PEDRO HENRIQUE BALISTIERO FATTORE

ALGORITMOS METAHEURISTICOS PARA PROJETO E OTIMIZAÇÃO DE FONTES DE TENSÃO DE REFERÊNCIA

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo

Curso de Engenharia da Computação

ORIENTADOR: Prof. Doutor João Navarro Soares Jr.

São Carlos

2014

F252a

Fattore, Pedro Henrique Balistiero
Algoritmos metaheuristicos para projeto e otimização de Fontes de Tensão de Referência / Pedro Henrique Balistiero Fattore; orientador João Navarro Soares Junior. São Carlos, 2014.

Monografia (Graduação em Engenharia de Computação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2014.

1. Circuitos Integrados MOS. 2. Fonte de tensão de referência. 3. Algoritmos metaheurísticos. I. Título.

FOLHA DE APROVAÇÃO

Nome: Pedro Henrique Balistiero Fattore

Título: "Algoritmos metaheurísticos para projeto e otimização de Fontes de Tensão de Referência"

Trabalho de Conclusão de Curso defendido em 21 / 11 / 2014.

Comissão Julgadora:	**Resultado:**
Prof. Dr. João Navarro Soares Júnior (Orientador) - SEL/EESC/USP	Aprovado
Prof. Dr. Maximiliam Luppe SEL/EESC/USP	Aprovado
Mestre Marcelo Patricio de Santana Doutorando - SEL/EESC/USP	Aprovado

Coordenador do Curso Interunidades - Engenharia de Computação:

Prof. Associado Evandro Luís Linhari Rodrigues

Sumário

Resumo

Fontes de tensão de referência têm a função de fornecer uma tensão constante independente das circunstâncias às quais o circuito pode estar sujeito, como variação da temperatura e tensão de alimentação. Projetar esses circuitos é, no entanto, algo bem complexo e exige vasto conhecimento. Este trabalho utiliza Algoritmo Genético, *Simulated Annealing*, *Particle Swarm* e uma variação do *Particle Swarm* proposta, junto com simulações elétricas, para projetar e otimizar fontes de tensão de referência, comparando os resultados entre as configurações e, se possível, tentar determinar se alguma delas é o melhor para as fontes testadas. A variação do *Particle Swarm* proposta aplica redes neurais para antecipar resultados de simulação. Embora não tenha sido encontrado o melhor algoritmo para os circuitos testados, foi mostrado que a variação do *particle swarm* proposta apresenta resultados promissores.

Palavras-chave: Circuitos integrados MOS, Fonte de tensão de referência, Algoritmos metaheurísticos.

Abstract

Reference voltage sources have the function of providing a constant voltage regardless of the circumstances to which the circuit may be subject, for instance temperature and supply voltage variances. However designing these circuits is something quite complex and requires a extensive knowledge. This work uses Genetic Algorithm, *Simulated Annealing*, *Particle Swarm* and a proposed variation of *Particle Swarm*, with electrical simulations, to design and optimize the reference voltage sources, comparing the results between them and, if possible, trying to determine which one is the best for all tested sources. The proposed variation of *Particle Swarm* applies neural networks to preview the simulation results. Although a best algorithm for the tested circuits has not been found, it was shown that the proposed variation of the *Particle Swarm* presents promising results.

Keywords: MOS integrated circuits, voltage reference, metaheuristic algorithms.

1 Introdução

1.1 Contextualização

Circuitos integrados são extensivamente utilizados em produtos eletrônicos atuais, sendo encontrados em notebooks, celulares, *tablets* e afins. Com a introdução dos transistores CMOS (do inglês *Complementary Metal-Oxide-Semiconductor*) nos circuitos integrados, foi possível melhorá-los em vários aspectos, tais como consumo de potência, facilidade de projeto, dimensões, custos de produção além da comunicação entre blocos analógicos e digitais.

Muitos circuitos, para que possam trabalhar corretamente, precisam de uma grandeza de referência, corrente ou tensão, de boa qualidade, cujas características se mantenham com a temperatura, variações de alimentação, variações nos processos de fabricação e tempo. Para muitos desses circuitos, essa grandeza é fornecida por uma fonte de tensão de referência; um circuito que fornece uma tensão estável, independente das circunstancias às quais o circuito está sujeito.

Geralmente, fontes de tensão de referência são compostas de dois blocos de circuitos com funcionamentos complementares, em relação à temperatura, para uma determinada grandeza, tensão ou corrente. Um dos blocos deve gerar uma grandeza diretamente proporcional à temperatura, PTA (do inglês *Proportional to Absolute*) e o outro, uma grandeza inversamente proporcional, CTA (do inglês *Complementary to Absolute*).

Para obtermos a fonte de tensão operando corretamente, devemos fazer com que a soma dessas duas grandezas anule as variações da grandeza PTAT (*Proportional to Absolute Temperature)* com as variações da grandeza CTAT (*Complementary to Absolute Temperature)* em um determinado ponto. Dessa forma, o resultado é uma grandeza constante e independente da temperatura, como mostrado na **Figura 1** a seguir:

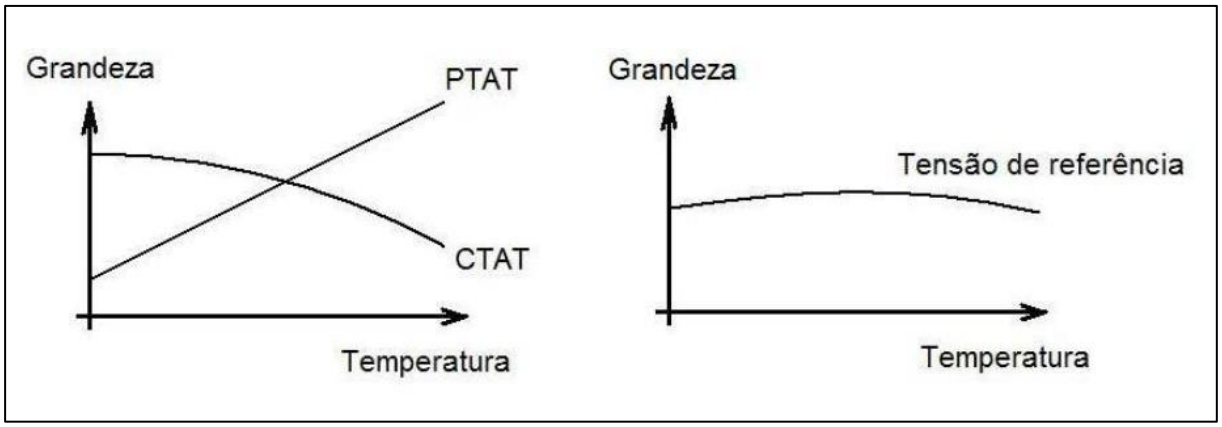

Figura 1. Variação das grandezas PTAT e CTAT de acordo com a temperatura, na esquerda, e grandeza resultante, na direita.

Desenvolver uma fonte de tensão de referência é algo complicado e exige conhecimentos avançados em circuitos eletrônicos. Essa mesma dificuldade aparece no projeto da maior parte dos circuitos analógicos, que exigem, normalmente, projetistas experientes. Uma opção para resolver esta dificuldade é a aplicação de algum meio automático para projeto e otimização. Alguns desses meios utilizam algoritmos metaheuristicos [8]. Estes algoritmos utilizam formas randômicas de geração de parâmetros para tentar encontrar a melhor solução possível para um determinado problema. Esse processo, onde se gera randomicamente soluções, é chamado de otimização estocástica.

Este trabalho consiste no projeto de circuitos de fontes de tensão de referência, utilizando como base circuitos desenvolvidos por Ueno [6] e por Ishibe [7], na tecnologia CMOS 0,35 μm da AMS (AustriaMicroSystem) [2]. No projeto são aplicados mecanismos metaheuristicos e simulações elétricas. Também são feitas comparações entre projetos/otimizações realizadas por diversas metaheuristicas para determinar as dimensões dos componentes, buscando alcançar o melhor funcionamento possível do circuito.

1.2 Objetivos do Trabalho

Este trabalho consiste em realizar o projeto e otimização de determinados circuitos de fonte de tensão de referência, por meio de diversos algoritmos metaheuristicos. Adicionalmente, configurações das metaheuristicas são alteradas visando aprender sobre quais algoritmos apresentam melhores resultados em termos tanto de velocidade de otimização e quanto de qualidade final do resultado.

