Algoritmos Genéticos Aplicados para Otimização dos Parâmetros de um Reconhecedor Automático de Fala

Ênio dos Santos Silva e Alberto Luiz Wiggers

***Resumo*— O estado-da-arte em sistemas de reconhecimento automático de fala (*automatic speech recognition* - ASR) envolve decodificadores complexos, constituídos por um grande número de parâmetros configuráveis que influenciam essencialmente na velocidade e no desempenho desses sistemas. Tais parâmetros são geralmente configurados manualmente através do conhecimento e da experiência de um especialista. Especificamente, para a obtenção de um desempenho ótimo é necessário um profundo conhecimento sobre cada parâmetro do decodificador e da relação de compromisso entre a taxa de precisão de acerto das palavras reconhecidas (*accuracy word rate* - AWR) e o fator de tempo empregado para o reconhecimento (*real time factor* - RTF). Nesse contexto, visando uma solução alternativa à configuração manual, este trabalho de pesquisa apresenta uma estratégia para a configuração automática dos parâmetros do decodificador *Julius*, usado como *engine* em um sistema de ASR. Particularmente, a estratégia adotada utiliza algoritmos genéticos para a obtenção de uma configuração ótima através da avaliação de uma função custo que relaciona AWR e RTF. Os resultados dos desempenhos de AWR e RTF são apresentados e comprovam a eficácia da estratégia adotada.**

***Palavras-Chave*— Algoritmos genéticos, decodificador Julius, reconhecimento automático de fala, otimização de parâmetros.**

***Abstract*— State-of-the-art automatic speech recognition (ASR) systems are comprised of complex decoders that have a large number of parameters that allow tuning for performance and speed. These parameters are often optimized manually, based on past experience and specialist knowledge. Specifically, to reach optimal performance, deep understanding of each decoder parameter is necessary, as well as an understanding about the tradeoff between accuracy word rate (AWR) and real time factor (RTF). In order to find the optimal decoding parameters without an manual tuning, this paper presents a strategy that automatically tunes the Julius decoder parameters, used as engine in an ASR system. Particularly, the proposed strategy uses genetic algorithms to reach an optimal tune by evaluation of a cost function involving AWR and RTF. The obtained results of AWR and RTF are presented confirming the effectiveness of the proposed strategy.**

***Keywords*— Genetic algorithms, Julius decoder, automatic speech recognition, parameters optimization.**

I. Introdução

Atualmente, os sistemas de reconhecimento automático de fala (*automatic speech recognition* - ASR) vêm ganhando um importante destaque e se tornando cada vez mais presentes em nosso cotidiano [1]–[3]. O sucesso das aplicações envolvendo sistemas de ASR é fortemente dependente da precisão e da velocidade do reconhecedor. Nesse contexto, após a etapa de treinamento dos modelos acústicos (MA) e modelos de linguagem (ML), do ASR (que geralmente é realizada via processamento *off-line* [4]), a etapa de decodificação, responsável pela busca da melhor sequência de palavras que represente o sinal de entrada, estabelece uma relação de compromisso entre desempenho, no sentido de taxa de precisão de acerto das palavras reconhecidas (*accuracy word rate* - AWR), e velocidade, medida pelo fator de tempo empregado para o reconhecimento (*real time factor* - RTF) [1].

Em [5], o desempenho da AWR é comparado em função do RTF para diferentes sistemas de ASR. Já em [6], sugere-se que tais medidas de performance variam em função dos modelos treinados e do conjunto de dados (*corpora*) utilizados no treinamento desses modelos. Em [7] e [8] é investigada a influência dos parâmetros, de treinamento e de decodificação (teste), nos desempenhos gerais dos sistemas de ASR.

Particularmente, a configuração manual de um sistema de ASR é uma tarefa difícil e necessita de um profundo conhecimento dos parâmetros envolvidos, assim como de uma expertise em relação a generalidade dos *corpora* de fala e de linguagem usados para o treinamento dos modelos [5]. Nesse contexto, a partir de um conhecimento *a priori*, a configuração manual dos parâmetros do decodificador é geralmente determinada sobre um ponto de operação conhecido [9]. No entanto, tais parâmetros diferem entre os decodificadores. Além disso, a dependência dos modelos e dos *corpora* implica que os resultados ótimos dos sistemas possam ser encontrados através de diferentes (múltiplas) configurações [5], [6].

