - O processo de seleção descarta os piores 1/3 da população.

Definição 12 - A substituição de uma geração utiliza o algorítmo de estado estacionário, onde uma apenas uma parte da população é substituída pelos seus descendentes.

Definição 13 - Os melhores indivíduos de cada geração são mantidos.

5. SIMULAÇÃO E ANÁLISE DOS RESULTADOS.

5.1. Trabalhos Prévios.

Simulação de redes é uma disciplina importante, que faz parte de todo o desenvolvimento de arquitetura eletrônica embarcada. Existem no mercado softwares especializados em simulação de redes, onde seu comportamento é avaliado sob determinados aspectos e parâmetros. Alguns destes softwares são muito utilizados no desenvolvimento e análise de arquiteturas eletrônicas embarcadas, dentro da indústria automotiva. Cada um destes utiliza sua biblioteca de protocolos utilizados na indústria automotiva e seu modelo de rede, baseados, entre outras, nas teorias apresentadas nos capítulos iniciais deste trabalho.

Em [32] onde foi apresentado um projeto de rede CAN a simulação de uma condição real, em veículo, se deu com a implementação do hardware.

Uma ferramenta de simulação e otimização aplicada ao protocolo SAE J1939 tem dois objetivos claros. O primeiro é de simular o comportamento temporal do protocolo J1939 em uma rede CAN, baseado nas definições do capítulo 3. O segundo é de, dada uma situação de perda de deadline por parte de alguma mensagem, otimizar a sequência de transmissão de mensagens, ou seja , sua prioridade, de forma a evitar a perda de deadline. Para executar estas atividades a ferramenta de simulação e otimização deve permitir a inserção de cada mensagem de um dado *database*, executar uma função de cálculo de tempo de resposta no pior caso, executar as funções de algorítmos genéticos de forma a otimizar os resultados.

5.2. Implementação em Excel/VBA.

A primeira ferramenta desenvolvida foi em linguagem VBA dentro de uma macro do programa Excel. A razão desta escolha foi devido à existência de uma tabela de mensagens formatada no Excel formando a base de dados, ou *database*. Primeiramente, esta tabela contendo mensagens SAE J1939, dispostas em ordem decrescente de prioridade foi considerada. Em seguida os cálculos foram

implementados. Um exemplo de database é mostrado na tabela 3, onde as colunas à direita contém as características das mensagens: taxa de transmissão, número de bytes no campo de dados, deadline.

Nota-se que , para cada mensagem, foi considerado que o valor do deadline é igual ao valor da taxa de transmissão, ou seja, se uma mensagem é enviada e não chega a o destino é caracteridado perda de *deadline*, ou *timeout* pois o receptor não recebeu a informação dentro do tempo requerido, o *deadline*.

5.2.1. Caracterização do Problema.

A primeira função de simulação calcula o tempo de resposta no pior caso considerando os diferentes modelos apresentados no capítulo 6, baseados em [14], [17], [18], [31]. O algorítmo foi escrito em sequência contendo os seguintes cálculos, para uma determinada database:

Cálculo de Cm.

Cálculo de Bm.

Cálculo do Wm e WCRT de acordo com [14].

Cálculo do Wm e WCRT de acordo com [17].

Cálculo do Wm e WCRT de acordo com [18].

Cálculo do Wm e WCRT de acordo com [31].

Considerações gerais:

- velocidade da rede: 250kbps.
- tempo de bit: 4μs.
- Quantidade máxima de Stuffbits: 54
- Jitter: 0,2ms

O código para cada cálculo está apresentado no anexo I.

Primeiramente os cálculos foram aplicados sobre um database, ou seja, um conjunto de 31 mensagens padrão , retiradas do protocolo SAE J1939. Os resultados são mostrados na tabela 4.

Mensagem	*Taxa de transmissão [ms]*	*Nº de bytes do campo de dados*	*Deadline [ms]*
M1	10	8	10
M2	50	8	50
M3	10	8	10
M4	50	8	50
M5	20	8	20
M6	100	8	100
M7	20	8	20
M8	5000	8	5000
M9	1000	8	1000
M10	100	8	100
M11	1000	8	1000
M12	100	8	100
M13	1000	8	1000
M14	100	8	100
M15	100	8	100
M16	1000	8	1000
M17	1000	8	1000
M18	1000	8	1000
M19	250	8	250
M20	1000	8	1000
M21	1000	8	1000
M22	1000	8	1000
M23	1000	8	1000
M24	1000	8	1000
M25	100	8	100
M26	1000	8	1000
M27	500	8	500
M28	100	8	100
M29	100	8	100
M30	100	8	100
M31	1000	8	1000

Tabela 3 - Tabela base.

Menssagem	Deadline [ms]	Rm sem modelo de erros E(t) [ms]	Rm com E(t) (Tindell, 1995) [14] [ms]	Rm com E(t) (Pinho, 2000a) [17] [ms]	Rm com E(t) (Punnekkat, 2000) [18] [ms]	Rm (Davis, 2007) considerando E(t) de (Tindell, 1994d) [31] [ms]
M1	10	1,28	25,36	15,46	20,17	25,24
M2	50	1,92	26,00	16,12	20,44	25,88
M3	10	2,56	27,92	17,44	21,92	27,80
M4	50	3,20	28,56	18,10	22,24	28,44
M5	20	3,84	29,84	18,76	23,09	29,72
M6	100	4,48	31,76	19,42	23,26	30,36
M7	20	5,12	33,04	20,08	24,20	32,92
M8	5000	5,76	33,68	23,38	24,20	33,56
M9	1000	6,40	34,32	24,04	24,22	34,20
M10	100	7,04	34,96	24,70	24,42	34,84
M11	1000	7,68	35,60	25,36	24,44	35,48
M12	100	8,32	36,24	26,02	24,64	36,12
M13	1000	8,96	36,88	26,68	24,66	36,76
M14	100	9,60	37,52	27,34	24,86	37,40
M15	100	10,24	38,16	28,01	25,07	38,04
M16	1000	12,16	38,80	28,67	25,09	38,68
M17	1000	12,80	39,44	29,33	25,11	39,32
M18	1000	13,44	40,08	29,99	25,13	39,96
M19	250	14,08	43,28	31,97	25,22	40,60
M20	1000	14,72	43,92	32,63	25,24	43,80
M21	1000	15,36	44,56	33,29	25,26	44,44
M22	1000	16,00	45,20	33,95	25,28	45,08
M23	1000	16,64	45,84	34,61	25,30	45,72
M24	1000	17,28	46,48	35,27	25,32	46,36
M25	100	17,92	47,12	35,93	25,54	47,00
M26	1000	18,56	47,76	36,59	25,56	47,64
M27	500	19,20	48,40	37,25	25,60	48,28
M28	100	19,84	49,04	37,91	25,82	48,92
M29	100	23,04	49,68	38,57	26,05	49,56
M30	100	23,68	50,32	39,23	26,27	50,20
M31	1000	24,32	53,52	39,89	26,30	53,40

Tabela 4 - Tempo de resposta de mensagens SAE J1939 [34].

O parâmetro *Rm* é o tempo de resposta no pior caso de cada mensagem SAE J1939 distribuída nesta sequência de prioridades, onde a primeira mensagem da tabela tem a maior prioridade, seguido das subsequentes. A comparação entre o valor de *Rm* e o *deadline* de cada mensagem nos indica se o tempo de resposta está compatível com a necessidade do cliente da informação contida em seu campo de dados.

Como exemplo, o valor de R_m desconsiderando o fator erro, E(t), da primeira mensagem, M1, foi obtido seguindo as equações (3), (23), (6) e (7), do capítulo 3, respectivamente:

$$R_m = W_m + C_m$$

Onde:

$$C_m = \left(\left\lfloor \frac{54 + 8S_m - 1}{4} \right\rfloor + 67 + 8S_m \right) \tau_{bit} \quad ,$$

$$W_m^{n+1} = B_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{W_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k \quad , \text{e}$$

$$B_m = \max_{k \in lp(m)} (C_k)$$

Como S_1 = 8 bytes e τ_{bit} = 0,004[ms], pois a velocidade da rede é 250Kbps, a componente C_1 é calculada como segue:

$$C_1 = \left(\left\lfloor \frac{54 + 8 \cdot 8 - 1}{4} \right\rfloor + 67 + 8 \cdot 8 \right) \cdot 0{,}004 \therefore C_1 = \left(\left\lfloor \frac{118}{4} \right\rfloor + 131 \right) \cdot 0{,}004 \therefore$$

$$C_1 = (29 + 131) \cdot 0{,}004 = 160 \cdot 0{,}004 \therefore$$

$$C_1 = 0{,}64[ms]$$

Considerando J_k = 0,2[ms], $W_0 = C_1$ [34], e considerando que não há mensagens de maior prioridade do que M1, a componente W_m é calculada como segue:

$$W_1^{n+1} = B_1 + \sum_{\forall k \in hp(1)} \left\lceil \frac{W_1^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k = \max_{x \in lp(1)} (C_x) + 0 = C_1 \therefore$$

$$W_1^1 = 0{,}64[ms]$$

Neste exemplo, como se trata da mensagem de maior prioridade, o segunto termo da equação W_m é zero. Como na tabela base utilizada para este cálculo, tabela 3, todas as mensagens têm a mesma quantidade de bytes no campo de dados, o termo C_m se torna o mesmo para todas. Consequentemente o termo B_m da mensagem de maior prioridade pode ser considerado igual ao C_m desta. A equação de cálculo de W_m precisa de iterações até que o valor W_m^{n+1} tenha convergido. Neste trabalho a equação W_m foi submetida a 10 iterações, e este número de iterações foi suficiente para a convergência do valor de W_m .

Por fim o valor de Rm é calculado, para mensagem M1, como segue:

$$R_1 = W_1 + C_1 = 0{,}64 + 0{,}64 \therefore$$
$$R_1 = 1{,}28[ms]$$

Um outro exemplo, mostrando as iterações de Wm é o cálculo de Rm para a mensagem M2, considerando 8 bytes no campo de dados:

$$C_2 = \left(\left\lfloor \frac{54 + 8 \cdot 8 - 1}{4} \right\rfloor + 67 + 8 \cdot 8 \right) \cdot 0{,}004 \therefore C_2 = \left(\left\lfloor \frac{118}{4} \right\rfloor + 131 \right) \cdot 0{,}004 \therefore$$
$$C_2 = (29 + 131) \cdot 0{,}004 = 160 \cdot 0{,}004 \therefore$$
$$C_2 = 0{,}64[ms]$$

Tempo de fila:

$$W_2^{n+1} = B_2 + \sum_{\forall k \in hp(2)} \left\lceil \frac{W_2^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k$$

Como só existe 1 mensagem de maior prioridade do que M2 a somatória só terá uma componente:

$$W_2^1 = B_2 + \left\lceil \frac{W_2^0 + J_1 + \tau_{bit}}{T_1} \right\rceil C_1 = 0{,}64 + \left\lceil \frac{0{,}64 + 0{,}2 + 0{,}004}{10} \right\rceil \cdot 0{,}64 = 1{,}28$$
$$W_2^2 = B_2 + \left\lceil \frac{W_2^1 + J_1 + \tau_{bit}}{T_1} \right\rceil C_1 = 0{,}64 + \left\lceil \frac{1{,}28 + 0{,}2 + 0{,}004}{10} \right\rceil \cdot 0{,}64 = 1{,}28[ms]$$

Neste exemplo, não seria mais necessário outra iteração de W_m pois o valor já convergiu para $W_2 = 1{,}28$[ms]. Então R_m fica:

$$R_2 = W_2 + C_2 = 1{,}28 + 0{,}64 \therefore$$
$$R_2 = 1{,}92\,[ms]$$

Pode-se verificar que alguns modelos são mais pessimistas do que outros devido aos diferentes parâmetros considerados por cada um. Na simulação apresentada pela tabela 4, as mensagens com taxa de transmissão mais altas e com prioridades maiores perderam seu deadline em quase todos modelos. Isto aconteceu devido ao fato de que foram considerados erros no processo de comunicação. Os modelos de erros aumentam o tempo de resposta de uma mensagem consideravelmente. A tendência é igual em todos os modelos: as mensagens menos prioritárias têm um tempo de resposta maior, no pior caso.

A segunda simulação foi feita acrescentando 20 mensagens proprietárias, iniciadas com a letra *P* , à base de dados. Mensagens proprietárias são aquelas, como já foi dito no capítulo 2, que podem ser acrescidas a um conjunto de mensagens J1939, em caso de necessidade e utilizando identificadores específicos e definidos para este fim. O método e parâmetros de cálculo foram os mesmos da simulação anterior. Os resultados são mostrados na tabela 5. Nesta simulação algumas mensagens proprietárias, de forma proposital, receberam valores altos para a taxa de transmissão. A consequência foi a perda de deadline, o que está identificado na tabela com fundo mais escuro. Percebe-se que para uma base de dados com um número grande de mensagens, a inserção de mensagens proprietárias deve ser feita analisando a mínima taxa de transmissão permitida de forma a salvaguardar os requisitos temporais da rede. Dependendo do tamanho do *database*, a inserção de mensagens proprietárias com taxas de transmissão de 50 ou até mesmo 100ms se torna não recomendável, devido ao risco de que, no pior caso, seus requisitos temporais não sejam atendidos. Está , portanto, caracterizado o problema.