1.3 Organização da Monografia

A Seção 2 contém alguns conceitos sobre transistores MOS e bipolar, sobre funcionamento de fontes de tensão de referência, funcionamento dos algoritmos utilizados e sobre as simulações e avaliação dos circuitos.

Na Seção 3 apresentam-se os resultados e estes são discutidos.

Na Seção 4 está a conclusão.

1.4 Ferramentas utilizadas

No trabalho foram utilizados:

- MatLab [9]: Ferramenta utilizada para executar o *software* de projeto e otimização, realizar os cálculos e exibir os gráficos;
- HSpice [11]: Software utilizado para realizar as simulações elétricas dos circuitos.

2 Conceitos utilizados

Existem muitas topologias utilizadas em fontes de tensão de referência, com diferentes dispositivos e configurações. As utilizadas neste trabalho se baseiam em transistores MOS e bipolar.

Segundo Silva em [4], durante muito tempo foi utilizado apenas os transistores bipolares na construção de circuitos eletrônicos. A partir da década de 70, com a melhora da tecnologia MOS e eliminação de algumas dificuldades na sua construção, os transistores MOS começaram a ganhar destaque, pois permitiam a construção de circuitos digitais mais simples e de menor consumo, além de permitir dimensões menores.

Com a diminuição das dimensões dos transistores houve o aumento na complexidade dos circuitos integrados, o que ocasionou também o aumento nas dificuldades de projeto. Portanto, é útil e importante que se desenvolvam formas confiáveis de projeto e otimização de circuitos micro eletrônicos.

2.1 Transistor Bipolar

O transistor bipolar é composto de uma estrutura de três regiões de cristais semicondutores, com duas regiões do mesmo tipo intercaladas por uma região do tipo oposto. Os tipos se referem ao tipo N, onde prevalecem elétrons na condução, e tipo P, onde prevalecem de lacunas. As regiões faladas anteriormente são conhecidas como Emissor, Base e Coletor. A **Figura 2** ilustra os dois tipos de transistores bipolares existentes, o transistor NPN e o transistor PNP.

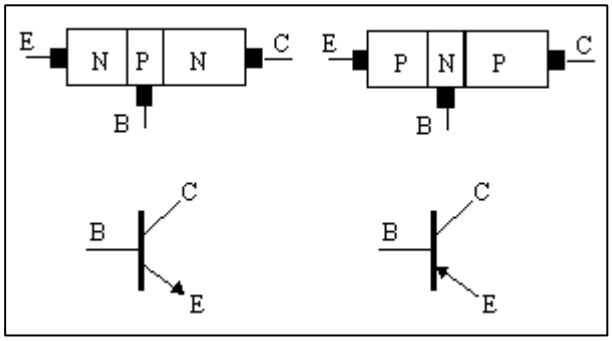

Figura 2. Ilustração dos transistores bipolares NPN, à esquerda, e do PNP, à direita com seus símbolos.

O Emissor é a região responsável por fornecer portadores de carga, elétrons, ou negativas. A Base é a região intermediária, que absorve uma pequena parte dos portadores. O Coletor é a região que recolhe a maior parte dos portadores emitidos pelo emissor. O transistor NPN fornece elétrons e o transistor PNP fornece lacunas. Essa transferência de portadores gera as correntes do transistor, compostas por correntes de base, de emissor e de coletor, como ilustrada na **Figura 3**.

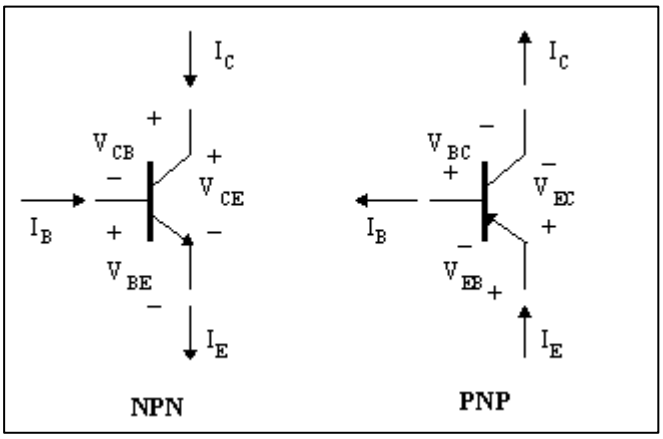

Figura 3. Tensões e correntes de um transistor bipolar.

É verificado no transistor bipolar que uma variação da corrente de base gera uma variação diretamente proporcional na corrente do coletor. É possível determinar relações das correntes e tensões aplicadas em um transistor, além do ganho corrente entre as correntes de base e de coletor. De acordo com Sedra em [1], em um transistor bipolar as seguintes relações são verificadas:

$$I_e = I_b + I_c \quad (1)$$

$$\beta = \frac{\Delta I_c}{\Delta I_b} \quad (2)$$

NPN: $V_{ce} = V_{be} + V_{cb}$ (3)

PNP: $V_{ec} = V_{eb} + V_{bc}$ (4)

$$I_c = I_s . e^{\frac{qV_{be}}{kT}} \quad (5)$$

onde I_e é a corrente do emissor, I_b é a corrente da base, I_c é a corrente do coletor, Δ indica variação de algum elemento, q é a carga do elétron, k é a constante de Boltzmann, T é a temperatura em kelvin, β é uma constante de valor alto (acima de 100) e I_s é a corrente de saturação do transistor.

O índice V_{xy} é a diferença de potencia entre dois pontos, ou seja, tensão em x menos a tensão em y. No transistor, a diferença entre tensões de base, emissor e coletor.

Sabendo que tanto I_s quanto V_{be} dependem da temperatura e fazendo a relação de duas correntes de coletor em temperaturas diferentes, [6], podemos escrever:

$$\frac{I_c(T)}{I_c(T_2)} = \left(\frac{I_s(T).\, e^{\frac{qV_{be}(T)}{kT}}}{I_s(T_2).\, e^{\frac{qV_{be}(T_2)}{kT_2}}} \right) \tag{6}$$

$$V_{be}(T) = T.\left[\frac{V_{be}(T_2)}{T_2} + \frac{k}{q}.\ln\left(\frac{I_c(T)}{I_c(T_2)}.\frac{I_s(T_2)}{I_s(T)} \right) \right] \tag{7}$$

A fórmula da corrente I_s é dada por:

$$I_s = \frac{q.A.n_i^2.\overline{D}}{N_B} \tag{8}$$

onde A é a área de junção base-emissor, n_i a concentração de portadores, $\overline{D}$ o valor da difusão efetiva dos portadores minoritários da base e N_B o número de Gummel, que é o total de impurezas por unidade de área na base.

Substituindo (8) em (7), temos:

$$V_{be}(T) = T.\left[\frac{V_{be}(T_2)}{T_2} + \frac{k}{q}.\ln\left(\frac{I_c(T)}{I_c(T_2)}.\frac{n_i^2(T_2).\overline{D}(T_2)}{n_i^2(T).\overline{D}(T)} \right) \right] \tag{9}$$

A fórmula de n_i^2 é dada por:

$$n_i^2(T) = E.T^3.e^{\left(-q.\frac{V_{GO}(T)}{k.T}\right)} \tag{10}$$

onde E é uma constante, que depende da massa de elétrons e lacunas e V_{GO} é a tensão de *bandgap* do silício.

Substituindo (10) em (9), temos:

$$V_{be}(T) = T.\left[\frac{V_{be}(T_2)}{T_2} - \frac{V_{Go}(T_2)}{T_2} + \frac{V_{Go}(T)}{T} + \frac{k}{q}.\ln\left(\frac{I_c(T)}{I_c(T_2)}.\frac{{T_2}^3.\overline{D}(T_2)}{T^3.\overline{D}(T)} \right) \right] \tag{11}$$

A relação de Einstein, que relaciona a constante de difusão com a mobilidade, e a fórmula que relaciona a mobilidade com temperatura são dadas respectivamente por:

$$\bar{\mu}(T) = \frac{q.\bar{D}(T)}{k.T} \quad (12)$$

$$\bar{\mu}(T) = C.T^{-\eta} \quad (13)$$

onde C e η são constantes resultantes da relação de Einsten, e η depende do processo de fabricação do transistor.