Para contornar essa dificuldade, em [7], [8] e [9] são propostas estratégias de configurações automáticas dos parâmetros de um decodificador. Nos referidos trabalhos, a configuração é realizada através de algoritmos de otimização de múltiplas variáveis, empregando técnicas de otimização lineares, não lineares, estocásticas e estratégias evolutivas.

O estado-da-arte em sistemas de ASR dispõe de diversos *engines* que podem ser usados na etapa de decodificação [1]. Dentre esses *engines* destaca-se o decodificador *Julius* [10], [11], totalmente disponível em código aberto e apresentando alto desempenho para reconhecimento em tempo real.

Portanto, visando uma solução alternativa à configuração manual, este trabalho de pesquisa apresenta uma estratégia para a otimização automática dos parâmetros do decodificador *Julius*, usado como *engine* no sistema de ASR aqui desenvolvido. Particularmente, a estratégia adotada utiliza algoritmos genéticos (AG) [12], [13] e avalia uma função custo que relaciona AWR e RTF. A estratégia usando AG é contrastada com a estratégia de configuração manual. O *corpora* de fala TIMIT [14] é usado para a análise de desempenho do sistema. Dessa forma, resultados de AWR e RTF são avaliados e comprovam a eficácia da estratégia adotada.

II. Sistema de ASR

Um sistema típico de ASR é composto por dois principais blocos de processamento, o *front-end* e o *back-end*. Na etapa de *front-end*, o sinal de entrada é recebido e o processo

Ênio dos Santos Silva e Alberto Luiz Wiggers, LABICON – Laboratório de Instrumentação e Controle, Engenharia de Controle e Automação, Instituto Federal de Santa Catarina (IFSC), Chapecó, SC, Brasil, e-mails: enio.silva@ifsc.edu.br; albertolww@yahoo.com.br.

de extração das matrizes de observação **O**, contendo informações codificadas da fala, é efetuado. Na etapa de *back-end*, é realizada a decodificação das matrizes de observação **O** e processada a "busca" das informações linguísticas, por exemplo, fonemas e/ou palavras, contidas no sinal de fala [2]. Particularmente, o bloco referente ao *back-end* é composto pelos blocos de dicionário fonético, MA, ML e pelo decodificador, conforme ilustrado na Figura 1.

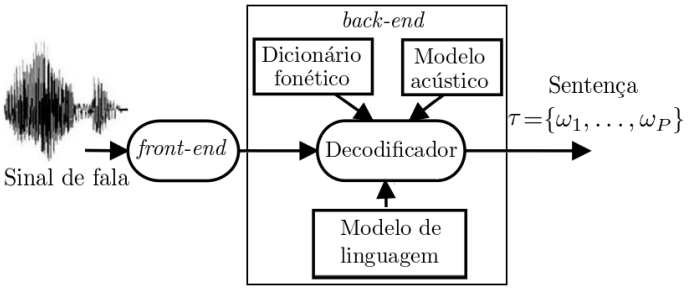

Fig. 1. Diagrama de blocos de um sistema de ASR.

Conceitualmente, o decodificador visa encontrar a sentença τ^* que maximize a probabilidade *a posteriori* dada por

$$\tau^* = \arg\max_{\tau} p(\tau|\mathbf{O}) = \arg\max_{\tau} \frac{p(\mathbf{O}|\tau)p(\tau)}{p(\mathbf{O})} \qquad (1)$$

onde $p(\mathbf{O}|\tau)$ representa a verossimilhança acústica (*likelihood*) entre a matriz de observação **O** e as palavras da sentença τ. Essa verossimilhança é determinada por um MA previamente treinado [2], [4]. Já $p(\tau)$ representa a probabilidade da sentença τ ocorrer dado o histórico de palavras já reconhecidas (e.g. N-grama). Essa probabilidade é determinada pelo ML também previamente treinado [4]. Visto que $p(\mathbf{O})$ não depende de τ, (1) é equivalente a

$$\tau^* = \arg\max_{\tau} p(\mathbf{O}|\tau)p(\tau). \qquad (2)$$

Geralmente, os sistemas de ASR usam estruturas de dados, tais como árvores lexicais hierárquicas, quebrando sentenças em palavras e palavras em unidades básicas como fones ou trifones [1]. O mapeamento das palavras para as unidades básicas e vice-versa é realizado através de um dicionário fonético [15].