5.2.2. Otimização.

Uma vez caracterizado o problema, ou seja, no pior caso, a ocorrência da perda de deadline de um dado número de mensagens, foi implementado o método escolhido para otimização: Algorítmos Genéticos. O algoritmo principal foi baseado nos conceitos apresentados no capítulo 4 e é descrito abaixo:

For i = 0 to *NumeroGerações*
    Avaliação()
    Fitness()
    Reprodução()
    Substituição()
End

O método *avaliação()* utiliza o cálculo de tempo de resposta no pior caso acrescido das penalidades e recompensas, conforme descrito no capítulo 4 para avaliar a população.

O método *Fitness()* ordena a população e seleciona os melhores.

O método *Reprodução()* contém as funções de permutação e cruzamento.

O método *Sunstituição()* substitui a população anterior pelos descendentes.

O código para cada método está apresentado no anexo I.

Utilizando um grupo de 31 mensagens padrão SAE J1939 mais 20 mensagens proprietárias, o algorítmo foi implementado conforme tabelas 7, 8 e 9. Um grande desafio dos algorítmos genéticos é evitar que haja a convergênica precoce para um máximo bom mas não ótimo. Para tal, a simulação foi repetida algumas vezes considerando variações nos parâmetros número de gerações, taxa de permutação, taxa de cruzamento. A métrica utilizada para avaliar uma simulação foi a média da função de avaliação de cada geração. Esta informação é mostrada nos gráficos logo ao lado das tabelas 7, 8 e 9.

Menssagem	Deadline [ms]	Rm sem modelo de erros E(t) [ms]	Rm com E(t) (Tindell, 1995) [14] [ms]	Rm com E(t) (Pinho, 2000a) [17] [ms]	Rm com E(t) (Punnekkat, 2000) [18] [ms]	Rm (Davis, 2007) [31] considerando E(t) de (Tindell, 1994d) [14] [ms]
M1	10	1,28	25,36	15,46	20,17	25,24
M2	50	1,92	26,00	16,12	20,44	25,88
M3	10	2,56	27,92	17,44	21,92	27,80
M4	50	3,20	28,56	18,10	22,24	28,44
M5	20	3,84	29,84	18,76	23,09	29,72
M6	100	4,48	31,76	19,42	23,26	30,36
M7	20	5,12	33,04	20,08	24,20	32,92
M8	5000	5,76	33,68	23,38	24,20	33,56
M9	1000	6,40	34,32	24,04	24,22	34,20
M10	100	7,04	34,96	24,70	24,42	34,84
M11	1000	7,68	35,60	25,36	24,44	35,48
M12	100	8,32	36,24	26,02	24,64	36,12
M13	1000	8,96	36,88	26,68	24,66	36,76
M14	100	9,60	37,52	27,34	24,86	37,40
M15	100	10,24	38,16	28,01	25,07	38,04
M16	1000	12,16	38,80	28,67	25,09	38,68
M17	1000	12,80	39,44	29,33	25,11	39,32
M18	1000	13,44	40,08	29,99	25,13	39,96
M19	250	14,08	43,28	31,97	25,22	40,60
M20	1000	14,72	43,92	32,63	25,24	43,80
M21	1000	15,36	44,56	33,29	25,26	44,44
M22	1000	16,00	45,20	33,95	25,28	45,08
M23	1000	16,64	45,84	34,61	25,30	45,72
M24	1000	17,28	46,48	35,27	25,32	46,36
M25	100	17,92	47,12	35,93	25,54	47,00
M26	1000	18,56	47,76	36,59	25,56	47,64
M27	500	19,20	48,40	37,25	25,60	48,28
M28	100	19,84	49,04	37,91	25,82	48,92
M29	100	23,04	49,68	38,57	26,05	49,56
M30	100	23,68	50,32	39,23	26,27	50,20
M31	1000	24,32	53,52	39,89	26,30	53,40
P32	10	24,96	57,36	45,84	28,83	57,24
P33	10	26,88	65,04	49,14	31,92	64,92
P34	10	28,80	69,52	57,07	35,76	69,40
P35	10	33,92	77,84	66,31	40,68	77,72
P36	1000	36,48	78,48	66,97	40,74	83,48
P37	500	37,12	79,12	67,64	40,86	84,12
P38	50	37,76	80,40	68,96	42,02	85,40
P39	100	38,40	86,16	69,62	42,63	86,04
P40	100	39,04	86,80	70,28	43,25	86,68
P41	50	39,68	88,08	75,56	44,56	87,96
P42	100	40,32	88,72	76,22	45,25	88,60
P43	100	46,08	89,36	76,88	45,96	89,24
P44	100	46,72	90,00	77,54	46,69	89,88
P45	100	47,36	94,48	78,20	47,44	90,52
P46	50	48,00	95,76	79,52	49,02	95,64
P47	100	48,64	96,40	80,18	49,86	96,28
P48	100	49,28	97,04	86,13	50,72	96,92
P49	10	49,92	138,00	96,70	61,08	137,88
P50	10	66,56	168,08	148,22	76,67	167,96
P51	1000	76,16	168,40	148,55	76,64	173,72

Tabela 5 - Tempo de resposta considerando mensagens proprietárias [34].

Primeiramente, a tabela 6 mostra uma simulação com 40% de taxa de permutação, 60% de taxa de cruzamento, 50 gerações de uma população com 50 indivíduos, e apenas duas penalidades: 1 – para cada deadline perdido, 2 – para genes duplicados.

Neste caso não houve modificações significativas nas características temporais deste database. Os problemas temporais identificados não foram eliminados. Em alguns casos, por exemplo, a mensagem P35, o tempo de resposta no pior caso foi reduzido, mas não o bastante para eliminar a perda de deadline.

Na simulação mostrada na tabela 7, os parâmetros do algorítmo foram modificados conforme segue: taxa de cruzamento de 60%, taxa de mutação de 30%, 150 gerações de uma população de 150 indivíduos. Além disto, uma penalidade e uma recompensa foram acrescidas: precedência e antecedência na fila de prioridades, de acordo com apresentado no capítulo 4.8, definição 9, itens 3 e 4.

Mensagem	Deadline [ms]	Rm sem função erro E(t) [ms]	Mensagem	Deadline [ms]	Rm sem função erro E(t) [ms]
Inicial			Otimizado		
M1	10	1,48	M1	10	1,48
M2	50	2,12	M2	50	2,12
M3	10	2,76	M10	100	2,76
M4	50	3,40	M3	10	3,4
M5	20	4,04	M4	50	4,04
M6	100	4,68	M6	100	4,68
M7	20	5,32	P42	100	5,32
M8	5000	5,96	M11	1000	5,96
M9	1000	6,60	M7	20	6,6
M10	100	7,24	M12	100	7,24
M11	1000	7,88	M13	1000	7,88
M12	100	8,52	M13	1000	8,52
M13	1000	9,16	M14	100	9,16
M14	100	9,80	M9	1000	9,8
M15	100	10,44	M15	100	10,44
M16	1000	12,36	M16	1000	12,36
M17	1000	13,00	M17	1000	13
M18	1000	13,64	M18	1000	13,64
M19	250	14,28	M19	250	14,28
M20	1000	14,92	M20	1000	14,92
M21	1000	15,56	M21	1000	15,56
M22	1000	16,20	M22	1000	16,2
M23	1000	16,84	M23	1000	16,84
M24	1000	17,48	M24	1000	17,48
M25	100	18,12	M25	100	18,12
M26	1000	18,76	M26	1000	18,76
M27	500	19,40	M27	500	19,4
M28	100	20,04	M28	100	20,04
M29	100	23,24	M29	100	22,6
M30	100	23,88	M31	1000	23,24
M31	1000	24,52	M8	5000	23,88
P32	10	25,16	P32	10	24,52
P33	10	27,08	P33	10	26,44
P34	10	29,00	P34	10	28,36
P35	10	34,12	P35	10	30,28
P36	1000	36,68	P36	1000	36,04
P37	500	37,32	P37	500	36,68
P38	50	37,96	P38	50	37,32
P39	100	38,60	P39	100	37,96
P40	100	39,24	M18	1000	38,6
P41	50	39,88	P41	50	39,24
P42	100	40,52	P42	100	39,88
P43	100	46,28	P43	100	40,52
P44	100	46,92	P44	100	45,64
P45	100	47,56	P35	10	46,28
P46	50	48,20	P46	50	49,48
P47	100	48,84	P47	100	50,12
P48	100	49,48	P48	100	58,44
P49	10	50,12	P49	10	59,08
P50	10	66,76	P50	10	68,68
P51	1000	76,36	P51	1000	78,92

Tabela 6 - 1ª simulação aplicada ao J1939.

Inicial			Otimizado		
Mensagem	Deadline [ms]	Rm sem função erro E(t) [ms]	Mensagem	Deadline [ms]	Rm sem função erro E(t) [ms]
M1	10	1,48	M1	10	1,48
M2	50	2,12	M12	100	2,12
M3	10	2,76	M3	10	2,76
M4	50	3,40	M5	20	3,4
M5	20	4,04	M28	100	4,04
M6	100	4,68	M6	100	4,68
M7	20	5,32	M9	1000	5,32
M8	5000	5,96	M8	5000	5,96
M9	1000	6,60	M10	100	6,6
M10	100	7,24	P37	500	7,24
M11	1000	7,88	M11	1000	7,88
M12	100	8,52	M2	50	8,52
M13	1000	9,16	M13	1000	9,16
M14	100	9,80	P33	20	9,8
M15	100	10,44	M15	100	10,44
M16	1000	12,36	M16	1000	12,36
M17	1000	13,00	P48	100	13
M18	1000	13,64	M18	1000	13,64
M19	50	14,28	M19	50	14,28
M20	1000	14,92	P40	100	14,92
M21	1000	15,56	M20	1000	15,56
M22	1000	16,20	M22	1000	16,2
M23	1000	16,84	P45	100	16,84
M24	1000	17,48	M24	1000	17,48
M25	100	18,12	M25	100	18,12
M26	1000	18,76	M26	1000	18,76
M27	500	19,40	M7	20	19,4
M28	100	20,04	P50	30	20,04
M29	100	23,24	M29	100	23,88
M30	100	23,88	P35	20	24,52
M31	1000	24,52	M14	100	25,8
P32	20	25,16	M31	1000	26,44
P33	20	26,44	P32	20	27,08
P34	20	27,72	P49	30	28,36
P35	20	29,00	M4	50	29
P36	1000	30,28	M30	100	29,64
P37	500	32,20	P36	1000	30,28
P38	50	32,84	P38	50	33,48
P39	100	33,48	P39	100	34,12
P40	100	34,12	M21	1000	34,76
P41	50	34,76	P34	20	35,4
P42	100	35,40	P41	50	36,68
P43	100	36,04	P42	100	37,32
P44	100	36,68	P44	100	37,96
P45	100	37,32	M23	1000	38,6
P46	50	37,96	P46	50	39,24
P47	100	38,60	P47	100	39,88
P48	100	39,24	M17	1000	40,52
P49	30	39,88	P43	100	46,28
P50	30	46,28	M27	500	46,92
P51	1000	47,56	P51	1000	47,56

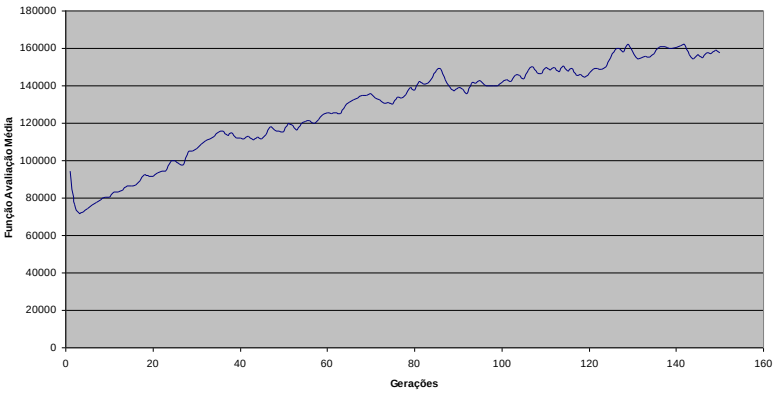

Figura 28 - Evolução da média das notas de avaliação.

Tabela 7 - Simulação com mudanças nos parâmetros.

Nesta simulação, mostrada na tabela 7, percebe-se evolução significativa nos resultados , onde 50% dos problemas temporais foram resolvidos. Porém esta não é a solução ótima para o problema, pois ainda existem mensagens com perda de deadline. A evolução do algorítmo é caracterizada pela figura 27, onde é mostrado a tendência da média dos valores de avaliação.

De acordo com [36] e [38] a taxa de mutação não deve ser alta para que o algorítmo não se torne excessivamente aleatório. Para evitar tal comportamento a simulação a seguir foi feita com taxa de cruzamento de 70%, taxa de permutação de 5%, 200 gerações de uma população de 200 indivíduos. Os resultados são mostrados na tabela 8.