Aplicando (12) e (13) em (11), temos:

$$V_{be}(T) = T.\left[\frac{V_{be}(T_2)}{T_2} - \frac{V_{Go}(T_2)}{T_2} + \frac{V_{Go}(T)}{T}\right] - \frac{k.T}{q}.\ln\left(\frac{T_2}{T}\right)^{4-\eta} + \frac{k.T}{q}.\ln\left(\frac{I_c(T)}{I_c(T_2)}\right) \quad (14)$$

$$V_{be}(T) = V_{Go}(T) - \frac{T}{T_2}[V_{Go}(T_2) - V_{be}(T_2)] - \frac{k.T}{q}.\ln\left(\frac{T_2}{T}\right)^{4-\eta} + \frac{k.T}{q}.\ln\left(\frac{I_c(T)}{I_c(T_2)}\right) \quad (15)$$

Por (15), dados os valores dos termos, tem-se que quando há um aumento da temperatura, mantida a corrente I_C constante, a tensão base-emissor diminuirá. Diz-se nesse caso que V_{be} é uma grandeza inversamente proporcional à temperatura CTAT. Essa relação vai se mostrar muito importante para fontes de tensão de referência.

2.2 Transistores MOS

Apenas na década de 70 a tecnologia MOS (do inglês *Metal Oxide Silicon*) começou a competir com os transistores bipolares em algumas áreas. Inicialmente foram utilizados transistores PMOS (transistores MOS com canal onde a condução é feita por lacunas). Estes possuíam um melhor desempenho, pois eram mais robustos, trabalhando melhor mesmo com problemas no óxido de porta e na interface dele com o silício, e seus circuitos tinham um funcionamento mais estável. Com o avanço da tecnologia, os problemas com o oxido foram sendo tratados e os transistores NMOS (transistores MOS com canal tipo onde a condução é feita por elétrons) se mostraram mais rápidos e eficientes. Atualmente, muitos circuitos trabalham com ambas os transistores, NMOS e PMOS, sendo chamados de circuitos CMOS. O transistor MOS tem uma estrutura semelhante à mostrada na **Figura 4**. Os símbolos usados para representá-los estão mostrados na **Figura 5**. De forma semelhante ao bipolar, o NMOS também tem três regiões, uma N (*source*), uma P (substrato ou *bulk*) e outra N (*dreno*) (ou P-N-P para o PMOS). As regiões de *source* e *dreno* são exatamente iguais; o que determina quem vai ser o *source* e o *dreno* são os potenciais à que eles estão sujeitos.

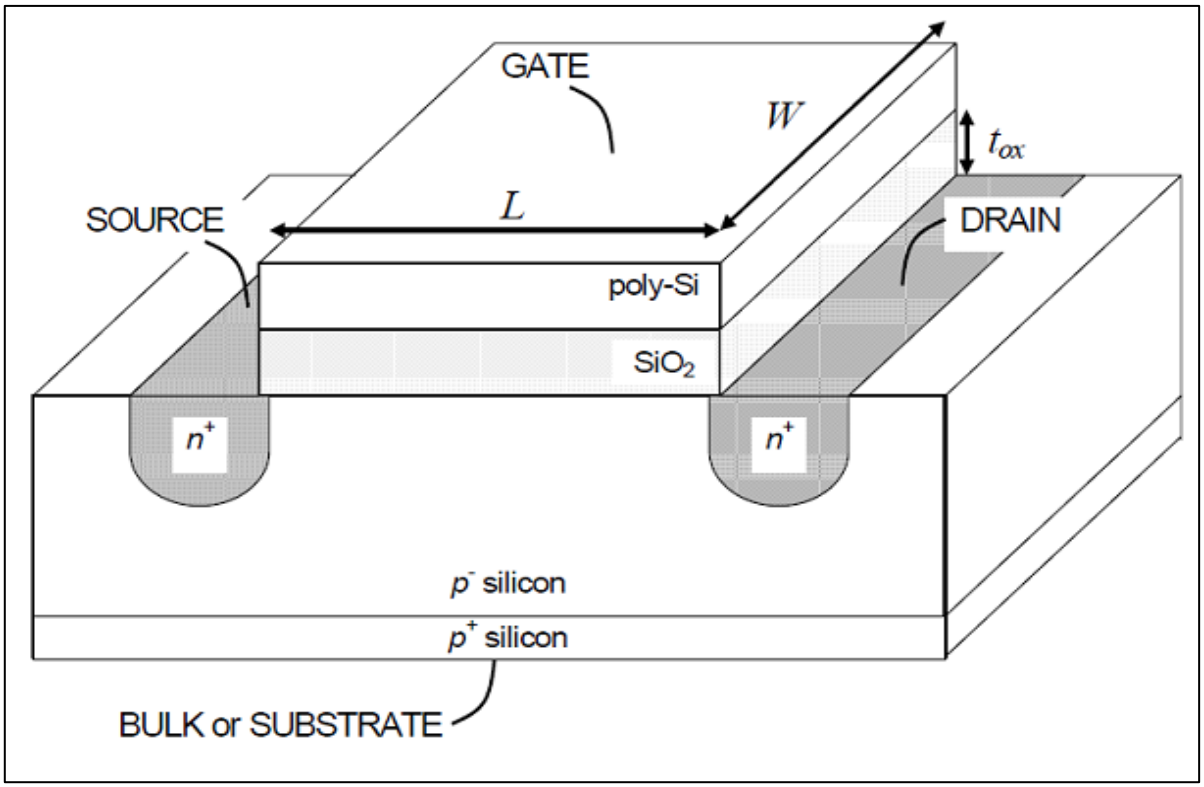

Figura 4. Transistor NMOS em corte.

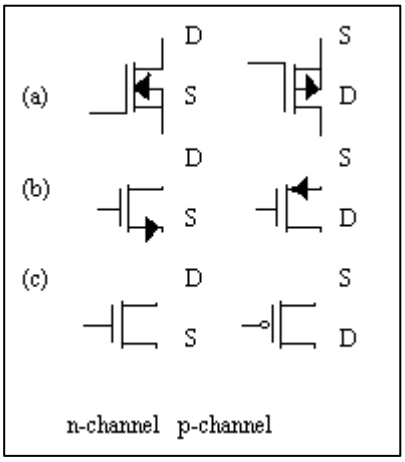

Figura 5. Símbolos de transistores NMOS, à esquerda, e PMOS, à direita.

No transistor MOS a tensão de controle é aplicada na parte intermediária, o *gate*. O *gate* é formado de um material condutor, metal ou um semicondutor policristalino muito dopado, que está entre o *source* e *dreno* e separado da região de substrato por uma camada de isolante, dióxido de silício muitas vezes – S_iO_2, isolando-o eletricamente.

Aplicando uma diferença de tensão suficientemente grande entre o *gate* e o *source*, é formado um canal de portadores, elétrons no NMOS e lacunas no PMOS, no substrato e por esse canal ocorre a passagem da corrente entre *source* e *dreno.* Essa tensão mínima para formação do canal de portadores é conhecia como Tensão de *Threshold* ou de limiar (V_t).

Os transistores MOS, dependendo da tensão entre *gate* e *source*, podem operar em quatro distintas regiões:

- Inversão Forte: nessa região, a tensão V_{gs} (entre o *gate* e o *source*) é maior que a tensão de limiar. Essa região é utilizada em projetos onde os transistores funcionarão como chaves ou para a amplificação de sinais;
- Inversão Fraca: nessa região, a tensão V_{gs} é muito próxima da tensão V_t. Essa região é utilizada quando se deseja que o transistor trabalhe com baixas potências e frequências;
- Inversão moderada: região entre a fraca e forte inversão. A operação de transistores nela não é modelada de maneira exata.
- Corte: nessa região, a tensão V_{gs} (entre o *gate* e o *source*) é menor que a tensão de limiar. A corrente que passa pelo canal é praticamente nula.

Normalmente se verifica a região de operação do transistor, Fraca, Moderada ou Forte inversão, analisando a corrente que passa no dreno. Um critério para determinar em qual região o transistor opera é apresentado na **Tabela 1**.

Tabela 1. Critério para determinar a região de operação do transistor.