A. Decodificação do Sinal de Fala

Resumidamente, após o treinamento dos modelos, o ASR na fase de reconhecimento usa o *front-end* para converter o sinal de entrada em matrizes discriminativas de observação **O** e o *back-end* para estimar a sequência de palavras (e/ou fonemas) τ mais compatível à matriz **O**, isto é, estimar a sentença τ^* que maximiza a *likelihood* (vide Equação (2)). No entanto, o tempo necessário para a realização da "busca" pela melhor sequência de palavras τ depende de diversos fatores. Dentre eles, os seguintes fatores se destacam.

- **Plataforma**: Depende da velocidade e da quantidade de memória do processador. Esse fator pode ser aprimorado usando *hardwares* mais recentes, *hardwares* dedicados ou usando paralelismo de processadores;
- **Modelos**: Dependem do tamanho do dicionário e da estratégia de treinamento dos ML e MA. Tais fatores influenciam na AWR e no RTF do sistema de ASR;
- **Algoritmo de decodificação**: Depende da eficiência nos cálculos das probabilidades $p(\tau|\mathbf{O})$ e $p(\tau)$. Esse fator pode ser melhorado através de implementações de algoritmos aproximados que permitam a restrição dos espaços de busca durante os cálculos;
- **Parâmetros de decodificação**: Dependem do algoritmo de decodificação usado. Tais parâmetros podem apresentar diversas configurações e, assim, resultarem em diferentes níveis de desempenho.

Os fatores supracitados são fortemente interdependentes e, portanto, uma mudança de um fator pode exigir uma mudança de outros fatores para produzir novamente um ponto ótimo de operação. Especialmente, os fatores referentes aos modelos e aos parâmetros do algoritmo de decodificação têm alta correlação, resultando em diferentes pontos ótimos de operação para diferentes configurações [6], [9].

B. O Decodificador Julius

O Julius [10], [11] é um decodificador de alto desempenho para reconhecimento de fala. Apresenta estratégia de decodificação em "duas passadas", isto é, a decodificação é feita em dois estágios conhecidos como avanço (*forward*) e retorno (*backward*) [15]. Nessa estratégia realiza-se primeiramente uma busca síncrona, no sentido do tempo, que visa facilitar uma segunda busca em sentido contrário, mais complexa e que requer maior esforço computacional. Esse método tem geralmente o efeito de acelerar a busca da passada de retorno, dado que o número de hipóteses a serem exploradas é bastante reduzido pela busca no sentido de avanço [15]. No primeiro passo (*forward*), é utilizado o algoritmo de busca baseada na largura efetiva de quadros síncronos (*frame synchronous beam search*) com um modelo de linguagem bi-grama. No segundo passo, é utilizado um modelo de linguagem N-grama (onde N > 2), em geral utiliza-se um tri-grama, onde a busca é realizada no sentido inverso (*backward*) através do algoritmo de busca baseada em decodificação de pilhas (*stack decoding search*) [15]. A Figura 2 apresenta a estrutura interna do decodificador Julius.

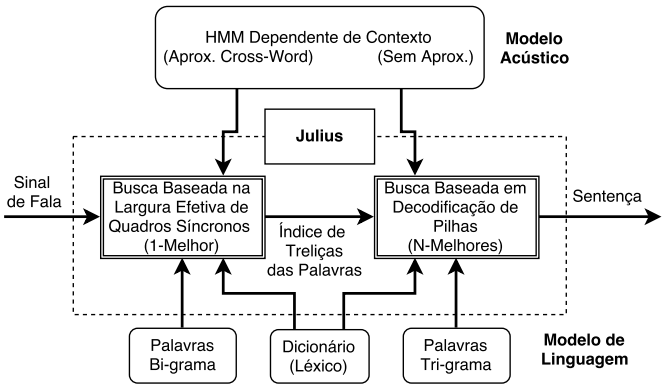

Fig. 2. Diagrama da estrutura interna do decodificador *Julius*.

Especificamente, para o reconhecimento de palavras conectadas, os MAs geralmente baseados em modelos escondidos de Markov (*hidden Markov models* - HMM) (representando fonemas ou trifones) são concatenados e, assim, formam as palavras. Tais palavras também são concatenadas e resultam na formação das frases. Em geral, tem-se uma rede formada por um grande conjunto de HMMs conectados [4], [15]. As probabilidades de transições, entre os estados de um HMM e transições entre as palavras, são determinadas respectivamente pelo MA e ML. Então, o espaço de solução da melhor sequência de palavras que represente o sinal de fala de entrada é expandido de maneira similar a uma árvore de treliças, onde a cada nó são encontradas novas possibili-

dades de sequências de palavras, isto é, novos caminhos [10], [11].