Mensagem	Deadline [ms]	Rm sem função erro E(t) [ms]	Mensagem	Deadline [ms]	Rm sem função erro E(t) [ms]
Inicial			Otimizado		
M1	10	1,48	P34	20	1,48
M2	50	2,12	M2	50	2,12
M3	10	2,76	M29	100	2,76
M4	50	3,40	M16	1000	3,4
M5	20	4,04	M5	20	4,04
M6	100	4,68	M6	100	4,68
M7	20	5,32	M9	1000	5,32
M8	5000	5,96	M8	5000	5,96
M9	1000	6,60	M7	20	6,6
M10	100	7,24	M10	100	7,24
M11	1000	7,88	M11	1000	7,88
M12	100	8,52	M12	100	8,52
M13	1000	9,16	M13	1000	9,16
M14	100	9,80	P33	20	9,8
M15	100	10,44	M15	100	10,44
M16	1000	12,36	M27	500	11,08
M17	1000	13,00	M18	1000	11,72
M18	1000	13,64	P49	30	12,36
M19	50	14,28	M19	50	13
M20	1000	14,92	M14	100	13,64
M21	1000	15,56	M23	1000	14,28
M22	1000	16,20	P43	100	14,92
M23	1000	16,84	P45	100	15,56
M24	1000	17,48	M26	1000	16,2
M25	100	18,12	M3	10	16,84
M26	1000	18,76	P50	30	18,12
M27	500	19,40	M4	50	18,76
M28	100	20,04	M25	100	19,4
M29	100	23,24	M21	1000	20,04
M30	100	23,88	M30	100	23,88
M31	1000	24,52	M31	1000	24,52
P32	20	25,16	P44	100	25,16
P33	20	26,44	M24	1000	25,8
P34	20	27,72	M1	10	26,44
P35	20	29,00	P35	20	28,36
P36	1000	30,28	P39	100	29,64
P37	500	32,20	P37	500	30,28
P38	50	32,84	P36	1000	33,48
P39	100	33,48	P38	50	34,12
P40	100	34,12	P40	100	34,76
P41	50	34,76	M22	1000	35,4
P42	100	35,40	P41	50	36,04
P43	100	36,04	P42	100	36,68
P44	100	36,68	M20	1000	37,32
P45	100	37,32	P32	20	37,96
P46	50	37,96	P46	50	39,24
P47	100	38,60	P47	100	39,88
P48	100	39,24	P48	100	40,52
P49	30	39,88	M17	1000	46,28
P50	30	46,28	M28	100	46,92
P51	1000	47,56	P51	1000	47,56

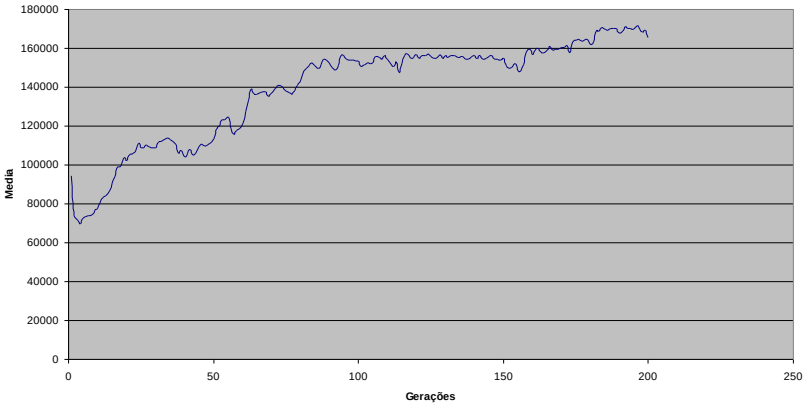

Figura 29 - Evolução da média das notas de avaliação.

Tabela 8 - Simulação reduzindo a taxa de permutação.

Nesta simulação, a convergência da função de avaliação foi mais rápida, porém mais uma vez para um resultado não ótimo, pois ainda existem mensagens com problemas temporais.

Uma outra simulação foi feita com os seguintes parâmetros: taxa de cruzamento de 80%, taxa de permutação de 3%, 250 gerações de uma população de 250 indivíduos. Os resultados são mostrados na tabela 9 e na figura 29.

Mensagem	Deadline [ms]	Rm sem função erro E(t) [ms]	Mensagem	Deadline [ms]	Rm sem função erro E(t) [ms]
Inicial			Otimizado		
M1	10	1,48	P34	20	1,48
M2	50	2,12	M2	50	2,12
M3	10	2,76	M29	100	2,76
M4	50	3,40	M16	1000	3,4
M5	20	4,04	M5	20	4,04
M6	100	4,68	M6	100	4,68
M7	20	5,32	M9	1000	5,32
M8	5000	5,96	M8	5000	5,96
M9	1000	6,60	M7	20	6,6
M10	100	7,24	M10	100	7,24
M11	1000	7,88	M11	1000	7,88
M12	100	8,52	M12	100	8,52
M13	1000	9,16	M13	1000	9,16
M14	100	9,80	P33	20	9,8
M15	100	10,44	M15	100	10,44
M16	1000	12,36	M27	500	11,08
M17	1000	13,00	M18	1000	11,72
M18	1000	13,64	P49	30	12,36
M19	50	14,28	M19	50	13
M20	1000	14,92	M14	100	13,64
M21	1000	15,56	M23	1000	14,28
M22	1000	16,20	P43	100	14,92
M23	1000	16,84	P45	100	15,56
M24	1000	17,48	M26	1000	16,2
M25	100	18,12	M3	10	16,84
M26	1000	18,76	P50	30	18,12
M27	500	19,40	M4	50	18,76
M28	100	20,04	M25	100	19,4
M29	100	23,24	M21	1000	20,04
M30	100	23,88	M30	100	23,88
M31	1000	24,52	M31	1000	24,52
P32	20	25,16	P44	100	25,16
P33	20	26,44	M24	1000	25,8
P34	20	27,72	M1	10	26,44
P35	20	29,00	P35	20	28,36
P36	1000	30,28	P39	100	29,64
P37	500	32,20	P37	500	30,28
P38	50	32,84	P36	1000	33,48
P39	100	33,48	P38	50	34,12
P40	100	34,12	P40	100	34,76
P41	50	34,76	M22	1000	35,4
P42	100	35,40	P41	50	36,04
P43	100	36,04	P42	100	36,68
P44	100	36,68	M20	1000	37,32
P45	100	37,32	P32	20	37,96
P46	50	37,96	P46	50	39,24
P47	100	38,60	P47	100	39,88
P48	100	39,24	P48	100	40,52
P49	30	39,88	M17	1000	46,28
P50	30	46,28	M28	100	46,92
P51	1000	47,56	P51	1000	47,56

Tabela 9 - Simulação com permutação 3% e 250 gerações.

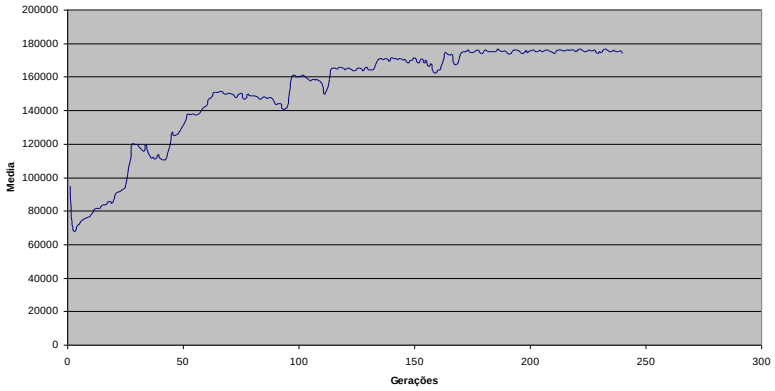

Figura 30 - Evolução da média das notas de avaliação.

Nesta simulação a convergência se acentuou em relação à anterior, como pode ser visto na figura 29, porém ainda não é uma solução ótima, pois restam mensagens com perda de deadline. A implementação de algorítmos genéticos em macros no Excel tem suas limitações de tempo de processamento, devido às leituras e atualizações constantes das células envolvidas. Foi decidido migrar a implementação para um ambiente orientado a objeto de forma a possibilitar maior capacidade de processamento focado nos métodos do algorítmo. Todos os resultados acima foram publicados em [45].

5.3. Implementação em C#.

O ambiente C# foi escolhido e a plataforma VisualStudio foi utilizada para a programação. A biblioteca Galib [40] foi utilizada como referência para o programa. O programa consiste em uma tela onde, inicialmente, o usuário pode inserir a base de dados inicial. Esta base de dados inicial é a que se deseja avaliar e otimizar, caso necessário. O arquivo precisa estar em formato txt e com os parâmetros das mensagens separados por ponto e vírgula. Após a seleção do arquivo via menu *arquivo*, *abrir*, *database*, o usuário deve inserir as informações como o database inicial via botão *inserir*.

Os parâmetros de cada mensagem a serem inseridos são: numero da mensagem, nome da mensagem, taxa de transmissão, deadline, número de bytes no campo de dados. Deve-se inserir os parâmetros da CAN: número máximo de stuffbits e o tempo de bit.

Após inserção das mensagens na sequência de prioridade, da mais alta para a mais baixa, e também dos parâmetros da CAN, o usuário solicita a avaliação do sistema. O valor do tempo de resposta no pior caso é mostrado logo após o número de prioridade, o nome e o deadline de cada mensagem.

Em seguida o usuário insere os parâmetros do algorítmo genético: taxa de cruzamento, taxa de permutação, número da população e número de gerações, e, então solicita a otimização via botão *otimizar*.

O programa está dividido em três classes como segue:

Classe Form1 - formulário principal , onde está contido o algorítmo principal:

CriaPopInicial()

For i = 0 to *NumeroGerações*
Avaliação()
Fitness()
Seleção()
Permutação()
Cruzamento()
Substituição()
End

Classe Gene – onde são criadas as mensagens inseridas na tela principal. Contém o costrutor para a estrutura Gene:

Gene()

Classe Database – classe derivada da classe CollectionBase, onde é criado o database inicial e também calculado o tempo de resposta no pior caso ao selecionar o botão *avaliar*. Contém os métodos:

Add()
CalculoWCRT()

Classe Cálculos – Onde é criada a população inicial, baseada no database inicial. Contém os métodos de calculo do tempo de bloqueio, *Bm*, tempo de transmissão, *Cm*, e também o calculo do tempo de resposta no pior caso. Onde são criadas as gerações, formadas por N databases, dependendo do número da população escolhido. Contém os métodos de calculo da função avaliação, do fitness, além dos métodos de seleção e substituição. Onde são executados os métodos relativos à reprodução

WCRT()
Avaliação()
Fitness()
Seleção()
Substituição()
CriaPopInicial()
Permutação()
Cruzamento()

Durante a execução do algorítmo principal a função avaliação, chamada da classe Cálculos, chama os cálculos de Bm e WCRT, que por sua vez, chama o calculo de Cm para cada mensagem. Após o calculo de WCRT o valor de avaliação é calculado. O valor de avaliação é característica de cada database, ou seja, de cada indivíduo de uma geração. Após os cálculos, as penalidades e recompensas são executadas, modificando o valor de avaliação. Foi implementado uma penalidade alta para o caso de se ter genes duplicados, uma penalidade baixa para o caso de um gene ser precedido de outro com deadline maior, e uma recompensa baixa para o caso de se ter um gene precedido por outro com deadline menor. Em cima destes resultados o fitness é calculado, posicionando cada database em forma de ranking. Baseado neste ranking, 66% dos melhores indivíduos são selecionados para a reprodução. Estes melhores indivíduos são submetidos aos métodos de reprodução, gerando 66% dos indivíduos da nova população, sendo que o número de indivíduos de uma população é um parâmetro fixo e não muda a cada geração. Os outros 33% são preenchidos pelos melhores indivíduos da geração anterior, caracterizando a aplicação do algorítmo do tipo estado estacionário. O resultado, após N gerações, é mostrado em um listbox como a sequência ótima de prioridade de mensagens, em relação ao database inicial. A evolução da média das notas de avaliação também é mostrada catacterizando a otimização proporcionada pelo algorítmo.

A figura 31 mostra a tela do software de simulação.