Região de Operação	**Condição**
Inversão Forte	*LIM*> 10
Inversão Fraca	*LIM*< 0,1
Inversão Moderada	0,1 <*LIM*< 10

O fator *LIM* é dado por

$$\mathrm{LIM} = \frac{I_d}{\frac{W}{L}\mu . Cox . 2n\ (\mathrm{U_T})^2} \tag{16}$$

onde I_d é a corrente de dreno, n *é o* fator de inclinação de inversão fraca (seu valor depende da tecnologia e varia entre 1,2 e 1,6) e o termo $U_T = \frac{kT}{q}$ é a tensão térmica.

Quando operando na região de forte inversão, um transistor MOS possui dois estados de funcionamento: saturação e triodo (**Figura 6**). Estes estados dependem das diferenças de potencial entre *source*, *dreno* e *gate*.

Para um transistor NMOS estar no estado triodo, devemos ter $V_{gs} > V_t$ e $V_{ds} < (V_{gs} - V_t)$. Este estado apresenta a corrente variando de acordo com a seguinte relação

$$I = \mu . Cox \frac{W}{L}\left[(V_{gs} - V_t)V_{ds} - \frac{1}{2}V_{ds}^2\right] \tag{17}$$

onde μ é a mobilidade dos portadores do canal, Cox é a capacitância por unidade de área formada entre o *gate* e o substrato, W é a largura do canal do transistor e L é o comprimento entre *source* e *dreno*.

Para um transistor NMOS estar no estado de saturação, devemos ter $V_{gs} > V_t$ e $V_{ds} > (V_{gs} - V_t)$. Neste estado, a corrente que passa pelo transistor se estabiliza, permanecendo constante independente do quanto se aumente a diferença de potencial entre *dreno* e *source*, como mostrado na **Figura 6**. A equação da corrente é:

$$I = \frac{\mu . Cox}{2}\frac{W}{L}(V_{gs}^2 - V_t)^2 \tag{18}$$

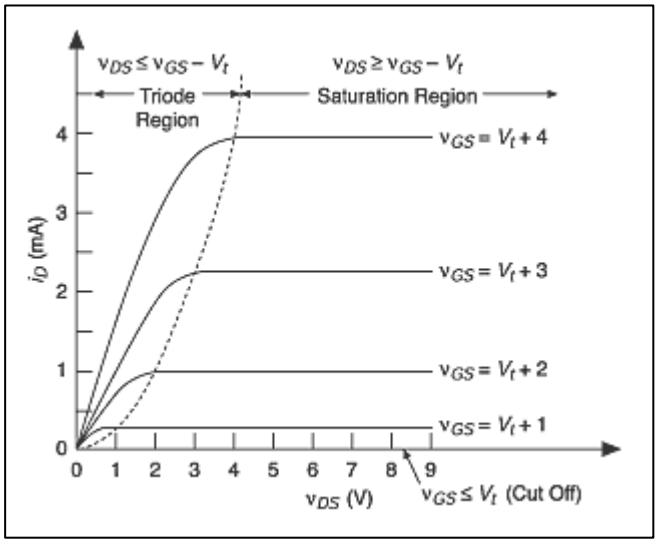

Figura 6. Gráfico de corrente de dreno versus V_{ds} em um transistor NMOS com diferentes valores de V_{gs}.

Quando operando na região de fraca inversão, a corrente de dreno de um transistor se dá pela seguinte fórmula:

$$I = \left(\frac{W}{L}\right) I_{D0} e^{\frac{V_{gb}}{nU_T}} \left(e^{\frac{-V_{sb}}{U_T}} - e^{\frac{-V_{db}}{U_T}} \right) \quad (19)$$

onde que I_{D0} é uma constante da tecnologia com dimensão de corrente e o índice b das tensões é referente ao *bulk* do transistor.

A tensão V_{gs} do transistor pode ser descrita como [5]:

$$V_{gs}(\mathrm{T}) = V_{TH}(\mathrm{T}) + V_{OFF} + \frac{n(T)}{n(T_2)} \left(V_{gs}(T_2) - V_{TH}(T_2) - V_{OFF} \right) \quad (20)$$

onde T_2 é uma temperatura de referência e V_{OFF} é uma constante obtida pelo modelo de simulação. Assumindo V_{TH} como escrito em (21), poderemos reescrever (20) como (22):

$$V_{TH}(T) = V_{TH}(T_2) + K_T \left(\frac{T}{T_2} - 1 \right) \quad (21)$$

$$V_{gs}(\mathrm{T}) \approx V_{gs}(T_2) + K_G \left(\frac{T}{T_2} - 1 \right) \quad (22)$$

com $K_G \cong K_T + V_{gs}(T_2) - V_{TH}(T_2) - V_{OFF}$, onde K_T é uma constante negativa. Quando o transistor está em fraca inversão, K_G também será um fator negativo, fazendo com que a tensão V_{gs} varie inversamente à temperatura, apresentando um comportamento CTAT.

2.3 Fonte de corrente

Uma estrutura importante para polarização e construção de fontes de tensão de referência é a fonte de corrente. Existem diversos tipos de fontes de corrente, com características variadas, mas uma fonte bastante utilizada é a fonte mostrada na **Figura 7** [3]:

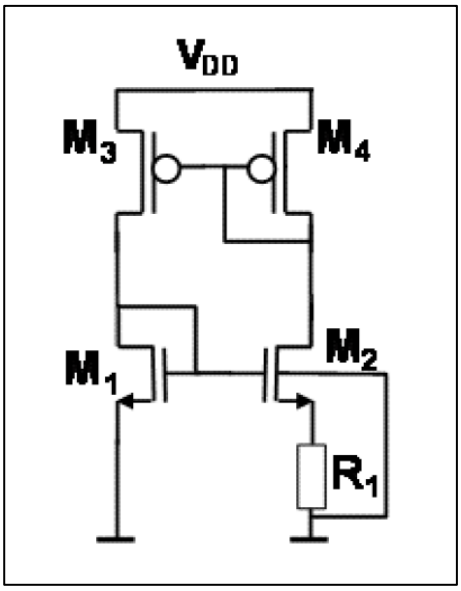

Figura 7. Fonte de corrente CMOS proporcional à temperatura.

Para que esta fonte opere corretamente, os transistores M_1e M_2 devem estar em fraca inversão. Sendo assim e utilizando (19) encontramos:

$$I_3 = I_1 = (\frac{W_1}{L_1})\, I_{D0} e^{\left(\frac{V_{g1}}{nU_T}\right)} \tag{23}$$

$$I_4 = I_2 = \left(\frac{W_2}{L_2}\right) I_{D0} e^{\left(\frac{V_{g2}}{nU_T} - \frac{-V_{S2}}{U_T}\right)} \tag{24}$$

onde I_{n} é a corrente de dreno, W_{n} largura de canal, e L_{n} comprimento de canal do transistor M_n.

Como as tensões de *gate* dos transistores M_1e M_2 são iguais e a tensão de *source* de M_2 é igual à queda de tensão do resistor, V_R, temos:

$$\frac{I_3}{I_4} = \frac{I_1}{I_2} = \frac{\left(\frac{W_1}{L_1}\right) I_{D0} e^{\left(\frac{V_{g1}}{nU_T}\right)}}{\left(\frac{W_2}{L_2}\right) I_{D0} e^{\left(\frac{V_{g2}}{nU_T} - \frac{-V_{S2}}{U_T}\right)}} = \frac{\left(\frac{W_1}{L_1}\right)}{\left(\frac{W_2}{L_2}\right)} e^{\left(\frac{V_R}{U_T}\right)} \tag{25}$$

Supondo que a relação (W/L) do transistor M_3 seja M vezes maior que (W/L) do M_4, consequentemente $I_3 = MI_4$, e sabendo que a queda de tensão no resistor é dada por $V_R = I_2.R_1$, temos:

$$I_2 = \frac{U_T}{R_1}.ln\left(\frac{\frac{W_2}{L_2}}{\frac{W_1}{L_1}}\mathrm{M}\right) \tag{26}$$

Com (26) vemos que a corrente gerada pela fonte de corrente é uma grandeza proporcional à temperatura absoluta, PTAT.

2.4 Circuitos utilizados

Existem muitos fatores a se considerar ao projetar e analisar uma fonte tensão de referência, tais como coeficiente de temperatura, consumo de potência, regulação de linha, área ocupada, entre outros. Cada um desses fatores influencia no dimensionamento dos dispositivos da fonte de tensão.