A cada nó da treliça, as probabilidades de cada caminho (τ, hipóteses) são calculadas através das probabilidades de transições a_{ij} e de emissões $b_j(\mathbf{O})$ dos estados de um HMM [1], [4]. A partir de um nó, quando existem múltiplos caminhos a serem seguidos $(\tau_1, \tau_2, ..., \tau_n)$, uma cópia da treliça é realizada e todos os caminhos são explorados em paralelo. Conforme as transições na treliça, as probabilidades de transição entre estados a_{ij} e as probabilidades de emissão dos estados $b_j(\mathbf{O})$ (no sentido da *likelihood*) são incrementadas e associadas a uma pontuação [2], [15]. Para evitar a busca por todos os caminhos possíveis, o número de nós é limitado e, assim, no final de cada passo, os N melhores caminhos (τ com maiores probabilidades *likelihood*) são mantidos. Esse processo é denominado de *pruning*, sendo realizado a cada passo com o objetivo de eliminar os caminhos (hipóteses) que apresentem pontuações abaixo de um limiar previamente definido. No *Julius*, tal limiar é definido pelos parâmetros B, BS, B2 e SB (apresentados na Seção IV-B). Mais detalhes sobre o processo de decodificação podem ser vistos em [10], [11].

III. Otimização via Algoritmo Genético - AG

Os AGs podem ser descritos como métodos numéricos de otimização baseados nos processos naturais de evolução e recombinação genética. Esses algoritmos simulam a evolução de uma espécie efetuando processos naturais de sobrevivência e reprodução das populações, de acordo com a adaptação dos indivíduos ao longo das gerações. Nesse contexto, as primeiras descrições sobre adaptação vêm da biologia, especialmente da teoria de seleção natural de *Charles Darwin*. Particularmente, o termo adaptação designa qualquer processo pelo qual uma estrutura seja modificada progressivamente para obter melhores desempenhos no seu ambiente [12], [13]. Em resumo, os principais termos associados aos AGs são descritos como segue:

- **Gerações**. Quantidade de iterações executadas;
- **População**. Conjunto de indivíduos (cromossomos);
- **Indivíduo (Cromossomo)**. Cadeia de genes. Representa uma solução do problema;
- **Gene**. Unidade básica do indivíduo (cromossomo). Representa uma variável do problema;
- **Operações genéticas**. Operações realizadas em cada indivíduo (cromossomo);
- **Espaço de busca**. Conjunto de soluções possíveis;
- **Função custo**. Nota da aptidão (desempenho) de cada indivíduo (cromossomo) na população.

Nesse processo de evolução genética, na tentativa de identificar uma solução ótima, os cromossomos mais aptos são selecionados a cada geração para se unirem uns aos outros e assim gerarem seus descendentes. Então, o conjunto resultante dos cromossomos (os descendentes) substituem a geração anterior, sendo possível que cromossomos (pais) especialmente aptos produzam relativamente mais descendentes e que certos cromossomos (membros de uma geração) sobrevivam (ou sejam clonados) para as próximas gerações. Para a avaliação do processo evolutivo, é necessário uma métrica pela qual a aptidão de um cromossomo possa ser determinada objetivamente. Tal métrica é aqui denominada como função custo.

O processo de otimização por AG é constituído principalmente pelas etapas de geração da população, da avaliação dos indivíduos, do processo de seleção e da execução de operadores genéticos que modificam os indivíduos de uma população. Especificamente, os operadores genéticos são definidos como cruzamento e mutação. Na operação genética de cruzamento, genes de diferentes cromossomos são cruzados (permutados) enquanto que na operação de mutação, genes isolados são modificados aleatoriamente de acordo com os seus correspondentes espaços de busca (possíveis valores a serem assumidos para cada gene).

IV. Implementação da Estratégia de Otimização

Neste trabalho de pesquisa, a fim de encontrar uma solução ótima para o problema da configuração dos parâmetros do decodificar *Julius*, uma estratégia usando AG foi implementada. O AG é configurado para realizar 40 gerações, contendo, em cada geração, uma população de 8 indivíduos. Inicialmente, a população é criada de maneira aleatória, sem qualquer conhecimento *a priori*. Em seguida, todos os indivíduos da população são testados e avaliados conforme a função custo (detalhes na Seção IV-C).