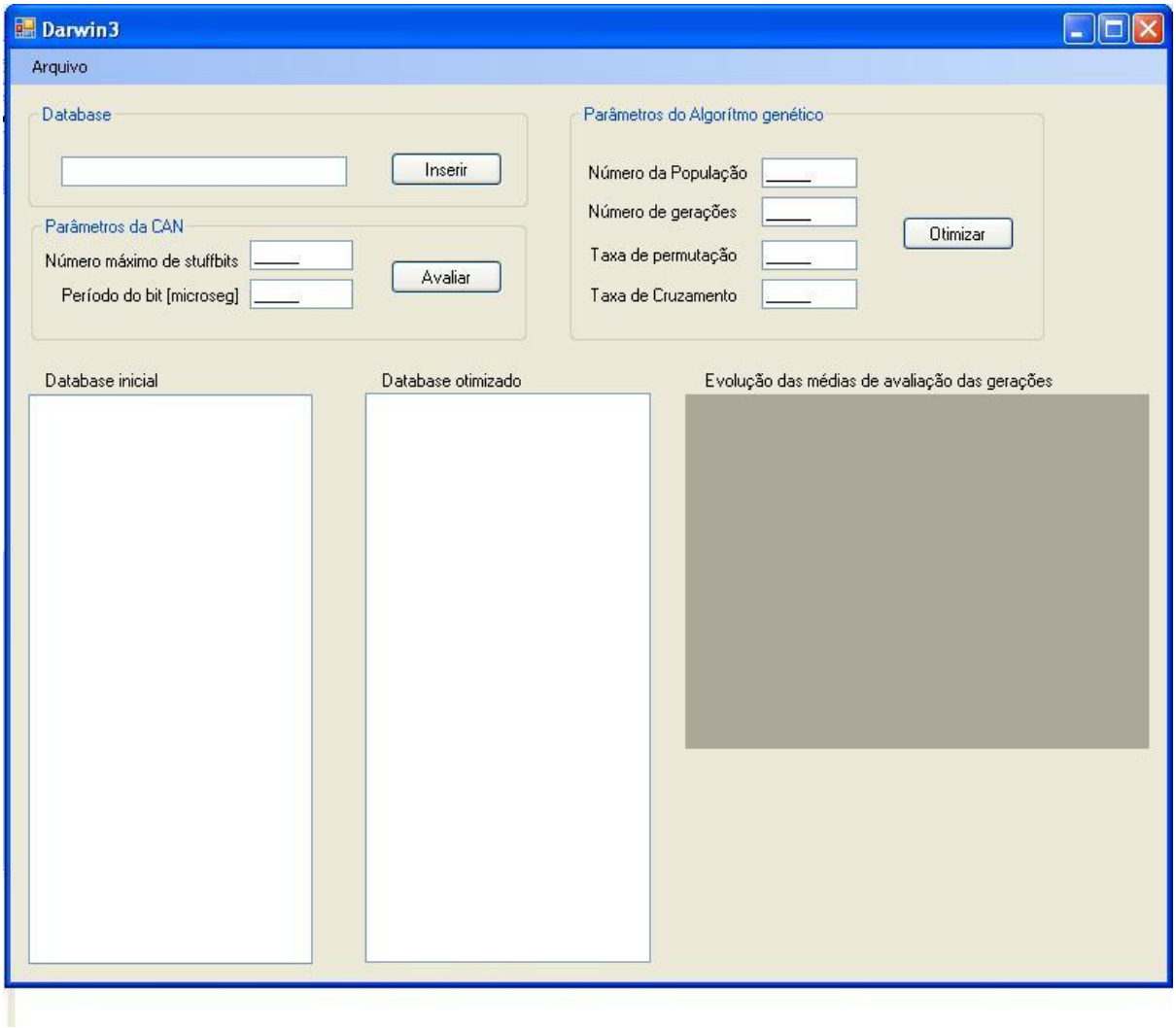

figura 31 – Tela inicial.

A figura 32 mostra um exemplo de inserção de um database inicial e em seguida com os parâmetros da CAN. Nesta simulação o database inicial foi gerado tomando 31 mensagens do protocolo J1939 acrescidas de 20 mensagens proprietárias criadas para exemplificar a situação de perda de *deadline*.

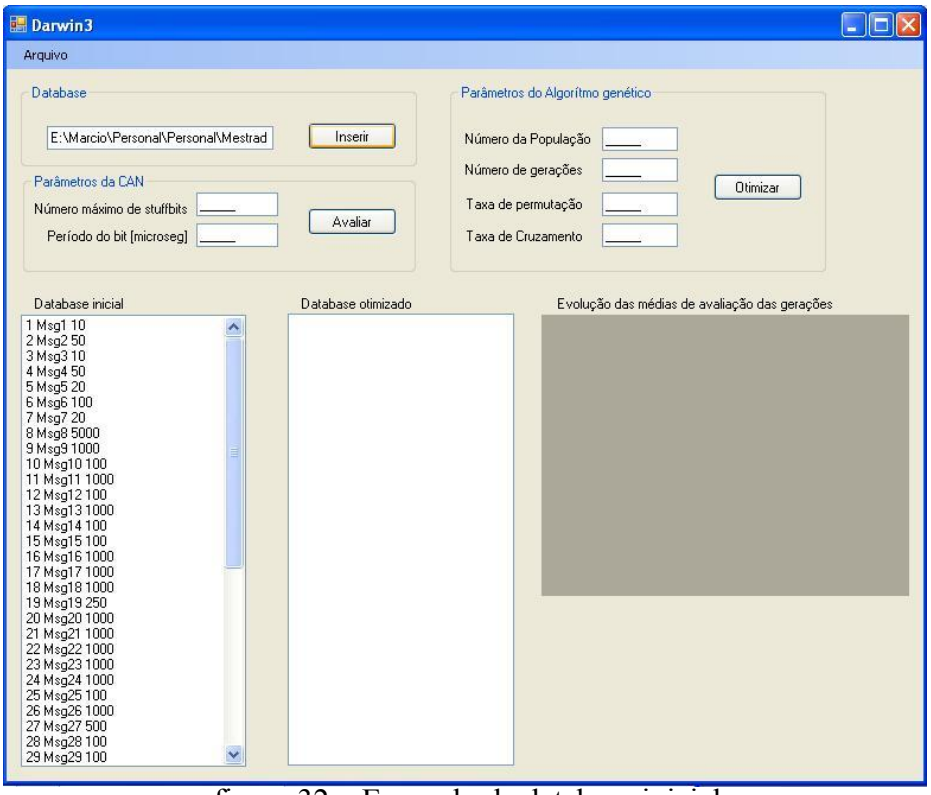

figura 32 – Exemplo de database inicial.

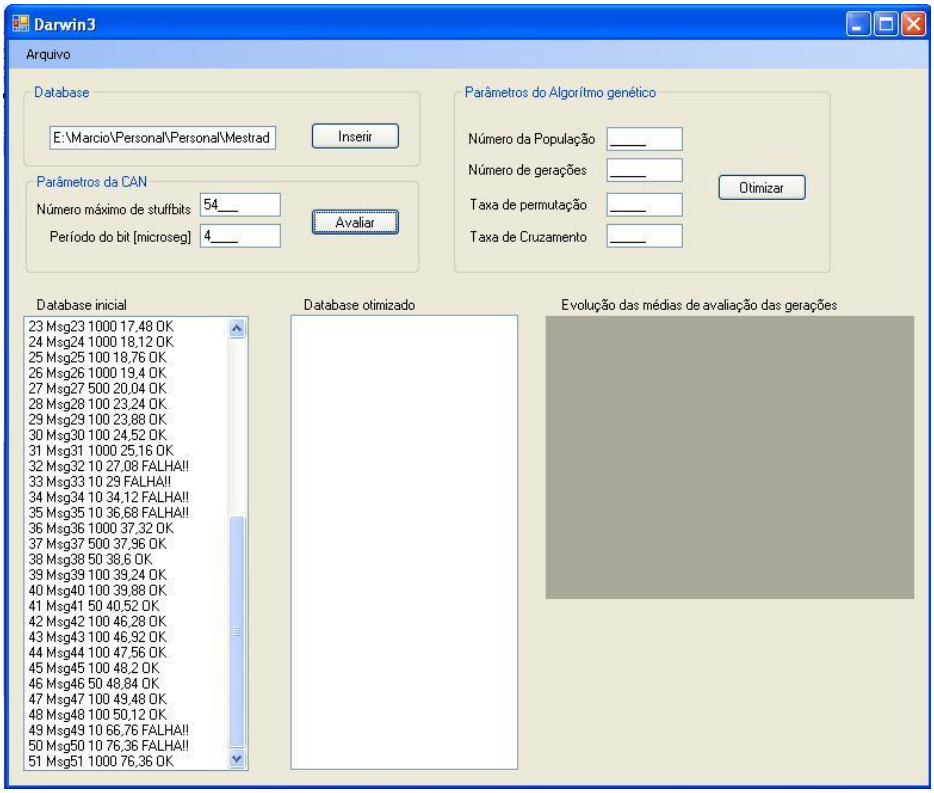

figura 33 – Exemplo de inserção dos parâmetros da CAN e avaliação.

Ao pressionar o botão avaliar, o cálculo de tempo de resposta no pior caso é feito sobre a base de dados inicial. As mensagens cujo tempo de resposta é maior do que seu *deadline* têm a indicação *"FALHA!!"*.

A figura 34 mostra um exemplo de inserção de parâmetros do algorítmo genético, considerando a taxa de permutação 10%, taxa de cruzamento 70%, 300 indivíduos em 300 gerações. O resultado não convergiu para o ótimo porém foi verificado um aumento da média das notas de avaliação, o que denota a evolução proporcionada pelo algorítmo.

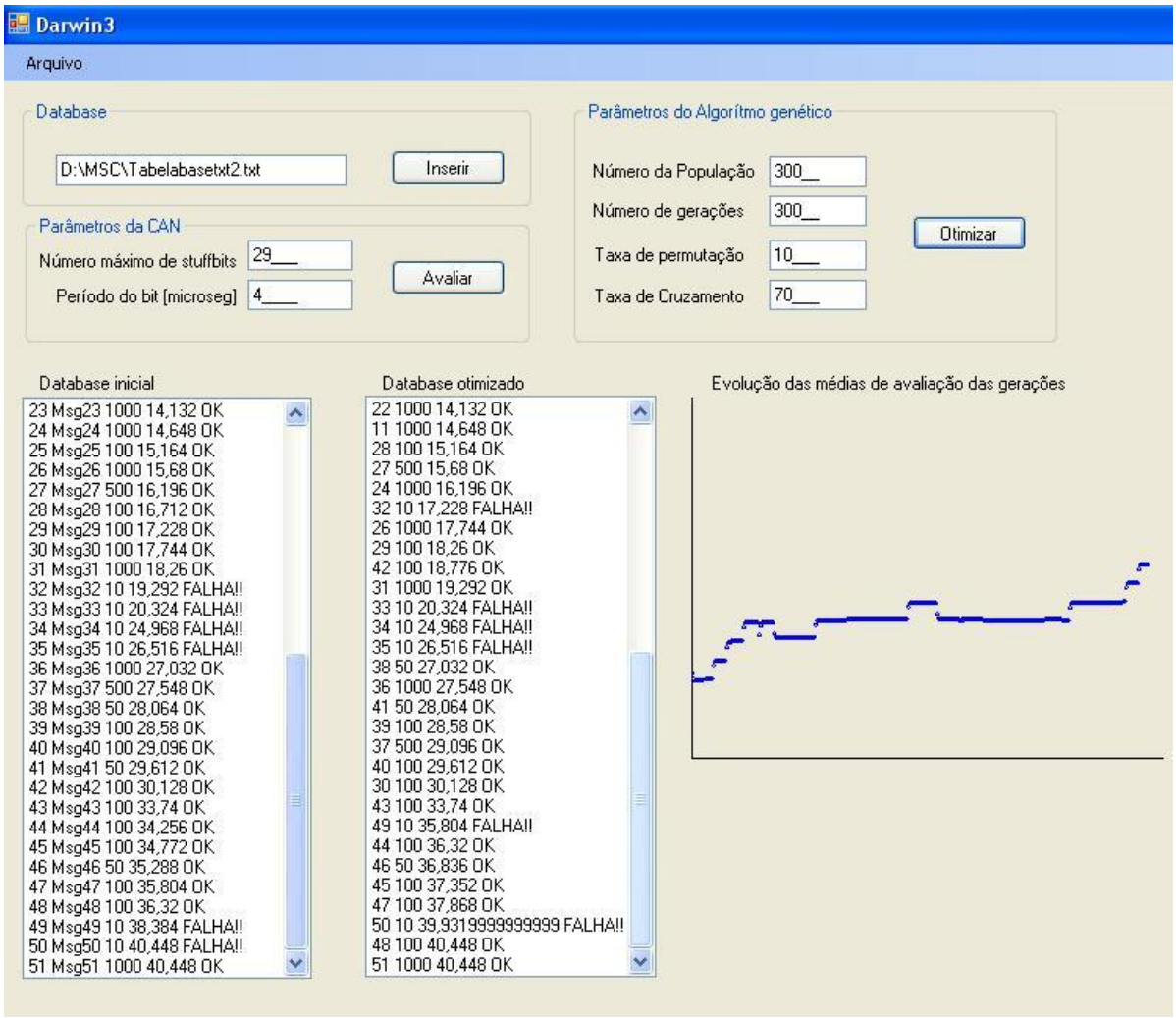

Figura 34 – Parâmetros do Algorítmo Genético e otimização.

6. CONCLUSÕES

6.1. Considerações gerais.

A aplicação das métricas estudadas na análise temporal do protocolo J1939 pode ajudar os desenvolvedores de redes embarcadas a analisar e validar a distribuição de mensagens, bem como, as definições de prioridades e taxas de transmissão. No início dos projetos uma análise inicial sobre a arquitetura de rede se faz necessária para que possam ser definidas as suas condições de contorno. Uma simulação do comportamento temporal da rede pode mostrar situações críticas, erros de perda de deadline e , mais importante, permitir aos desenvolvedores testar diferentes configurações, caso necessário. Este estudo é feito em novos projetos, como em [26], onde o modelo proposto em [12] foi utilizado para calcular o tempo de resposta das mensagens criadas para um veículo não tripulado. Além disto, durante a vida de um projeto, muita vezes se faz necessário acrescentar novas tecnologias, por força de legislação ou força de mercado, afetando uma dada arquitetura de rede e a quantidade de mensagens a serem transmitidas. Nestes casos a utilização de uma ferramenta de simulação também se faz necessária, por permitir a análise prévia do efeito da inserção de grupos de mensagens em uma determinada matriz de mensagens.