Coeficiente de temperatura quantifica o quanto a tensão de saída varia com variações da temperatura do circuito. Uma relação para o coeficiente de temperatura é

$$\mathbf{CT} = \frac{V_{MAX}-V_{MIN}}{T_{MAX}-T_{MIN}}\frac{1}{V_{REF}}10^6 \tag{27}$$

onde V_{MAX} é o máximo valor da tensão de saída para as variações da temperatura, V_{MIN} é o mínimo valor da tensão de saída, T_{MAX} e T_{MIN} são, respectivamente, a máxima e a mínima temperaturas que o circuito está sujeito e V_{REF} o valor desejado para a saída. CT é dado em *ppm* (partes por milhão) por unidade de temperatura.

A regulação de linha quantifica o quanto a tensão de saída vai variar com variações da tensão de alimentação. Uma relação para a regulação de linha é

$$\mathbf{RL} = \frac{V_{MAX}-V_{MIN}}{V_{DMAX}-V_{DMIN}}\frac{1}{V_{REF}}10^6 \tag{28}$$

onde V_{DMAX} é o máximo valor da tensão de alimentação e V_{DMIN} é o mínimo valor da tensão de alimentação. RL é dado em ppm/V.

Dependendo da qualidade da fonte de referência, CT e RL devem ser bem reduzidos. Valores tais como CT=20 ppm/^{0}C e RL= 500 ppm/V caracterizam uma boa fonte de tensão.

Os circuitos de fonte de tensão escolhidos para este trabalho foram: Circuito Ishibe em [6], Circuito Ueno completo em [7] e Circuito Ueno Modificado, que é, como sugere o nome, uma variação do circuito proposto por Ueno. Estes circuitos apresentam boas características e representam três topologias diferentes. A seguir são apresentados os circuitos na **Figura 8**, **Figura 9** e **Figura 10**, respectivamente, e é feita uma breve explicação do funcionamento deles.

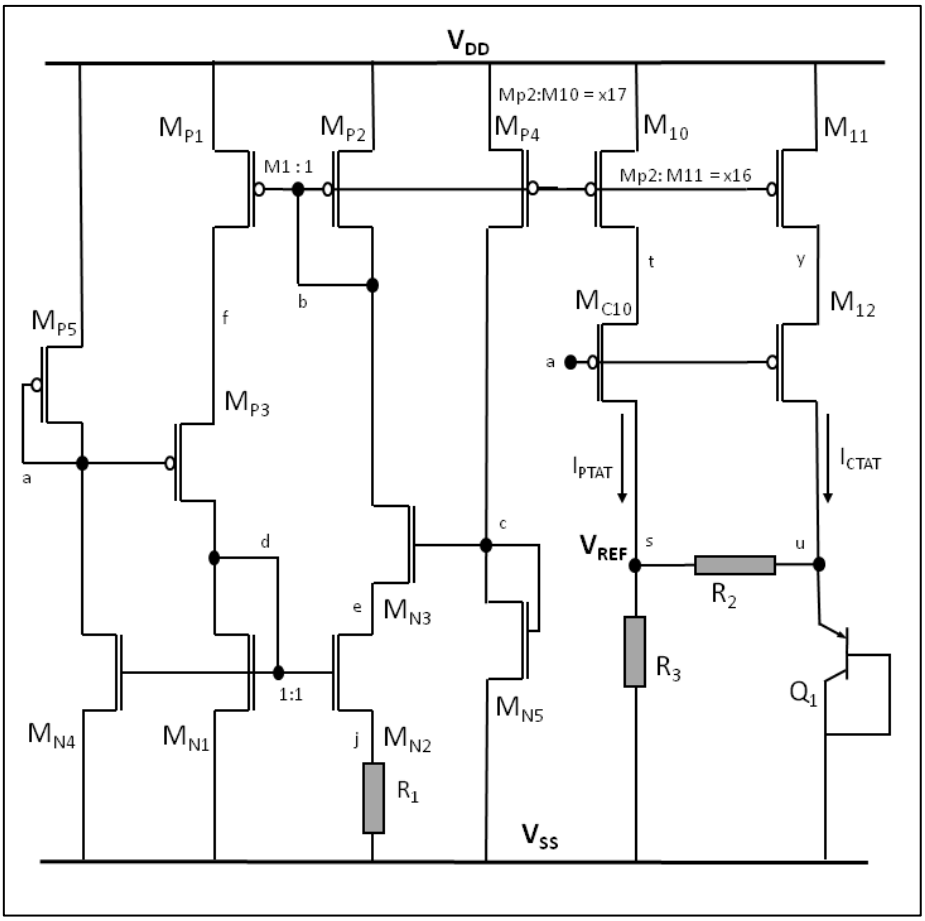

Figura 8. Circuito de Fonte de Tensão Ishibe.

O Circuito fonte de tensão Ishibe, **Figura 8**, mantém a tensão de saída constante equilibrando a soma de uma corrente proporcional à temperatura I_{PTAT}, gerada pela fonte de corrente e espelhada pelo transistorM_{10}, com outra corrente inversamente proporcional I_{CTAT} gerada a partir do transistor bipolar Q_1. A soma das duas correntes tende a variar muito pouco, garantindo pouca variação na queda de tensão no resistor R_3.

Ele utiliza uma variação da fonte de corrente mostrada anteriormente, composta pelos transistores M_{p1}, M_{p2}, M_{p3}, M_{n1}, M_{n2} e M_{n3} e o resistor R_1. Esta corrente gerada pela fonte de corrente é espelhada pelo transistor M_{10} e somada à corrente em R_2 que é proporcional a (V_{REF} – V_{eb}). A corrente resultante, passando pelo resistor R_3, pode ser ajustada para ser constante com a temperatura, acarretando em V_{REF} uma tensão praticamente constante.

Os transistores M_{p3} e M_{n3} são transistores *cascode* e foram adicionados para que a corrente na fonte de corrente permaneça praticamente constante quando houver variações

na tensão de alimentação V_{dd}. Os pares de transistores M_{p4}-M_{n5} e M_{p5}-M_{n4} foram adicionados para polarizar o circuito, de modo a fornecer tensões que garantam o funcionamento correto dos transistores *cascode*.

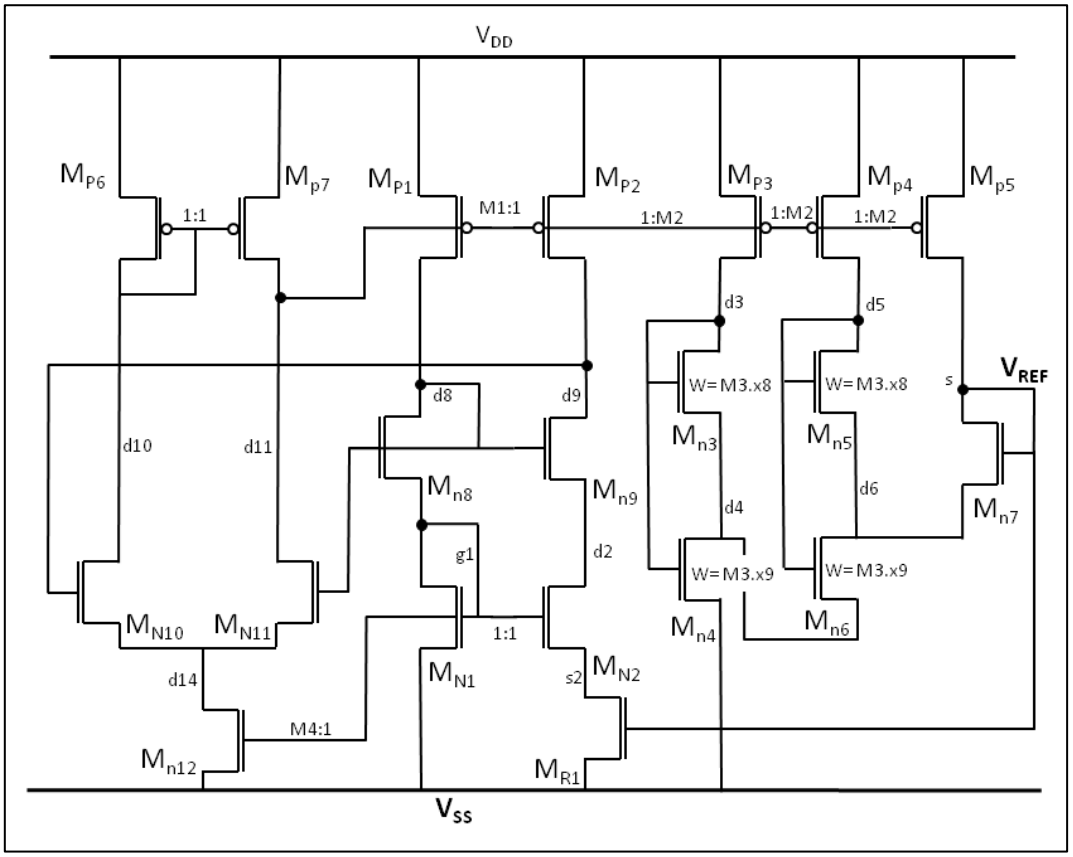

Figura 9. Circuito de Fonte de Tensão Ueno completo.