Na primeira geração, todos os indivíduos são considerados aptos e assim selecionados para a etapa de geração de uma nova população, efetuada através de operações genéticas. Especificamente, no processo de operação genética seis indivíduos são sorteados para a operação de cruzamento (entre pares) e dois indivíduos são designados para a operação de mutação, na qual a quantidade de genes mutados e a definição dos novos valores, são definidos aleatoriamente de acordo com uma distribuição uniforme. Sugere-se aqui que a seleção dos pares de indivíduos para o processo de cruzamento também é realizada por distribuição uniforme, descartando a probabilidade de indivíduos mais aptos gerarem mais descendentes. Tal procedimento foi adotado com o objetivo de manter a diversidade da população ao longo das gerações, visto que, se mantivéssemos a descendência sendo proporcional à aptidão de cada indivíduo, um grande número de cópias de um pequeno número de bons indivíduos poderia ser realizada, diminuindo a diversidade da população e podendo ocasionar em uma convergência prematura do algoritmo para uma solução não desejada. A Figura 3 ilustra a estratégia do AG para a configuração automática dos parâmetros do decodificar Julius.

A. Estratégia de Seleção Elitista

Especialmente em problemas de otimização, um procedimento frequentemente empregado é o elitismo. Neste procedimento, os melhores indivíduos de uma geração são preservados, isto é, suas cópias (clones) são passadas diretamente para a geração seguinte, garantindo que esses indivíduos não sejam destruídos nas etapas de cruzamento e mutação.

Após a avaliação da segunda geração, é iniciado o processo de seleção elitista dos dois melhores indivíduos de todas as gerações. Tal procedimento garante a permanência dos dois melhores indivíduos ao longo das gerações. Na literatura, o modelo de seleção elitista normalmente é acompanhado de outros métodos de seleção [13]. Aqui, a seleção elitista é seguida pelo método de seleção por ordenação linear, selecionando apenas os seis melhores melhores da geração corrente para comporem (juntamente com os dois melhores de todas as gerações) o conjunto dos oito indivíduos mais aptos. Tais indivíduos sofrem as operações genéticas supracitadas, de cruzamento e mutação, para que uma nova população seja criada.

O quadro a seguir apresenta o algoritmo implementado.

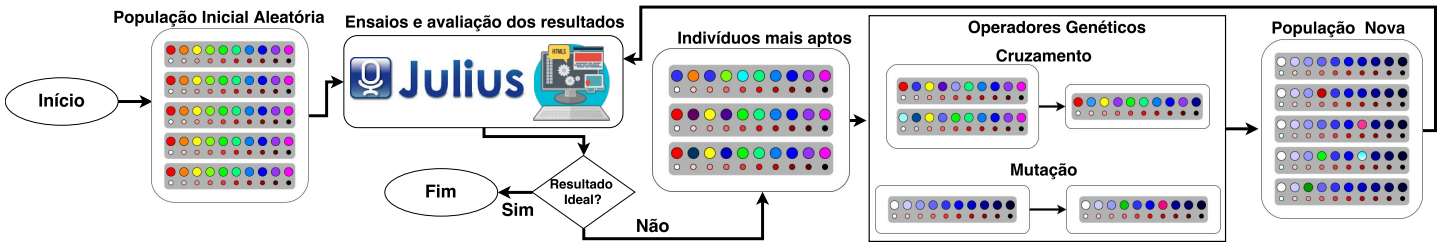

Fig. 3. Estratégia do AG para a configuração automática dos parâmetros do decodificar *Julius*.

```
Algoritmo Genético Implementado
    Gera a população inicial
        Gera 8 indivíduos aleatoriamente
        Atribui uma nota de aptidão aos
            indivíduos
    Repete (evolução das gerações)
        Seleciona os 2 melhores indivíduos
            dentre todas as gerações
        Seleciona os 6 melhores indivíduos
            da última geração
        Recebe os 8 indivíduos selecionados
            Sorteia 6 indivíduos para o
                cruzamento
            Envia os outros 2 indivíduos
                para a mutação
            Gera a nova população via
                operações genéticas
        Avalia a nova população
            Atribui uma nota de aptidão aos
                novos indivíduos
    Continua até a quadragésima geração
Fim
```

B. Definição do Espaço de Busca

Após a definição sobre quais parâmetros serão considerados como genes de um indivíduo, deve-se estimar uma escala de abrangência que tais parâmetros possam assumir. Tal escala é denominada de espaço de soluções ou espaço de busca, devendo abranger inúmeras possibilidades de respostas e preferencialmente conter possíveis soluções ótimas. A seguir, são apresentados os principais parâmetros do decodificador *Julius* considerados neste trabalho, assim como o espaço de busca de cada parâmetro.