Dado um grupo de 31 mensagens conhecidas, provenientes da norma SAE J1939, acrescidas de 20 mensagens proprietárias definidas apenas para efeito de caracterização do problema discutido neste trabalho, o tempo de resposta no pior caso foi calculado utilizando diferentes modelos. Os resultados são diferentes para cada modelo. Os modelos mais recentes tendem a ser mais pessimistas, pois abrangem mais detalhes do comportamento real do sistema. Apesar dos diferentes resultados, foi percebida a mesma tendência em todos os modelos. Isto significa que para uma mensagem com perda de *deadline*, foi percebido que na maioria dos modelos foi detectado o problema. A utilização de um ou outro modelo fica a cargo do desenvolvedor e do risco inerente à utilização de um modelo mais simples ou mais complexo.

O comportamento temporal da rede, calculado em cima de modelos matemáticos precisa ser comparado com o comportamento real do sistema. Em

[28] um experimento foi proposto e o comportamento do barramento sob condições de erro foi simulado e medido em bancada.

Uma das propostas deste trabalho foi de analisar os modelos existentes em relação ao cálculo do tempo de resposta no pior caso, aplicado a um barramento CAN, com protocolo SAE J1939. Os cálculos foram feitos usando um software de simulação desenvolvido em VBA. Os resultados serão utilizados para validar redes embarcadas em situações reais na indústria automotiva, durante a fase de conceituação dos produtos.

Uma vez que a análise temporal de uma rede tenha sido feita e problemas tenham sido encontrados, se faz necessária uma análise em cima do projeto de forma a encontrar soluções que evitem a perda de *deadline* por parte de alguma mensagem. Foi proposta neste trabalho a utilização de algorítmos genéticos para executar a tarefa de encontrar uma solução que elimine problemas encontrados durante a análise temporal. O algorítmo genético foi concebido para atuar sobre a prioridade das mensagens. A partir de uma lista de mensagens inicial, com sua sequência de prioridades, o algorítmo cria soluções modificando as prioridades das mensagens, isto é, modificando sua posição no escalonamento da rede. O tempo de resposta no pior caso faz parte da avaliação de cada possível solução, juntamente com penalidades e recompensas de acordo com características da solução. Os parâmetros do algorítmo genético foram implementados de forma a permitir o usuário ajustar o software, fazendo-o convergir para uma solução ótima. A aleatoriedade do algorítmo genético é uma função da taxa de permutação e de cruzamento. Usar valores errados para estes parâmetros pode fazer com que o algorítmo nunca encontre a solução desejada. Um outro fator importante é a concepção da função de avaliação. A função de avaliação deve exprimir exatamente as características que precisam ser consideradas na definição de uma solução boa ou ruim.

Neste trabalho foi implementado, primeiramente, o algorítmo genético em VBA. As simulações mostram a evolução das soluções. Em seguida, por questões de velocidade de processamento, foi desenvolvido um outro software em C# onde também pôde-se notar a evolução das notas de avaliação. A implementação de algorítmos genéticos na otimização do escalonamento de mensagens é uma forma eficaz de se obter a solução para problemas temporais em redes CAN, protocolo

SAE J1939. Apesar das 31 mensagens do protocolo SAE J1939 já terem seus identificadores definidos, as simulações deste trabalho foram feitas considerando todas as mensagens por motivo de melhor visualização dos resultados.

6.2. Próximos Passos.

Neste trabalho um algorítmo genético foi utilizado para otimizar a sequência de transmissão de mensagens SAE J1939 em uma rede CAN. A atuação do algorítmo, como foi dito antes, se deu sobre a prioridade das mensagens, ou seja, a ordem pela qual as mensagens ganharão acesso ao barramento.

Um próximo passo em direção a um resultado mais eficiente é a atuação do algorítmo genético não só na prioridade das mensagens, mas também dentro de uma faixa de valores permissíveis para taxa de transmissão, e também sobre outras características das mensagens.

É preciso considerar também em trabalhos futuros as mensagens multi-pacote que são transmitidas em situações como por exemplo ao executar a parametrização de um módulo eletrônico no final da linha de produção de um veículo. Neste trabalho estas mensagens multi-pacote não foram consideradas pois não são enviadas durante a operação normal do veículo, portanto não correm o risco de atrasarem a transmissão de outra mensagem.

Além disto, as mensagens de definição de endereço de rede também não foram consideradas pelo mesmo motivo das mensagens multi-pacote.

7. LISTA DE REFERÊNCIAS

[1] CANBus Specification Version 2.0 (A and B), Robert Bosch Gmbh, 1991.

[2] ISO 11898 – Road Vehicles – Interchange of Digital Information – Controller Area Network for High-Speed Communication, 1993.

[3] SAE International, SAE J1939/11,*Phisical Layer 250Kbps, Twisted Shielded Pair* , 1994.

[4] SAE International, SAE J1939/15, *Phisical Layer 250Kbps, Twisted Pair* , 1994.

[5] SAE International, SAE J1939/21, *Data Link Layer,* 1994.

[6] SAE International, SAE J1939/31, *Network Layer,* 1994.

[7] SAE International, SAE J1939/71, *Vehicle Application Layer,* 1994.

[8] SAE International, SAE J1939/73, Application *Layer - Diagnostics,* 1994.

[9] SAE International, SAE J1939/81, *Network Management,* 1994.

[10] LIU, C.; LAYLAND J., "Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment", *Journal of the ACM*, 20(1):46--61, 1973.

[11] TINDELL, K.; BURNS, A.; WELLINGS, A. , "An extendible approach for analysing fixed priority hard real-time tasks", *Real-time Systems* 6(2) pp. 133-151, 1994.

[12] TINDELL, K.; BURNS, A, "Guaranteed Messages Latencies for Distributed Safety-Critical Hard-Real-Time Systems Control Networks", YCS 229, Dep. Computer Science, University of York, Junho 1994.

[13] TINDELL, K.; BURNS, A.; HANSSON, H.; WELLINGS, A., "Analysing real-time Communications: Controller Area Network", *15th IEEE Real Time Systems Symposium*, pages 259-265, 1994.

[14] TINDELL, K.; BURNS, A.; WELLINGS, A., "Calculating Controller Area Network (CAN) Message Response Time", *Control Engineering Practice*, vol. 3, no. 8, pp. 1163-- 1169, 1995.

[15] ZUBERI, K.M.; SHIN, K.G. , "Non-Preemptive Scheduling of Message on Controller Are Network for Real Time Control Applications", *IEEE Transactions On Robotics And Automation,* pp. 240 -249, 1995.

[16] RUFINO, J.; RODRIGUES L.; VERISSIMO, P.; ARROZ, G.; ALMEIDA, C., "Fault Tolerant Broadcasts in CAN", 1998. *28th IEEE international symposium on fault-tolerant computing (FTCS'98)*, pp 150–159, 1998.

[17] PINHO, L.M.; VASQUES, F.; TOVAR E., "Integrating inaccessibility in response time analysis of CAN networks". *3rd IEEE International Workshop on Factory Communication Systems*, pp. 77-84, Porto, Portugal, Setembro 2000.

[18] PUNNEKKAT, S.; HANSSON, H.; NORSTRÖM, C., "Response Time Analysis under Error for CAN", 2000. *6th Real Time Technology and Applications symposium (RTAS)*. pp. 258-265, Washington DC, 2000.

[19] PINHO, L.M.; VASQUES, F., "Timing Analysis of Reliable Real-Time Communication in CAN Networks", *13th Euromicro Conference on Real-Time Systems,* Estocolmo, Suécia, 2000.

[20] PINHO, L.M.; VASQUES, F.; FERREIRA, L., "Programming Atomic Multicast in CAN", *10th International Real-time ADA Workshop*, Avila, Espanha, Setembro, 2000 .

[21] PINHO, L.M.; VASQUES, F., "Improved Fault Tolerant Broadcasts in CAN", *8th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 305-313, 2001.

[22] NOLTE, T.; PUNNEKKAT, S.; HANSSON, H.; NORSTRÖM, C., "Using Bit-Stuffing Distrtibutions in CAN analysis", *IEEE Real-Time Embedded Systems Workshop*, Dec. 3, 2001.

[23] NOLTE, T.; HANSSON, H.; NORSTRÖM, C., "Minimize CAN Response-Time Jitter by Message Manipulation", *Proc. Of the 8th Real-time and Embedded Technology and Applications Symposium*, pp 197-206, 2002.

[24] BROSTER, I.; BURNS, A.; RODRÍGUEZ-NAVAS, G., "Probabilistic Analysis of CAN with faults", *In Proceedings of the 23rd Real-time Systems Symposium*, pp 3-5, 2002.

[25] NOLTE, T., "Reducing Pessimism in CAN Response Time Analysis ", *Mälardalen Real-Time Research Centre, Department of Computer Engineering, Mälardalen University*, Västeras, Suécia, 2002.

[26] SANTOS, M.M.D.; STEMMER, M.R.; VASQUES, F., "Schedulability Analysis of Messages in a CAN Network Applied to a Unmanned Airship", *IEEE 2002 28th Annual Conference of the Industrial Electronics Society IECON 02*, pp: 1909 – 1914, Nov. 2002.

[27] NOLTE, T.; SJÖDIN, M.; HANSSON, H., "Server-based scheduling of the CAN Bus", *Mälardalen Real-Time Research Centre, Department of Computer Engineering, Mälardalen University*, Västeras, Suécia, 2003.

[28] FERREIRA, J.; OLIVEIRA, A.; FONSECA, P.; FONSECA, J., "An Experiment to Access Bit Error-Rate in CAN*", 3rd Intl Workshop on Real-Time Networks* , Catania, Italia, Junho 2004.

[29] GAUJAL, B.; NAVET, N., "Fault Confinement mechanisms on CAN: analysis and improvements", IEEE Transactions on vehicular Technologies, ,vol 54, pp 1103-1113. Maio 2005.

[30] BROSTER, I., "Flexibility in Dependable Real Time Communication", *D.Phil Thesis*, Department of Computer Science, University of York, Ago 2003.

[31] DAVIS, R. et all, "Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised", *Real-time Systems Journal,* vol.35, no. 3, pp. 239-272, Abr. 2007.

[32] GUIMARÃES, A.; SARAIVA, A.M., "Um Roteiro de Implementação de Uma Rede CAN", *In proceeding of SIMEA*, São Paulo, 2003.

[33] GUIMARÃES, A.; SARAIVA, A.M, "O Protocolo Internacional CAN Bus na Comunicação de Dados de Sistemas Agrículas: Presente e Futuro", *In proceedings of 13th SAE Brasil International Congress*, São Paulo, 2004.

[34] CAMPOS, M.F.; FRANCO, L.R., "Analyzing SAE J1939 Messages Worst Case Response Time", *In proceedings of 18th SAE Brasil International Congress*, São Paulo, 2009.

[35] CAMPOS, M.F.; FRANCO, L.R, "Optimizing J1939 Messages Response Time by Using Evolutionary Algorithms", *In proceedings of 18th SAE Brasil International Congress*, São Paulo, 2009.

[36] WEISE, T., "Global Optimization Algorithms – Theory and Practice", *e-book*, 2ª edição, 2009. Disponível em: www.it-weise.de [acesso em 10.06.09].

[37] WHITLEY, D., “Genetic Algorithms and Evolutionary Computing”, *Colorado State University*, 2002, Disponível em: http://www.cs.colostate.edu/~genitor/2002/encyclo.pdf [acesso em 05.19.09]

[38] ENGELMANN, A.S., “Uma proposta de otimização de comunicação no protocolo foundation fieldbus utilizando Algorítmos Genéticos”, *Dissertação de Mestrado*, IESTI, Universidade Federal de Itajubá, 2003.

[39] WHITLEY, D., “A genetic Algorithm tutorial”, Computer Science Department, Colorado State University, 1996, disponível em: http://www.cs.colostate.edu/~genitor/MiscPubs/tutorial.pdf [acesso em 01.07.09]

[40] WALL, M., “GAlib: A C++ Library for genetic Algorithms Components”, *Version 2.4*, Mechanical Engineering Department, Massachusetts Institute of Technology, 1996. Disponível em: http://lancet.mit.edu/ga/ [acesso em 01.07.09]

[41] WALL, M., “A Genetic Algorithm for Resource-Constrained Scheduling”, Mechanical Engineering Department, Massachusetts Institute of Technology, DPhil Thesis, 1996. Disponível em: http://lancet.mit.edu/~mbwall/phd/ [acesso em 01.07.09].

[42] FISSGUS, U., “Scheduling using Genetic Algorithms”, *Technical Report*, Computer Science Department, University of Halle-Wittenberg, 2000. Available on: http://www.mathematik.uni-halle.de/reports/sources/2000/00-01report.ps [acesso em 22.06.09].

[43] SANTOS, M.M.Santos, “Tecnologia de Redes Automotivas”, *In proceeding of AEA embedded electronics course*, São Paulo, 2004.

[44] CUGNASCA, C.E., SARAIVA, A.M., “Redes embarcadas em máquinas agrícolas para agricultura de precisão: novos desafios e tendências”, *in proceedings of IV Simpósio de Agricultura de Precisão,* Viçosa, 2007.

[45] CAMPOS, M.F.; FRANCO, L.R.H., "Evolução da Proposta de Otimização do Tempo de Resposta de Mensagens CAN SAE J1939 Usando Algorítmos Genéticos", *VI Simpósio de Excelência em Gestão e Tecnologia – SEGET 2009*, Resende, 2009.