O Circuito fonte de tensão Ueno completo, **Figura 9**, utiliza uma variação da fonte de corrente apresentada. Nela o resistor é substituído por um transistor NMOS operando na região triodo. Com isso, podem-se obter menores correntes sem a necessidade de grandes áreas. Também foi utilizado um amplificador diferencial, composto pelos transistores M_{p6}, M_{p7}, M_{n10}, M_{n11} e M_{n12}, visando manter as tensões V_{gd} dos transistores M_{p1} e M_{p2} iguais e diminuir a interferência que alterações na tensão de alimentação V_{dd} possam causar na tensão gerada.

Nessa fonte de tensão não é utilizado um transistor bipolar para gerar a corrente CTAT. Ao invés disso, ele monta uma estrutura composta apenas de transistores MOS e estes são responsáveis por gerar a tensão CTAT, e a tensão de referência se dá na

combinação das diferenças de tensões entre *gate* e *source* dos transistoresM_{n3}a M_{n7}, como mostrado nas fórmulas a seguir [5]:

$$V_{REF} = V_{gs4} - V_{gs3} + V_{gs6} - V_{gs5} + V_{gs7} \tag{29}$$

Todos esses transistores devem estar operando em fraca inversão. Aplicando-se (19) e substituindo suas respectivas tensões *gate-source*, temos:

$$V_{REF} = V_{gs4} + nV_T \left[-\ln\left(\frac{I_{D3}}{\left(\frac{W}{L}\right)_3 I_0}\right) + \ln\left(\frac{I_{D6}}{\left(\frac{W}{L}\right)_6 I_0}\right) - \ln\left(\frac{I_{D5}}{\left(\frac{W}{L}\right)_5 I_0}\right) + \ln\left(\frac{I_{D7}}{\left(\frac{W}{L}\right)_7 I_0}\right)\right] \tag{30}$$

onde I_{Di} é a corrente de dreno e $(W/L)_i$ é a relação (W/L) do transistor M_{ni}.

Sabendo que as correntes espelhadas em M_{p3}, M_{p4} e M_{p5} são iguais, chamaremos de I_M, temos que as correntes que passam por M_{n3}, M_{n5}, M_{n6} e M_{n7} são, respectivamente, I_M, I_M, $2I_M$ e I_M, podemos simplificar (30) por:

$$V_{REF} = V_{gs4} + nV_T ln\left(\frac{2\left(\frac{W}{L}\right)_3\left(\frac{W}{L}\right)_5}{\left(\frac{W}{L}\right)_6\left(\frac{W}{L}\right)_7}\right) \tag{31}$$

Substituindo agora a tensão *gate-source* do transistor M_{n4}, sabendo que a corrente que passa por ele é $3I_M$, temos:

$$V_{REF} = nV_T ln\left(\frac{3I_M}{\left(\frac{W}{L}\right)_4 I_0}\right) + nV_T ln\left(\frac{2\left(\frac{W}{L}\right)_3\left(\frac{W}{L}\right)_5}{\left(\frac{W}{L}\right)_6\left(\frac{W}{L}\right)_7}\right) \tag{32}$$

Sabendo que I_0 pode ser descrita como (33), podemos reescrever (32) como:

$$I_0 = I_{D0} e^{\left(-\frac{V_{TH}}{nV_T}\right)} \tag{33}$$

$$V_{REF} = nV_T ln\left(\frac{3I_M}{\left(\frac{W}{L}\right)_4 I_{D0} e^{\left(-\frac{V_{TH}}{nV_T}\right)}}\right) + nV_T ln\left(\frac{2\left(\frac{W}{L}\right)_3\left(\frac{W}{L}\right)_5}{\left(\frac{W}{L}\right)_6\left(\frac{W}{L}\right)_7}\right) \tag{34}$$

$$V_{REF} = V_{TH} + nV_T ln\left(\frac{6\left(\frac{W}{L}\right)_3\left(\frac{W}{L}\right)_5 I_M}{\left(\frac{W}{L}\right)_6\left(\frac{W}{L}\right)_7\left(\frac{W}{L}\right)_4 I_{D0}}\right) \tag{35}$$

Aplicando-se (21), temos:

$$V_{REF}(T) = V_{TH}(T_2) - K_T + T\left(\frac{K_T}{T_2} + \frac{n\mathrm{k}}{\mathrm{q}} ln\left(\frac{6\left(\frac{W}{L}\right)_3\left(\frac{W}{L}\right)_5 I_M}{\left(\frac{W}{L}\right)_6\left(\frac{W}{L}\right)_7\left(\frac{W}{L}\right)_4 I_{D0}}\right)\right) \quad (36)$$

Ueno [7] desenvolveu esta fonte buscando o coeficiente de temperatura mais baixo possível. Como buscamos um TC tendendo a zero quando T está próximo de T_2, devemos ter que $\frac{\partial V_{REF}(T)}{\partial \mathrm{T}}\Big|_{T=T_2} = 0$. Sendo assim, temos:

$$V_{REF}(T) = V_{TH}(T_2) - K_T \quad (37)$$

Como os dois termos são constantes, a tensão de saída da fonte de corrente também é constante, porém não pode ser configurada, dependendo de $V_{TH}(T_2)$ na tecnologia dos componentes usados na fonte.

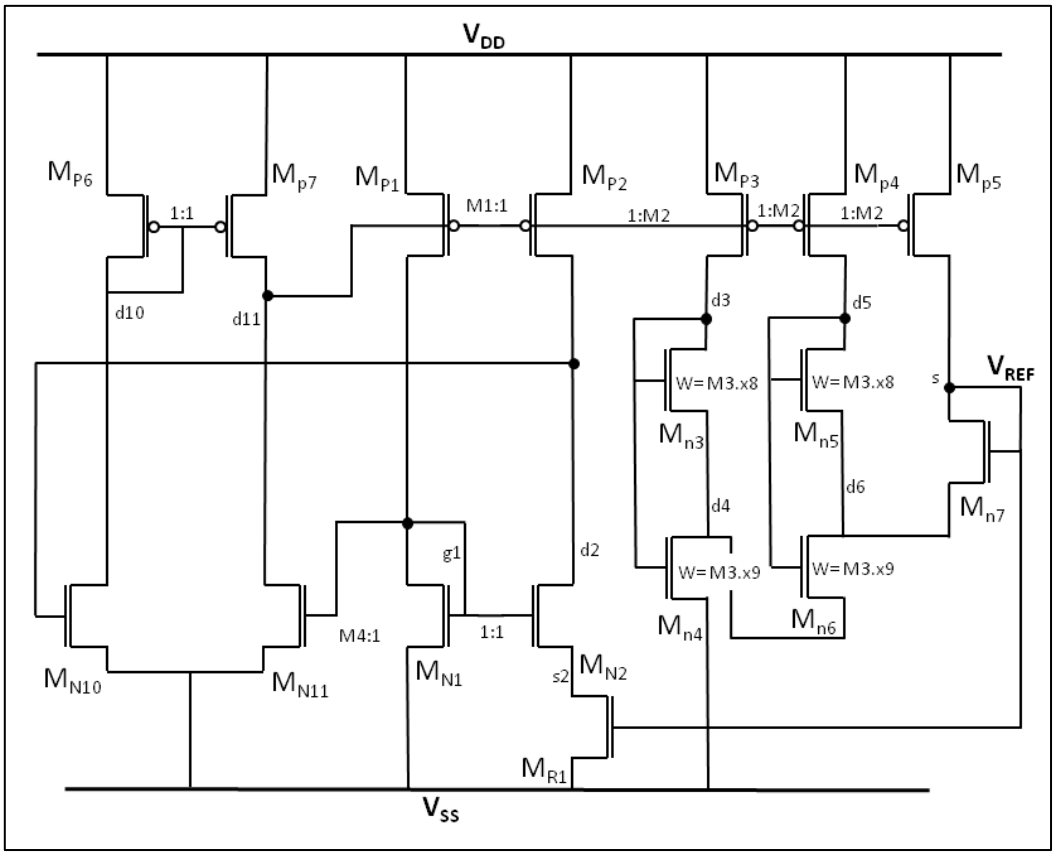

Figura 10. Circuito de Fonte de Tensão Ueno modificado.