- **TMIX**. Seleção de misturas de gaussianas para a aceleração do cálculo da *likelihhod* $p(\mathbf{O}|\tau)$ no MA. Diretamente proporcional a AWR e RTF. Variação de 2 a 8.
- **IWCD1 BEST** (IB). Para a 1ª passada, aproxima o MA trifone *cross-word* pelo valor médio das N melhores *likelihood* de trifones *interword* de mesmo contexto. A relação com AWR e RT depende da qualidade do MA. Variação de 2 a 8.
- **LMP** e **LMP2**. Atribui o peso do ML e a penalidade de inserção de palavras na 1ª e 2ª passada, respectivamente. A relação com AWR e RT depende da qualidade do ML. Variação de 5 a 25 para o peso do ML; e de 5 a 20 para a penalidade de inserção de palavras.
- **B**, **BS**, **B2** e **SB**. Conforme discutido na Seção II-B, tais parâmetros definem os números de nós e o limiar de corte (pontuação) das treliças dos HMMs na 1ª e 2ª passada. Sendo B e B2 associados ao número de nós; e BS e SB associados ao limiar de corte da 1ª e 2ª passada respectivamente. São diretamente proporcionais a AWR e RTF. B varia de 500 a 5000, BS de 20 a 500, B2 de 50 a 1000 e SB de 60 a 80.
- **N**. Define o número de sentenças candidatas para a avaliação final da máxima *likelihood*[1]. Diretamente proporcional ao RTF. A relação com AWR depende da qualidade do MA. Variação de 1 a 7.
- **LOOKUPRANGE** (LK). Define o número de quadros antes e depois nos quais são realizadas a expansão de palavras. Diretamente proporcional ao RTF. A relação com a AWR depende da qualidade do MA e ML. Variação de 2 a 10.
- **S**. Número máximo de hipóteses que podem ser armezenadas durante o processo de busca. Diretamente proporcional a AWR e RTF. Aqui, assumiu-se o valor padrão de 500.
- **M**. Número de hipóteses expandidas necessárias para interromper o processo de busca. Diretamente proporcional a AWR e RTF. No entanto, é geralmente utilizado como fator de segurança para evitar falhas nas buscas. Aqui, assumiu-se o valor padrão de 2000.

C. Função Custo Adotada

Neste artigo, os valores de AWR e RTF são considerados para a formulação da função de aptidão

$$\eta = \text{AWR} - \alpha \times \text{RTF} \quad \forall\, \alpha = 2,5\%. \tag{3}$$

Sendo RTF calculado pela divisão entre o tempo utilizado pelo sistema, para o reconhecimento de todas as frases testadas, e a duração real das frases.

O valor de AWR é obtido conforme a Equação (4)

$$\text{AWR} = \frac{\text{R} - \text{I}}{\text{T}} \times 100\%, \tag{4}$$

onde T é o número total de palavras da sentença testada, R é o número de palavras reconhecidas corretamente e I o número de palavras inseridas além de T.

Ressalta-se que o parâmetro α foi definido como 2,5% para representar uma perda de 2,5% em η para cada acréscimo de 1 em RTF.

V. Resultados e Análise de Desempenho

Para os testes do sistema de ASR, foi adotado o *corpora* TIMIT [14], do qual foram utilizadas 4400 sentenças para treinamento ($\approx 3,75$ horas) e 1680 para teste ($\approx 1,5$ horas). No *frontend*, foram usados sinais de fala amostrados a 16 kHz convertidos para 38 coeficientes cepstrais em escala Mel mais a energia de cada quadro de 20 ms (vide [2]).