[46] HOLLAND J.H., "Adaptation in Natural and Artificial Systems", *MIT Press*, 1992. (1ª edição, 1975).

8. ANEXO I

8.1. Códigos em Visual Basic.

Cálculo do tempo de transmissão de cada mensagem no barramento : Cm

```
For m = 8 To 58 Step 1

   i = Application.WorksheetFunction.Floor(((Nbit + (8 * Sm) - 1) / 4), 1)
   Cm = (Nbit + 13 + (8 * Sm) + i) * Tbit

Next m
```

Cálculo do tempo máximo de bloqueio Bm:

```
For m = 8 To 58 Step 1
   For j = m To 58 Step 1
      If Cells(j, 11) > Cells(j + 1, 11) Then
      Bm = Cells(j, 11)
      Else
      Bm = Cells(j + 1, 11)
      End If
   Next j
   Cells(m, 12) = Bm
Next m
```

Cálculo da contribuição do erro por (Tindell, 1995), com Wm e Rm:

```
For m = 8 To 58 Step 1
   i = 0
   Sum = 0
      For i = 1 To 10
      Wm = Wmm
      Sum = 0
      For j = m To 8 Step -1
            Sum = Sum + (Application.WorksheetFunction.Ceiling(((Wm +
Jm + Tbit) / Tj), 1)) * Cj
   Next j
      Ner = nburst + (Application.WorksheetFunction.Ceiling(((t + Jm) /
Ter), 1)) - 1
      E = Ner * (0.64 + 31 * Tbit)
      Wmm = Bm + Sum + E
      Next i
```

```
    Rm = Cm + Wmm
  Next m
```

Calculo da contribuição do erro por (Pinho, 2000) , com Wm e Rm:

```
  For m = 8 To 58 Step 1
    Sum = 0
    SumT = 0
   For i = 1 To 10
      Wm = Wmm
      Sum = 0
      For j = m To 8 Step -1
            Sum = Sum + (Application.WorksheetFunction.Ceiling(((Wm +
Jm + Tbit) / Tj), 1)) * Cj
      Next j
      tINA = 0.64 + 31 * Tbit + 3 * Tbit
      INAbus = nbus * ((Wm + Cm) / t) * tINA
      INAtransc = 16 * tINA
      Wmm = Bm + Sum + INAbus + INAtransc
    Next i
   Rm = Cm + Wmm
  Next m
```

Cálculo da contribuição do erro por (Punnekkat, 2000), com Wm e Rm:

```
  O = 31 * Tbit + 0.64
  For m = 8 To 58 Step 1
    Sum = 0
    Rm = Cm + Wmm
      For i = 1 To 10 Step 1
      Wm = Wmm
      Sum = 0
      For j = m To 8 Step -1
        Bua          =          Application.WorksheetFunction.Min(n,
Application.WorksheetFunction.Ceiling(((t Mod Tgf) / tuf), 1))
        Bub = Application.WorksheetFunction.Floor(t / Tgf, 1)
        Bu = Application.WorksheetFunction.Min(n * b, Bub + Bua)
        Re          =          Application.WorksheetFunction.Max(0,
Application.WorksheetFunction.Ceiling((t - Tgf * b) / rf, 1))
        En = Bu * (O + Application.WorksheetFunction.Max(0, Ii - Tbit))
+ Re * (O + Application.WorksheetFunction.Max(0, Ii - Tbit))
        Sum = Sum + (((Wm + Tbit) / Tj) * Cj)
      Next j
      Wmm = Bm + Sum + En
      Next i
    Rm = Cm + Wmm
  Next m
```

Cálculo do tempo de fila, Wm , e do tempo de resposta no pior caso, Rm , de acordo com (Davis, 2007):

```
For m = 8 To 58 Step 1
   i = 0
   Sum = 0
```

Cálculo do período de barramento ocupado:tmm

```
   For i = 1 To 10
      TM = tmm
      Sum = 0
      For j = m To 8 Step -1
         Sum = Sum + (Application.WorksheetFunction.Ceiling(((TM + Jm) /
Tj), 1)) * Cj
      Next j
      tmm = Bm + Sum + E
   Next i
```

Cálculo de quantas instancias estarão disponíveis durante período ocupado

```
   Qm = Application.WorksheetFunction.Ceiling(((tmm + Jm) / Tj), 1)
```

Cálculo do W e R para cada instância

```
   Wmmi = 0
   Rmi = 0
   For k = 0 To Qm - 1
      For i = 1 To 10
         Wm = Wmm
         Sum = 0
         For j = m To 8 Step -1
            Sum = Sum + (Application.WorksheetFunction.Ceiling(((Wm +
Jm + Tbit) / Tj), 1)) * Cj
         Next j
      Wmm = Bm + k * Cm + Sum + E
      Next i
      Rm = Cm + Wmm + Jm - k * Tj
```

Cálculo do pior W e R dentre as instâncias de uma mensagem

```
      If Rm > Rmi Then
            Rmi = Rm
      End If
      If Wmm > Wmmi Then
            Wmmi = Wmm
      End If
   Next k
Next m
```

Rotina para Otimização do escalonamento baseado em GA

```
PopIni
Ngen = Cells(5, 19)
For u = 1 To Ngen
   avaliação
   fitness
   reprodução
   substituição
Next u
```

Criando a população inicial

```
Private Sub PopIni()
NpopIni = Cells(4, 19)
For i = 1 To NpopIni
   '----Preparando indivíduo para permutação----
   For j = 8 To 58
      Cells(j, 15) = Cells(j, 1)
   Next j
   '----Permutação inicial------
   Randomize
   pt1 = Int((51) * Rnd + 1)
   Randomize
   pt2 = Int((51) * Rnd + 1)
   Cells(7, 15) = Cells(pt2, 15)
   Cells(pt2, 15) = Cells(pt1, 15)
   Cells(pt1, 15) = Cells(7, 15)
   '-----Segunda permutação------
   Randomize
   pt1 = Int((51) * Rnd + 1)
   Randomize
   pt2 = Int((51) * Rnd + 1)
   Cells(7, 15) = Cells(pt2, 15)
   Cells(pt2, 15) = Cells(pt1, 15)
   Cells(pt1, 15) = Cells(7, 15)
   '-----Posicionando na tabela------
   For j = 1 To 51
      Cells(j, 15 + i) = Cells(j, 15)
   Next j
Next i
End Sub
```

Função avaliação - Avaliar o grupo com relação ao tempo de resposta

```
Private sub avaliação ()

For x = 16 To 15 + NpopIni
   For z = 8 To 58
      For k = 8 To 58
         If Cells(z, x) = Cells(k, 1) Then
            Cells(z + 54, 6) = Cells(k, 1)
            Cells(z + 54, 7) = Cells(k, 7)
            Cells(z + 54, 8) = Cells(k, 8)
            Cells(z + 54, 9) = Cells(k, 9)
            Cells(z + 54, 10) = Cells(k, 10)
         End If
      Next k
   Next z
'------------ Implementação da função avaliação----------------------------------
   Wm = 0
   Cj = 0
   Tj = 0

   'Calculo do tempo de transmissão de cada mensagem no barramento : Cm
   For m = 62 To 112 Step 1
      i = Application.WorksheetFunction.Floor(((Nbit + (8 * Sm) - 1) / 4), 1)
      Cm = (Nbit + 13 + (8 * Sm) + i) * Tbit
      Cells(m, 11) = Cm
   Next m

   'Calculo do tempo máximo de bloqueio: Bm
   For m = 62 To 112 Step 1
      For j = m To 112 Step 1
         If Cells(j, 11) > Cells(j + 1, 11) Then
            Bm = Cells(j, 11)
         Else
            Bm = Cells(j + 1, 11)
         End If
      Next j
      Cells(m, 12) = Bm
   Next m

   'Calculo do atraso de fila, Wm , e do tempo de resposta no pior caso, Rm.
(Davis,2007)
   Wmm = Cells(62, 12)
   tmm = Cells(62, 11)
   av1 = 0
   For m = 62 To 112 Step 1
      i = 0
      Sum = 0
```

```
        SumT = 0

    'Cálculo do período de barramento ocupado
        For i = 1 To 10
            TM = tmm
            Sum = 0
            For j = m - 1 To 62 Step -1
                Sum = Sum + (Application.WorksheetFunction.Ceiling(((TM + Jm) /
Tj), 1)) * Cj
            Next j
            tmm = Bm + Sum
        Next i

    'Calculando instancias estarão durante período ocupado
        Tj = Cells(m, 7)
        Qm = Application.WorksheetFunction.Ceiling(((tmm + Jm) / Tj), 1)

    'Calculando W e R para cada instância
        Cm = Cells(m, 11)
        Wmmi = 0
        Rmi = 0
        For k = 0 To Qm - 1
            For i = 1 To 10
                Wm = Wmm
                Sum = 0
                For j = m - 1 To 62 Step -1
                    Sum = Sum + (Application.WorksheetFunction.Ceiling(((Wm +
Jm + Tbit) / Tj), 1)) * Cj
                    'Sum = Sum + (((Wm + Jm + Tbit) / Tj) * Cj)
                Next j
            Wmm = Bm + k * Cm + Sum
            Next i
            Rm = Cm + Wmm + Jm - k * Tj

        'Calculanto o pior W e R dentre as instâncias de uma mensagem
            If Rm > Rmi Then
                Rmi = Rm
            End If
            If Wmm > Wmmi Then
                Wmmi = Wmm
            End If
        Next k

    'Repassando o pior caso para tabela
        Cells(m, 13) = Wmmi
        Cells(m, 14) = Rm
```

```
'Calculando o delta fitness e somatória

    Cells(m, 15) = Cells(m, 9) - Cells(m, 14)
    av1 = av1 + Cells(m, 15)
Next m

'Calculando penalidades
'Penalidade 1 - gene duplicado
j = 0
For m = 62 To 112
    For i = 62 To 112
        If Cells(m, 6) = Cells(i, 6) Then
            j = j + 1
        End If
    Next i
Next m
If j > 51 Then
    av1 = av1 - 50000
End If

'Penalidade 2 - predecedencia temporal
j = 0
For m = 62 To 112
    If Cells(m, 7) > Cells(m + 1, 7) Then
        av1 = av1 - 250
    End If
    If Cells(m, 7) < Cells(m + 1, 7) Then
        av1 = av1 + 2500
    End If
    If Cells(m, 7) > Cells(m - 1, 7) Then
        av1 = av1 + 2500
    End If
    If Cells(m, 7) < Cells(m - 1, 7) Then
        av1 = av1 - 250
    End If
Next m

'Seleciona o Melhor de cada geração
If av1 > Cells(113, 1) Then
    For z = 62 To 112
        Cells(z, 1) = Cells(z, 6)
        For m = 8 To 58
            If Cells(z, 1) = Cells(m, 1) Then
                Cells(z, 2) = Cells(m, 3)
            End If
        Next m
        Cells(z, 3) = Cells(z, 7)
        Cells(z, 4) = Cells(z, 9)
```

```
            Cells(z, 5) = Cells(z, 14)
        Next z
        Cells(113, 1) = av1
        Cells(113, 2) = Cells(113, 2) + 1
    End If
    Cells(62, x) = av1
Next x
Ngen = Cells(5, 19)
k = 1
z = 1 + (Ngen - Cells(4, 25))
For x = 16 To 15 + NpopIni
    Worksheets("Resultados").Cells(z, k) = Cells(62, x)
    k = k + 1
Next x
End Sub

'--------------------------------  cálculo da função fitness ----------------------------

Private Sub fitness()
'Calcula somatória das avaliações
SumAv1 = 0
For x = 16 To 15 + NpopIni
    SumAv1 = SumAv1 + Cells(62, x)
Next x

'Calcula média das avaliações - fitness
fit = 0
media = SumAv1 / NpopIni
For x = 16 To 15 + NpopIni
    fit = Cells(62, x) / media
    Cells(63, x) = fit
Next x

'Ordena os indivíduos
For x = 16 To 15 + NpopIni
    Cells(64, x) = 0
    Cells(65, x) = 0
Next x
z = 0
For i = 1 To NpopIni
    For x = 16 To 15 + NpopIni
        If Cells(65, x) = 0 Then
            If Cells(63, x) >= Cells(59, 16) Then
                Cells(59, 16) = Cells(63, x)
                z = x
            End If
        End If
    Next x
```

```
    For x = 16 To 15 + NpopIni
        If z = x Then
            Cells(64, x) = i
            Cells(65, x) = 1
        End If
    Next x
    Cells(59, 16) = -50000
Next i
End Sub

'---------------------------------------  Reprodução -------------------------------

Private Sub reprodução()
'Limpa área para reprodução
For x = 16 To 15 + NpopIni
    For z = 70 To 120
        Cells(z, x) = 0
        Cells(z + 54, x) = 0
    Next z
Next x

'Permutação Seletiva
TP = Cells(3, 19)
For x = 16 To 15 + NpopIni
    If Cells(64, x) < ((2 * NpopIni) / 3) Then
        For z = 8 To 58
            Cells(z + 62, x) = Cells(z, x)
        Next z
        Randomize
        i = Rnd
        If TP > i Then
            Randomize
            pt1 = Int((50) * Rnd + 70)
            Randomize
            pt2 = Int((50) * Rnd + 70)
            If Cells(pt1, x) >= Cells(pt1 + 1, x) Then
                Cells(7, 15) = Cells(pt1 + 1, x)
                Cells(pt1 + 1, x) = Cells(pt1, x)
                Cells(pt1, x) = Cells(7, 15)
            End If
            If Cells(pt1, x) <= Cells(pt1 - 1, x) Then
                Cells(7, 15) = Cells(pt1 - 1, x)
                Cells(pt1 - 1, x) = Cells(pt1, x)
                Cells(pt1, x) = Cells(7, 15)
            End If
            If Cells(pt2, x) >= Cells(pt2 + 1, x) Then
                Cells(7, 15) = Cells(pt2 + 1, x)
                Cells(pt2 + 1, x) = Cells(pt2, x)
```

```
            Cells(pt2, x) = Cells(7, 15)
        End If
        If Cells(pt2, x) <= Cells(pt2 - 1, x) Then
            Cells(7, 15) = Cells(pt2 - 1, x)
            Cells(pt2 - 1, x) = Cells(pt2, x)
            Cells(pt2, x) = Cells(7, 15)
        End If
        Randomize
        pt1 = Int((50) * Rnd + 70)
        Randomize
        pt2 = Int((50) * Rnd + 70)
        Cells(7, 15) = Cells(pt2, x)
        Cells(pt2, x) = Cells(pt1, x)
        Cells(pt1, x) = Cells(7, 15)
      End If
   End If
Next x

'--------------------------------Cruzamento-----------------------------------------------

TC = Cells(2, 19)
For x = 16 To 15 + NpopIni
   If Cells(64, x) < 2 * NpopIni / 3 Then
      Randomize
      i = Rnd
      If i < TC Then
         Randomize
         pt2 = Int((50) * Rnd + 70)
         Do
         Randomize
         pt1 = Int((NpopIni - 1) * Rnd + 16)
         Loop Until Cells(64, pt1) < (2 * NpopIni / 3)
         For z = 70 To pt2
            Cells(z + 54, 14) = Cells(z, x)
            Cells(z + 54, 15) = Cells(z, pt1)
         Next z
         For z = pt2 To 120
            Cells(z + 54, 14) = Cells(z, pt1)
            Cells(z + 54, 15) = Cells(z, x)
         Next z
         For z = 70 To 120
            Cells(z, x) = Cells(z + 54, 14)
            Cells(z, pt1) = Cells(z + 54, 15)
         Next z
      End If
   End If
Next x
End Sub
```

```
'-------------------------------- Substituição da população --------------------------
Private Sub substituição()
'Conta os filhos da geração anterior
i = 0
For x = 16 To 15 + NpopIni
   If Cells(70, x) = 0 Then
      i = i + 1
   End If
Next x

'Seleciona os melhores individuos da geração anterior
k = i
For m = 16 To 15 + i
   For x = 16 To 15 + NpopIni
      If Cells(124, m) = 0 Then
         If Cells(64, x) = k Then
            For z = 8 To 58
               Cells(z + 116, m) = Cells(z, x)
            Next z
         End If
      End If
   Next x
   k = k - 1
Next m

'Posiciona os filhos da geração anterior nas vagas deixadas pelos descartados
m = 16 + i
For x = 16 To 15 + NpopIni
   If Cells(70, x) <> 0 Then
      If Cells(70, x) <> 2 Then
         For z = 70 To 120
            Cells(z + 54, m) = Cells(z, x)
         Next z
         m = m + 1
         Cells(65, x) = 2
      End If
   End If
Next x

'Substitui a população pela nova geração ( novos+ melhores da geração
anterior)
For m = 16 To 15 + NpopIni
   For z = 124 To 174
      Cells(z - 116, m) = Cells(z, m)
   Next z
Next m
End Sub
```