O circuito fonte de tensão Ueno modificada (**Figura 10**) tem a mesma configuração e funcionamento do circuito anterior em relação ao subcircuito de tensão, trabalhando com as

relações das tensões V_{gs} dos transistores M_{n3} a M_{n7}. A diferença deste circuito para o anterior está na fonte de corrente. Neste caso o transistor M_{n12} foi retirado do amplificador diferencial. Também foram retirados os transistores cascode M_{n8} e M_{n9}. Ambas as alterações foram realizadas para permitir que a fonte de corrente opere com tensões mais baixas de alimentação. Por outro lado ele terá maior dependência com a tensão de alimentação.

Os circuitos são parametrizados, de modo que as dimensões dos dispositivos são representadas por variáveis que serão determinadas. Atribuem-se intervalos de valores para essas variáveis. A descrição de um transistor, por exemplo, ficaria como a seguir:

```
Mn2   d2   g1   s2   0   MODN W='X6*1u'   L='X5*1u'
```

onde Mn2 é o nome do transistor, d2, g1, s2 e 0 são os nós onde estão conectados dreno, gate, source e bulk dos transistores, MODN quer dizer que é um transistor NMOS, W é a largura do transistor, parametrizado pela variável X6 e L é o comprimento do gate, parametrizado pela variável X5.

As descrições completas dos circuitos estão no Apêndice A, Apêndice B e Apêndice C. A **Tabela 2**, **Tabela 3** e **Tabela 4** mostram quais dimensões dos transistores são parametrizadas por quais variáveis nos circuitos Ishibe, Ueno completo e Ueno modificado, respectivamente:

Tabela 2. Relações entre as variáveis de parametrização e as dimensões dos componentes do Circuito Ishibe.

Variáveis	**Dimensões**
X1	Lp1, Lp2, Lp4, L10, L11
X2	Lp3, Lc10, L12
X3	Lp5
X4	Ln1, Ln2, Ln4
X5	Ln3
X6	Ln5
X7	Wp1, Wp2, Wp3, W10, W11, Wc10, W12
X8	Wp4
X9	Wp5
X10	Wn1, Wn2, Wn3
X11	Wn4
X12	Wn5
X13	R1
X14	Cálculo de R2
X15	Cálculo de R3
X16	Relação de tamanho entre Mp2 e M11
X17	Relação de tamanho entre Mp1 e M10

Tabela 3. Relações entre as variáveis de parametrização e as dimensões dos componentes do Circuito Ueno completo.

Variáveis	Dimensões
X1	Lp1, Lp2, Lp3, Lp4, Lp5
X2	Wp1, Wp2, Wp3, Wp4, Wp5
X3	Lp6, Lp7
X4	Wp6, Wp7
X5	Ln1, Ln2, Ln12
X6	Wn1, Wn2, Wn8, Wn9, Wn12
X7	Ln3, Ln4, Ln5, Ln6, Ln7
X8	Multiplicador de Mn3 e Mn5
X9	Multiplicador de Mn4 e Mn6
X10	Wn7
X11	Ln8, Ln9
X12	Ln10, Ln11
X13	Wn10, Wn1
X14	Lr1
X15	Wr1

Tabela 4. Relações entre as variáveis de parametrização e as dimensões dos componentes do Circuito Ueno modificado.

Variáveis	Dimensões
X1	Lp1, Lp2, Lp3, Lp4, Lp5
X2	Wp1, Wp2, Wp3, Wp4, Wp5
X3	Lp6, Lp7
X4	Wp6, Wp7
X5	Ln1, Ln2, Ln10, Ln11
X6	Wn1, Wn2, Wn10, Wn11
X7	Ln3, Ln4, Ln5, Ln6, Ln7
X8	Multiplicador de Wn3 e Wn5
X9	Multiplicador de Wn4 e Wn6
X10	Wn7
X11	Lr1
X12	Wr1

Para avaliar os circuitos existe uma função, a função *Fitness*, que dá notas aos circuitos com base em resultados de simulação. São também definidas situações de funcionamento, como tensões mínima e máxima de alimentação, variação de temperatura, e também valores alvos para outros parâmetros de desempenho, como tensão de saída, consumo, regulação de linha, entre outros. Para cada um desses parâmetros, foi definido

um peso. A *Fitness* utiliza esses pesos e os valores reais, comparando-os com os valores desejados, para gerar a nota.

2.5 Algoritmos

Os algoritmos metaheuristicos escolhidos para projeto e otimização foram: Algoritmo Genético (GA), *Simulated Annealing* (SA), *Particle Swarm Optimization* (PSO) e PSO com rede neural (PSON). A seguir, uma descrição do funcionamento destes algoritmos, baseado em [8], e como foram utilizados nas simulações.

O Algoritmo Genético, como o próprio nome sugere, é baseado na teoria de cruzamento e seleção das espécies, buscando sempre o melhor indivíduo. Consideram-se uma ou mais populações com determinado números de indivíduos cada e trabalha-se com a reprodução entre os indivíduos para geração de novas populações. Para este trabalho, cada indivíduo é uma possível fonte de tensão e seus genes são as variáveis de otimização procuradas.

No GA o programa gera uma população de circuitos, com dimensões para todos os dispositivos, simula-os, atribui notas de acordo com os parâmetros de desempenho escolhidos e salva o circuito que apresentou o melhor resultado. Em seguida, ele gera uma nova população de circuitos a partir de alguns dos circuitos passados como pais, utilizando os genes destes, que são copiados, cruzados e sofrem mutação, para criar circuitos filhos. A nova população é simulada novamente e novas notas são atribuídas. Caso haja um novo melhor resultado, ele é armazenado e o antigo descartado, caso contrário, o antigo é mantido, e o processo de criar populações e simular é repetido até que se atinja o número máximo de simulações determinado.

O problema desse algoritmo é conseguir balancear o número de indivíduos e de populações. O número total de indivíduos deve ser grande o suficiente para garantir uma boa convergência e distribuído em um certo número de populações de forma manter a variedade nas dimensões dos circuitos, mas não exagerado, de modo a realizar simulações excessivas e repetidas. Em contrapartida, ele é um método robusto, garantindo um resultado geral satisfatório.

Para o algoritmo genético foram alteradas duas características para analises: a quantidade de indivíduos em cada população e o número de populações. Foram realizadas aqui simulações com as populações de 50 e 100 indivíduos e com 1 a 4 populações distintas.

O *Simulated Annealing* funciona com base em algoritmos de Hill-Climbing [8], utilizando um parâmetro de probabilidade adicional. Partindo de um circuito inicial, que será também o circuito atual S, é feita a simulação dele e dada uma nota de acordo com os parâmetros de desempenho escolhidos. Em seguida, o programa gera um novo circuito R em torno do primeiro, com alguma pequena modificação em algumas dimensões de seus dispositivos, simula este novo circuito e o compara com o circuito S. Caso R seja melhor, o sistema faz de R o novo circuito atual S e segue com o processo, criando um novo circuito em torno de R. Caso R seja pior, ainda há uma probabilidade do algoritmo substituir S por R. A probabilidade de substituição é dada pela fórmula:

$$P(t,R,S) = e^{\frac{Quality(r)-Quality(s)}{t}} \tag{38}$$

onde *Quality*(r) é a nota obtida pelo circuito R, *Quality*(s) é a nota obtida pelo circuito S e t é a temperatura a qual eles estão submetidos.