Para o treinamento do MA, a ferramenta HMM *toolkit* (HTK) [4] foi adotada. Os MAs foram gerados com base em HMMs contínuos de 5 estados com topologia esquerda-direita, constituídos por modelos dependentes do contexto (trifones *cross-word*) computados a partir de 47 monofones, de acordo com as etapas descritas em [2] e [4]. A partir de todas as frases presentes no *corpora* de treinamento

[1]No *Julius*, devido as aproximações, não é estritamente garantido que o candidato com a máxima *likelihood* seja o primeiro.

e usando a ferramenta MITLM [2], estimou-se um ML trigrama com a técnica de suavização de *Kneser-Ney* [1].

Após o treinamento do MA e ML, o sistema de ASR foi testado com o decodificador *Julius* usando duas estratégias de configuração. Inicialmente, uma configuração manual foi realizada com base no conhecimento *a priori* dos autores. Em seguida, uma nova configuração, sem qualquer conhecimento *a priori*, foi realizada conforme a estratégia de otimização usando AG descrita na Seção IV, os resultados são apresentados na Figura 4.

Na Figura 4(a), a função de aptidão (função custo η) é avaliada, apresentando os envelopes superior e inferior, o valor médio e a variância ao longo das gerações. Nota-se, da Figura 4(a), que a aptidão máxima da população é crescente até a convergência para uma solução ótima. A Figura 4(b) apesenta as curvas de desempenho de AWR e RTF, considerando os melhores indivíduos de cada geração. Dessa figura, observa-se que a partir da 10ª geração é possível encontrar valores satisfatórios de AWR e RTF. Já a Figura 4(c) apresenta a distribuição de AWR e RTF para todos os indivíduos do processo, com destaque especial para os melhores indivíduos de cada geração. Finalmente, na Figura 4(d) é ilustrada a concentração dos resultados em torno dos valores de pico das gerações.

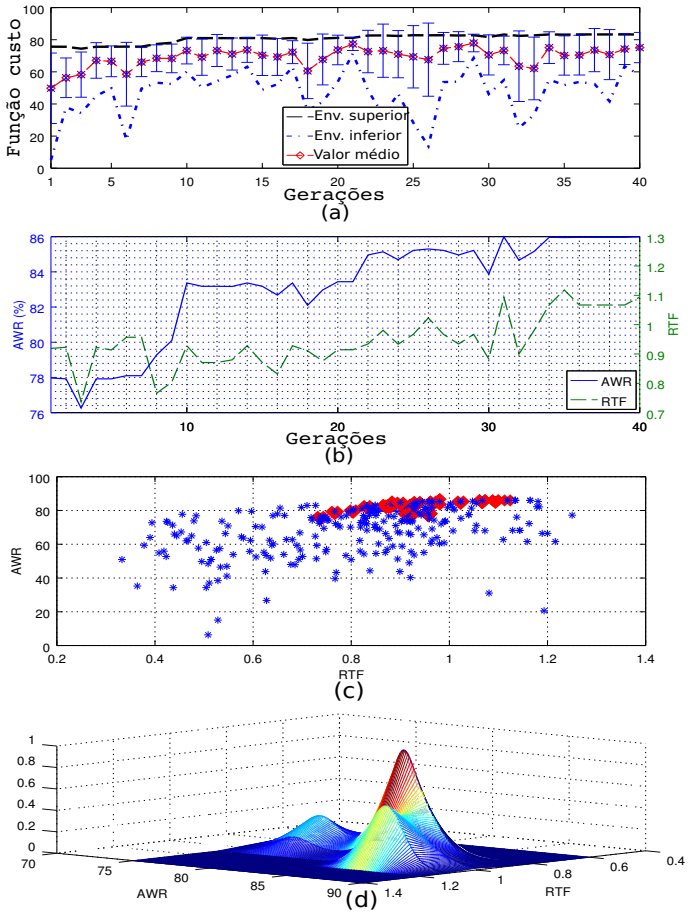

Fig. 4. Curvas de desempenho do AG: (a) Função custo; (b) AWR e RTF; (c) Distribuição de AWR vs. RTF; (d) Valores de pico.

Especificamente, o melhor indivíduo, a partir da 34ª geração, apresenta uma nota de aptidão igual a 83,29%, atribuída por uma AWR de 85,96% e um RTF de 1,067, ou seja, a solução encontrada pelo AG resultou em um sistema rápido e preciso. Já o desempenho do sistema de ASR usando a configuração manual resultou em um desempenho inferior, com uma menor AWR e em um maior RTF. A Tabela I apresenta os indivíduos selecionados e os seus correspondentes desempenhos nos sistemas de ASR para as duas diferentes estratégias de configuração.