8.2. Códigos em C#.

Código principal:

```
namespace Darwin3
{
   public partial class Form1 : Form
   {
      String line = null;
      public Databaseinicial DBI;
      double[,] dbini;

      public Form1()
      {
         InitializeComponent();
         DBI = new Databaseinicial();
      }

      private void databaseToolStripMenuItem_Click(object sender,
EventArgs e)
      {
         opf_db.Filter = "Text files (*.txt)|*.txt";
         if (opf_db.ShowDialog() == DialogResult.OK)
         {
            tb_tabela.Text = opf_db.FileName;
         }
      }

      private void bt_inserir_Click(object sender, EventArgs e)
      {
         StreamReader sr = new StreamReader(opf_db.FileName);
         line = sr.ReadLine();

         while (line != null)
         {
            String[] linesplit = line.Split(new char[] { ';' });
            Gene novogene = new Gene(Convert.ToInt32(linesplit[0]),
Convert.ToString(linesplit[1]), Convert.ToInt32(linesplit[2]),
Convert.ToInt32(linesplit[3]), Convert.ToInt32(linesplit[4]), 0);
            DBI.Add(novogene);

            Atualizar();

            line = sr.ReadLine();
         }
         sr.Close();
      }
```

```
        private void Atualizar()
        {
            lb_inicio.Items.Clear();
            for (int i = 0; i < DBI.Count; i++)
            {
                lb_inicio.Items.Add(DBI[i].msgprio + " " + DBI[i].msgname +
" " + DBI[i].msgdead);
            }
        }

        private void bt_avaliar_Click(object sender, EventArgs e)
        {
            DBI.stf = Convert.ToInt32(mk_stuff.Text);
            DBI.bt = (Convert.ToDouble(mk_bittime.Text)) / 1000;
            DBI.CalculoWCRT(DBI);

            lb_inicio.Items.Clear();
            for (int i = 0; i < DBI.Count; i++)
            {
                if (DBI[i].msgwcrt >= DBI[i].msgdead)
                {
                    lb_inicio.Items.Add(DBI[i].msgprio + " " + DBI[i].msgname
+ " " + DBI[i].msgdead + " " + DBI[i].msgwcrt + " " + "FALHA!!");
                }
                else
                {
                    lb_inicio.Items.Add(DBI[i].msgprio + " " + DBI[i].msgname
+ " " + DBI[i].msgdead + " " + DBI[i].msgwcrt + " " + "OK");
                }
            }
        }

        private void bt_otimizar_Click(object sender, EventArgs e)
        {
            Calculos Calc = new Calculos();
            dbini = new double[DBI.Count, 5];
            int npop = Convert.ToInt32(mk_pop.Text);
            int nger = Convert.ToInt32(mk_ger.Text);
            int txper = Convert.ToInt32(mk_perm.Text);
            int txcr = Convert.ToInt32(mk_cross.Text);
            double[,,] Melhoresind = new double[nger, DBI.Count, 5];
            double[,] Omelhor = new double[DBI.Count, 5];
            double[,] Pxy = new double[2, nger];
            double[] melhoresnotas = new double[nger];
            double[] medias = new double[nger];
            int dblength = DBI.Count;
            double stuffb = Convert.ToDouble(mk_stuff.Text);
```

```
double bttime = (Convert.ToDouble(mk_bittime.Text))/1000;
double mnota = 0;

for (int i = 0; i < DBI.Count; i++)
{
   dbini[i, 0] = DBI[i].msgprio;
   dbini[i, 1] = DBI[i].msgtx;
   dbini[i, 2] = DBI[i].msgdatabyte;
   dbini[i, 3] = DBI[i].msgdead;
   dbini[i, 4] = DBI[i].msgwcrt;
}

Calc.Criapop(dbini, dblength, npop);

for (int i = 0; i < nger; i++)
{
   Calc.avaliar(dblength, npop, stuffb, bttime);

   medias[i] = Calc.media;
   melhoresnotas[i] = Calc.melhornota;

   for (int j = 0; j < DBI.Count; j++)
   {
      for (int k = 0; k < 5; k++)
      {
         Melhoresind[i, j, k] = Calc.melhorind[j, k];
      }
   }
   Calc.selecionar(dblength, npop);
   Calc.reproduzir(dblength, npop, txper, txcr);
   Calc.substituir(dblength, npop);
}

for (int i = 0; i < nger; i++)
{
   if (melhoresnotas[i] > mnota)
   {
      mnota = i;

      for (int j = 0; j < DBI.Count; j++)
      {
         for (int k = 0; k < 5; k++)
         {
            Omelhor[j, k] = Melhoresind[(int)mnota, j, k];
         }
      }
   }
}
```

```
            lb_otimizado.Items.Clear();

            for (int i = 0; i < DBI.Count; i++)
            {
               if (Omelhor[i, 3] > Omelhor[i, 4])
               {
                  lb_otimizado.Items.Add(Omelhor[i, 0] + " " + Omelhor[i, 3]
+ " " + Omelhor[i, 4] + " " + "OK");
               }
               else
               {
                  lb_otimizado.Items.Add(Omelhor[i, 0] + " " + Omelhor[i, 3]
+ " " + Omelhor[i, 4]+ " " + "FALHA!!");
               }
            }

            Graphics gp = pb.CreateGraphics();
            gp.Clear(this.BackColor);
            Pen p1 = new Pen(Color.Black, 2);
            Pen p = new Pen(Color.Blue, 1);

            gp.DrawLine(p1, 0, pb.Height, pb.Width, pb.Height);
            gp.DrawLine(p1, 0, 0, 0, pb.Height);

            for (int i = 0; i < nger; i++)
            {
               Pxy[0, i] = i*(pb.Width/nger);
               Pxy[1, i] = 237 - (pb.Height*medias[i])/100000;

               gp.DrawEllipse(p, (float)(Pxy[0, i] - 1.0), ((float)(Pxy[1, i] -
1.0)), 2, 2);
            }
         }
      }
   }
```

Classe Database:

```
namespace Darwin3
{
   public class Databaseinicial:CollectionBase
   {
      public int stf = 0;
      public double bt = 0;

      public void Add(Gene novogene)
      {
```

```
            InnerList.Add(novogene);
        }

        public Gene this[int index]
        {
            get
            {
                if (index > Count)
                {
                    return null;
                }
                return (Gene)InnerList[index];
            }
        }

        public double CalculoWCRT(Databaseinicial db)
        {
            double[] Cm = new double[db.Count];
            double[] Bm = new double[db.Count];
            double[] Wm = new double[db.Count];
            double[] Rm = new double[db.Count];
            double Bmi;
            double Tm;
            double Tmi;
            double sum;
            double Wmi;
            double Wmx;
            double Wmm;
            double Rmi;
            double av = 0;
            int count;

            for (int i = 0; i < db.Count; i++)
            {
                Cm[i] = (Convert.ToDouble(stf) + 13 + (8 * db[i].msgdatabyte)
+ (Math.Floor((Convert.ToDouble(stf) + (8 * db[i].msgdatabyte) - 1) / 4)))
* bt;
            }

            for (int i = 0; i < db.Count; i++)
            {
                Bmi = 0;

                for (int j = i; j < db.Count - 1; j++)
                {
                    if (Cm[i] > Cm[j])
                    {
                        Bmi = Cm[i];
```

```
                }
                else
                {
                    Bmi = Cm[j];
                }
            }
            Bm[i] = Bmi;
        }

        for (int i = 0; i < db.Count; i++)
        {
            Rmi = 0;
            Tmi = 0;
            Wmi = 0;
            Wmx = 0;
            sum = 0;
            Tm = Cm[i];
            Wmm = Bm[i];

            for (int j = 0; j < 11; j++)
            {
                Tmi = Tm;
                sum = 0;
                for (int k = 0; k < db.Count; k++)
                {
                    sum = sum + ((Math.Ceiling(((Tmi + 0.2) / (db[k].msgtx)))
* Cm[k]));
                }

                Tm = Bm[i] + sum;
            }

            int Tj = db[i].msgtx;
            double Qm = Math.Ceiling((Tm + 0.2) / Tj);

            for (int j = 0; j < Qm; j++)
            {
                for (int k = 0; k < 11; k++)
                {
                    Wmx = Wmm;
                    sum = 0;
                    for (int l = i; l >=0; l--)
                    {
                        sum = sum + (Math.Ceiling((Wmx + 0.2 + bt) /
db[l].msgtx)) * Cm[l];
                    }
                    Wmm = Bm[i] + j*Cm[i] + sum;
                }
```

```
            db[i].msgwcrt = Cm[i] + Wmm + 0.2 - j * db[i].msgtx;

            if (db[i].msgwcrt > Rmi)
            {
               Rmi = db[i].msgwcrt;
            }

            if (Wmm > Wmi)
            {
               Wmi = Wmm;
            }
         }

         Wm[i] = Wmi;
         db[i].msgwcrt = Rmi;

         av = av + db[i].msgdead - db[i].msgwcrt;
      }

// PENALIDADE POR PERDER DEADLINE

      for (int i = 0; i < db.Count; i++)
      {
         if (db[i].msgwcrt >= db[i].msgdead)
         {
            av = av - 5000;
         }
      }

// PENALIDADE POR DUPLICIDADE
      count = 0;
      for (int i = 0; i < db.Count; i++)
      {
         for (int j = 0; j < db.Count; j++)
         {
            if (db[i].msgname == db[j].msgname)
            {
               count += 1;
            }
         }
      }

      if (count > db.Count)
      {
         for (int i = 0; i < (count - db.Count); i++)
         {
            av = av - 50000;
```

```
            }
        }

    //PENALIDADE POR PRECEDÊNCIA

        for (int i = 1; i < db.Count - 1; i++)
        {
            if (db[i].msgdead > db[i + 1].msgdead)
            {
                av = av - 250;
            }
            if (db[i].msgdead < db[i + 1].msgdead)
            {
                av = av + 2500;
            }
            if (db[i].msgdead > db[i - 1].msgdead)
            {
                av = av + 2500;
            }
            if (db[i].msgdead < db[i - 1].msgdead)
            {
                av = av - 250;
            }
        }
        return av;
    }
}

public class Gene
{
    public int msgprio;
    public string msgname;
    public int msgtx;
    public int msgdead;
    public int msgdatabyte;
    public double msgwcrt;

    public Gene()
    {

    }

    public Gene(int nprio, string nname, int ntx, int ndatabyte, int ndead,
double nwcrt)
    {
        msgprio = nprio;
        msgname = nname;
        msgtx = ntx;
```

```
            msgdead = ndead;
            msgdatabyte = ndatabyte;
            msgwcrt = nwcrt;
        }
    }
}
```

Classe Calculos:

```
namespace Darwin3
{
    class Calculos
    {
        public double[, ,] popinicial;
        public double[, ,] popatual;
        public double[,] melhorind;
        public double media;
        double[, ,] selecao;
        double[] fit;
        double stf = 0;
        double bt = 0;
        public double melhornota = 0;

        public double WCRT(double[,] db, int dblength, double nstf, double
btm)
        {
            double[] Cm = new double[dblength];
            double[] Bm = new double[dblength];
            double[] Wm = new double[dblength];
            double[] Rm = new double[dblength];
            double Bmi;
            double Tm;
            double Tmi;
            double sum;
            double Wmi;
            double Wmx;
            double Wmm;
            double Rmi;
            double av = 0;
            int count;

            for (int i = 0; i < dblength; i++)
            {
                Cm[i] = (nstf + 13 + (8 * db[i, 2]) + (Math.Floor(nstf + (8 *
db[i, 2]) - 1) / 4)) * btm;
            }
```

```
            for (int i = 0; i < dblength; i++)
            {
                Bmi = 0;

                for (int j = i; j < dblength - 1; j++)
                {
                    if (Cm[i] > Cm[j])
                    {
                        Bmi = Cm[i];
                    }
                    else
                    {
                        Bmi = Cm[j];
                    }
                }
                Bm[i] = Bmi;
            }

            for (int i = 0; i < dblength; i++)
            {
                Rmi = 0;
                Tmi = 0;
                Wmi = 0;
                Wmx = 0;
                sum = 0;
                Tm = 0;
                Wmm = 0;
                Tmi = 0;
                Tm = Cm[i];
                Wmm = Bm[i];

                for (int j = 0; j < 11; j++)
                {
                    Tmi = Tm;
                    sum = 0;
                    for (int k = i; k >= 0; k--)
                    {
                        sum = sum + ((Math.Ceiling(((Tmi + 0.2) / db[k, 1])) *
Cm[k]));
                    }

                    Tm = Bm[i] + sum;
                }

                double Tj = 0;
                Tj = db[i, 1];
                double Qm = 0;
                Qm = Math.Ceiling((Tm + 0.2) / Tj);
```

```
                for (int j = 0; j < Qm; j++)
                {
                    for (int k = 0; k < 11; k++)
                    {
                        Wmx = Wmm;
                        sum = 0;
                        for (int l = i; l >= 0; l--)
                        {
                            sum = sum + (Math.Ceiling((Wmx + 0.2 + bt) / db[l,
1])) * Cm[l];
                        }
                        Wmm = Bm[i] + j * Cm[i] + sum;
                    }

                    db[i, 4] = Cm[i] + Wmm + 0.2 - j * db[i, 1];

                    if (db[i, 4] > Rmi)
                    {
                        Rmi = db[i, 4];
                    }

                    if (Wmm > Wmi)
                    {
                        Wmi = Wmm;
                    }
                }

                Wm[i] = Wmi;
                db[i, 4] = Rmi;

                av = av + db[i, 3] - db[i, 4];
            }
            // PENALIDADE POR PERDER DEADLINE

            for (int i = 0; i < dblength; i++)
            {
                if (db[i, 4] >= db[i, 3])
                {
                    av = av - 10000;
                }
            }
            // PENALIDADE POR DUPLICIDADE
            count = 0;
            for (int i = 0; i <dblength; i++)
            {
                for (int j = 0; j < dblength; j++)
                {
```

```
            if (db[i, 0] == db[j, 0])
            {
               count += 1;
            }

         }
      }

      if (count > dblength)
      {
         for (int i = 0; i < (count - dblength); i++)
         {
            av = av - 50000;
         }
      }

      //PENALIDADE POR PRECEDÊNCIA

      for (int i = 1; i < dblength - 1; i++)
      {
         if (db[i, 3] > db[i + 1, 3])
         {
            av = av - 250;
         }
         if (db[i, 3] < db[i + 1, 3])
         {
            av = av + 2500;
         }
         if (db[i, 3] > db[i - 1, 3])
         {
            av = av + 2500;
         }
         if (db[i, 3] < db[i - 1, 3])
         {
            av = av - 250;
         }
      }

      return av;

   }

   public void Criapop(double[,] dbini, int dblength, int npop)
   {
      double[,] dbpivot1 = new double[dblength, 5];
      double[] c = new double[5];
      popinicial = new double[npop, dblength, 5];
```

```
            for (int i = 0; i < dblength; i++)
            {
               dbpivot1[i, 0] = dbini[i, 0];
               dbpivot1[i, 1] = dbini[i, 1];
               dbpivot1[i, 2] = dbini[i, 2];
               dbpivot1[i, 3] = dbini[i, 3];
               dbpivot1[i, 4] = dbini[i, 4];
            }

            for (int i = 0; i < npop; i++)
            {
               Random k = new Random((int)DateTime.Now.Ticks);
               Random l = new Random();

               int a = k.Next(0, dblength);
               int b = l.Next(0, dblength);

               for (int j = 0; j < 5; j++)
               {
                  c[j] = dbpivot1[a, j];
                  dbpivot1[a, j] = dbpivot1[b, j];
                  dbpivot1[b, j] = c[j];
               }

               a = k.Next(0, dblength);
               b = l.Next(0, dblength);

               for (int j = 0; j < 5; j++)
               {
                  c[j] = dbpivot1[a, j];
                  dbpivot1[a, j] = dbpivot1[b, j];
                  dbpivot1[b, j] = c[j];
               }

               for (int j = 0; j < dblength; j++)
               {
                  for (int m = 0; m < 5; m++)
                  {
                     popinicial[i, j, m] = dbpivot1[j, m];
                  }
               }
            }
         }

         public void avaliar(int dblength, int npop, double stuffb, double
bttime)
         {
            double[,] ind = new double[dblength, 5];
```

```
double[,] wcrt = new double[dblength, npop];
double[] av = new double[npop];
double soma = 0;
int indexmelhornota = 0;
fit = new double[npop];
melhorind = new double[dblength, npop];
media = 0;
stf = stuffb;
bt = bttime;

for (int i = 0; i < npop; i++)
{
   for (int j = 0; j < dblength; j++)
   {
      for (int k = 0; k < 5; k++)
      {
         ind[j, k] = popinicial[i, j, k];
      }
   }
   av[i] = WCRT(ind, dblength, stf, bt);

   for (int j = 0; j < dblength; j++)
   {
      popinicial[i, j, 4] = ind[j, 4];
   }
}

for (int i = 0; i < npop; i++)
{
   if (av[i] > melhornota)
   {
      indexmelhornota = i;
   }

   soma = soma + av[i];
}

melhornota = av[indexmelhornota];

for (int i = 0; i < dblength; i++)
{
   for (int j = 0; j < 5; j++)
   {
      melhorind[i, j] = popinicial[indexmelhornota, i, j];
   }
}
media = soma / npop;
```

```
    for (int i = 0; i < npop; i++)
    {
        fit[i] = av[i] / media;
    }
}

public void selecionar(int dblength, int npop)
{
    selecao = new double[npop, dblength, 5];
    bool[] marca = new bool[npop];
    double mfit = 0;
    int index = 0;

    for (int i = 0; i < npop; i++)
    {
        marca[i] = false;
    }

    for (int i = 0; i < npop; i++)
    {
        for (int j = 0; j < npop; j++)
        {
            if ((marca[j] = false) && (fit[j] > mfit))
            {
                mfit = fit[j];
                index = j;
            }
        }
        marca[index] = true;

        for (int j = 0; j < dblength; j++)
        {
            for (int k = 0; k < 5; k++)
            {
                selecao[i, j, k] = popinicial[index, j, k];
            }
        }
    }
    int cb = Convert.ToInt32(0.75 * npop);
    for (int i = npop -1; i > cb; i--)
    {
        for (int j = 0; j < dblength; j++)
        {
            for (int k = 0; k < 5; k++)
            {
                selecao[i, j, k] = 1;
            }
        }
```

```
    }
}

public void reproduzir(int dblength, int npop, int txper, int txcross)

{
    int cb = Convert.ToInt32((2/3) * npop);
    int ca = npop - cb;
    double[,] pivot2 = new double[dblength, 5];
    double[,] filho1 = new double[dblength, 5];
    double[,] filho2 = new double[dblength, 5];
    double[] c = new double[5];

    //PERMUTAÇÃO
    for (int i = 0; i < cb; i++)
    {
        Random TP = new Random();
        int Txp = TP.Next(0, 100);

        if (Txp < txper)
        {
            for (int j = 0; j < dblength; j++)
            {
                pivot2[j, 0] = selecao[i, j, 0];
                pivot2[j, 1] = selecao[i, j, 1];
                pivot2[j, 2] = selecao[i, j, 2];
                pivot2[j, 3] = selecao[i, j, 3];
                pivot2[j, 4] = selecao[i, j, 4];
            }

            Random k = new Random((int)DateTime.Now.Ticks);
            Random l = new Random();
            int a = k.Next(1, dblength - 1);
            int b = l.Next(1, dblength - 1);

            if (pivot2[a, 3] > pivot2[a + 1, 3])
            {
                for (int j = 0; j < 5; j++)
                {
                    c[j] = pivot2[a, j];
                    pivot2[a, j] = pivot2[a + 1, j];
                    pivot2[a + 1, j] = c[j];
                }
            }

            if (pivot2[a, 3] < pivot2[a - 1, 3])
            {
                for (int j = 0; j < 5; j++)
```

```
            {
                c[j] = pivot2[a, j];
                pivot2[a, j] = pivot2[a - 1, j];
                pivot2[a - 1, j] = c[j];
            }
        }

        if (pivot2[b, 3] > pivot2[b + 1, 3])
        {
            for (int j = 0; j < 5; j++)
            {
                c[j] = pivot2[b, j];
                pivot2[b, j] = pivot2[b + 1, j];
                pivot2[b + 1, j] = c[j];
            }
        }

        if (pivot2[b, 3] < pivot2[b - 1, 3])
        {
            for (int j = 0; j < 5; j++)
            {
                c[j] = pivot2[b, j];
                pivot2[b, j] = pivot2[b - 1, j];
                pivot2[b - 1, j] = c[j];
            }
        }

        for (int j = 0; j < dblength; j++)
        {
            for (int m = 0; m < 5; m++)
            {
                selecao[i, j, m] = pivot2[j, m];
            }
        }
    }
}

//CRUZAMENTO
for (int i = 0; i < cb; i++)
{
    Random TC = new Random();
    int TxC = TC.Next(0, 100);

    if (TxC < txcross)
    {
        Random p = new Random((int)DateTime.Now.Ticks);
        Random q = new Random(100);
        int d = p.Next(0, cb);
```

```
            int e = q.Next(0, dblength);

            for (int j = 0; j < e; j++)
            {
               for (int m = 0; m < 5; m++)
               {
                  filho1[j, m] = selecao[i, j, m];
                  filho2[j, m] = selecao[d, j, m];
               }
            }

            for (int j = e; j < dblength; j++)
            {
               for (int m = 0; m < 5; m++)
               {
                  filho1[j, m] = selecao[d, j, m];
                  filho2[j, m] = selecao[i, j, m];
               }
            }

            for (int j = 0; j < dblength; j++)
            {
               for (int m = 0; m < 5; m++)
               {
                  selecao[i, j, m] = filho1[j, m];
                  selecao[d, j, m] = filho2[j, m];
               }
            }
         }
      }
   }

   public void substituir(int dblength, int npop)
   {
      popatual = new double[npop, dblength, 5];

      for (int i = npop -1; i >(Convert.ToInt32((2/3) * npop)); i--)
      {
         for (int j = 0; j < dblength; j++)
         {
            for (int k = 0; k < 5; k++)
            {
               selecao[i, j, k] = popinicial[npop - 1 - i, j, k];
            }
         }
      }
```

```
                for (int i = 0; i < npop; i++)
                {
                   for (int j = 0; j < dblength; j++)
                   {
                      for (int k = 0; k < 5; k++)
                      {
                         popatual[i, j, k] = selecao[i, j, k];
                      }
                   }
                }

                for (int i = 0; i < npop; i++)
                {
                   for (int j = 0; j < dblength; j++)
                   {
                      for (int k = 0; k < 5; k++)
                      {
                         popinicial[i, j, k] = popatual[i, j, k];
                      }
                   }
                }
             }
          }
       }
```