TABELA I
COMPARAÇÃO ENTRE ESTRATÉGIAS DE CONFIGURAÇÃO

Parâmetros otimizados manualmente									
TMIX	IB	LMP	B	BS	LMP2	B2	SB	N	LK
8	3	22 10	3000	200	22 10	350	300	3	2
Parâmetros otimizados via AG									
TMIX	IB	LMP	B	BS	LMP2	B2	SB	N	LK
4	5	22 13	4250	340	24 12	850	420	5	9

Desempenho	**AWR**(%)	**RTF**	η
Otimização manual	82,38	1,8563	77,739
Otimização via AG	**85,96**	**1,0670**	**83,292**

VI. CONCLUSÕES E COMENTÁRIOS FINAIS

Neste trabalho de pesquisa, uma estratégia de otimização dos parâmetros do decodificar *Julius* foi apresentada. Essa estratégia implementou um AG eficiente capaz de encontrar uma configuração ótima que satisfez o desempenho do sistema de ASR tanto no sentido de AWR quanto no sentido de RTF, tornando o sistema veloz e preciso. Tal estratégia foi contrastada com a otimização manual e obteve como resultado uma ganho no desempenho, apresentando um aprimoramento de 3,58% na AWR e uma redução de 42,52% no RTF. Os resultados das avaliações objetivas apresentados na Figura 4 e na Tabela I ratificam tais afirmações e confirmam a eficácia da estratégia implementada.

REFERÊNCIAS

[1] D. Yu and L. Deng, *Automatic Speech Recognition: A Deep Learning Approach (Signals and Communication Technology)*. New York, USA: Springer, 2015.

[2] E. Silva and D. Silva, "Desenvolvimento de um sistema robótico controlado por reconhecimento automático de fala com adaptação ao locutor," in *Proc. of the 12th IEEE/IAS Int. Conf. on Ind. App. (INDUSCON)*, Curitiba, PR, Brasil, Nov. 2016, pp. 2258–2266.

[3] J. Kennedy, S. Lemaignan, C. Montassier, P. Lavalade, B. Irfan, F. Papadopoulos, E. Senft, and T. Belpaeme, "Child speech recognition in human-robot interaction: evaluations and recommendations," in *Proc. of the 12th A. ACM Int. Conf. on Human-Robot Interaction (HRI'17)*, Vienna, Austria, 2017, pp. 52–61.

[4] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book for HTK V3.4*. Cambridge, ENG: Cambridge University Press, 2006.

[5] A. V. Ivanov, P. L. Lange, D. Suendermann-Oeft, V. Ramanarayanan, Y. Qian, Z. Yu, and J. Tao, "Speed vs. accuracy: Designing an optimal asr system for spontaneous non-native speech in a real-time application," in *Proc. of 7th Int. Work. Spoken Dialogue Systems IWSDS*, Saariselk, Finland, 2016.

[6] A. E. Hannani and T. Hain, "Data dependence of decoder search parameters," Department of Computer Science, University of Sheffield, Sheffield, UK, Tech. Rep., 2010.

[7] S. Watanabe and J. Le Roux, "Black box optimization for automatic speech recognition," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Florence, ITA, 2014, pp. 3256–3260.

[8] T. Le Nguyen, D. Stein, and M. Stadtschnitzer, "Gradient-free decoding parameter optimization on automatic speech recognition," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Florence, ITA, 2014, pp. 3261–3265.

[9] A. E. Hannani and T. Hain, "Automatic optimization of speech decoder parameters," *IEEE Signal Process. Letters*, vol. 17, no. 1, pp. 95–98, 2010.

[10] A. Lee, T. Kawahara, and K. Shikano, "Julius—an open source real-time large vocabulary recognition engine," 2001.

[11] A. LEE, *The Julius Book*, 1st ed., rev.4.1.5, Ed., Kyoto, JP, 2010.

[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Addison-Wesley Longman Publishing Co, 1989.

[13] B. Coppin, *Artificial Intelligence Illuminated*. Jones & Bartlett Learning, 2004.

[14] C. Lopes and F. Perdigao, "Phone recognition on the timit database," *Speech Technologies/Book*, vol. 1, pp. 285–302, 2011.

[15] C. P. Silva, "Um software de reconhecimento de voz para português brasileiro," Master's thesis, Pós-graduação em Engenharia Elétrica, Universidade Federal do Pará, Belém, PA, Brasil, 2010.