Pela fórmula, se a nota do circuito R é muito pior, a subtração resulta em um número muito negativo, gerando uma probabilidade próxima de zero. Caso R não seja muito pior, a chance de substituir é maior. Há ainda o parâmetro t a ser considerado. Quando t é um valor grande, o resultado da probabilidade é próxima de 1,0, independente de quão ruim seja o circuito, o que acarreta uma grande chance de substituição. Isso é feito para garantir um movimento aleatório no espaço, em um primeiro momento, e evitar que ótimos locais limitem as buscas. À medida que se diminui o valor de t, as chances de trocar S por um circuito pior vão diminuindo e apenas substituição por circuitos melhores são garantidas. O Valor de t é reduzido ao longo do processo de otimização.

Para o *Simulated Annealing* foram alteradas duas características para análises: o valor inicial da temperatura t e quantos melhores resultados precisam ser encontrados para fazer o *re-annealing.* Esse *re-annealing* é necessário para retornar a temperatura ao valor inicial, pois, a partir de um determinado momento, ela é tão baixa que a chance de trocar o circuito atual se torna praticamente zero. Para limitar o mínimo da temperatura, também foi alterado o arquivo saupdates.m do Matlab, colocando um mínimo de temperatura igual a 0.0001. Foram realizadas aqui simulações com valores iniciais de t de 2.0, 1.5 e 1.0 e *re-annealing* quando encontrar 50, 75 e 100 resultados ótimos, totalizando 9 configurações distintas.

O PSO é outro algoritmo que tem como base a evolução, mas tem uma diferença básica. Ao invés de considerar a evolução genética dos indivíduos, ele é baseado no agrupamento e migração dos mesmos para buscar um melhor resultado. Utilizando os

conhecimentos mútuos e na experiência própria e de outros seres, cada indivíduo consegue decidir sua melhor solução.

Primeiro determina-se o número de partículas que se deseja. Cada uma dessas partículas representa uma fonte de tensão a ser otimizada. Em seguida, posicionam-se essas partículas em lugares aleatório no espaço, com um vetor velocidade, também aleatório. Este espaço é multidimensional e é caracterizado pelas variáveis, ou seja, cada dimensão está relacionada a uma variável Xn, logo as coordenadas físicas das partículas neste espaço determinam os valores atribuídos à cada variável.

Durante a execução do algoritmo, são armazenados:

- Os melhores lugares que cada partícula descobriu até o momento (um local por partícula);
- O melhor lugar que algum informante da partícula descobriu até o momento. Os informantes de uma partícula são outras partículas, escolhidas aleatoriamente a cada iteração;
- O melhor lugar descoberto até agora por qualquer partícula;

A cada iteração o algoritmo calcula a nota de cada partícula através da *Fitness* e atualiza os melhores locais, caso tenha encontrado algum. Ele também atualiza o vetor velocidade de todas as partículas, utilizando uma combinação dos vetores que apontam para os locais ótimos armazenados e altera a posição das partículas de acordo com seus respectivos vetores velocidades. O desempenho deste algoritmo está ligado à quantidade de partículas utilizadas, uma vez que mais partículas acarretam mais locais testados aumentando as chances de encontrar boas soluções com menos simulações, porém aumenta o tempo que demora cada simulação.

Na **Figura 11**, um esquemático, contendo ilustrações dos três algoritmos explicados anteriormente:

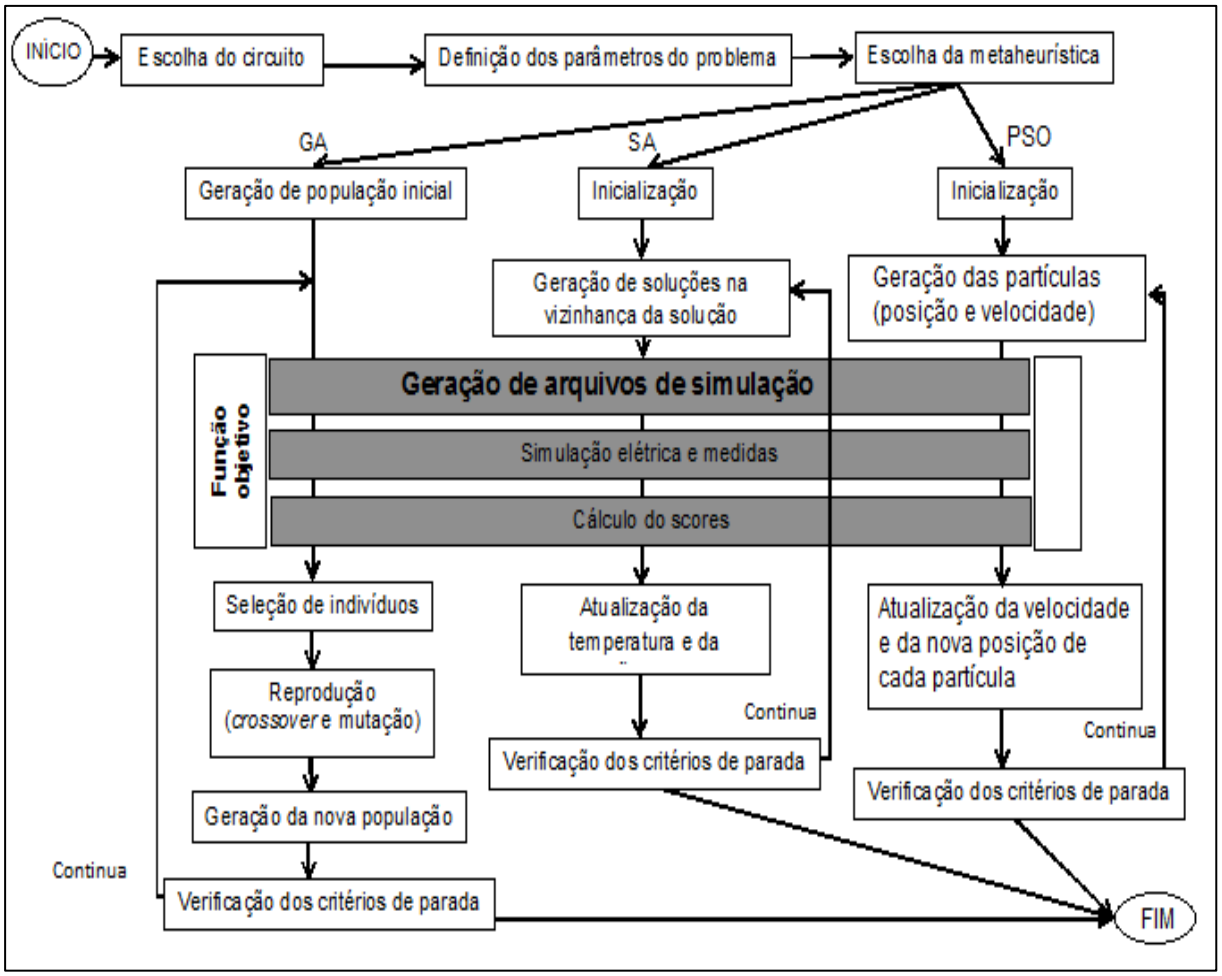

Figura 11. Esquemático completo de uma execução, para os algoritmos GA, SA e PSO.

O PSON utiliza o mesmo procedimento do PSO, mas conta com auxílio de redes neurais, que são treinadas para tentar prever quais partículas devem alcançar melhores notas sem necessidade de realizar simulações. Em uma primeira etapa, $\mathbf{N_{total}}$ partículas são geradas e todas avaliadas, cada uma recebendo a sua nota. A rede neural é então criada e treinada, com os melhores $\mathbf{N_{trei}}$ resultados já obtidos, para aprender a identificar quais partículas são as mais promissoras. As partículas têm seus vetores velocidade e sua posição atualizados, como padrão do PSO. A partir desse ponto, não são simuladas mais todas as partículas, mas apenas um grupo menor, $\mathbf{N_{teste}}$, que são aquelas mais promissoras para atingir a melhor nota. Também a rede neural é constantemente retreinada.

Com a aplicação de redes neurais, visamos aumentar a qualidade dos resultados, buscando circuitos melhores, simulando uma abrangência maior sem aumentar o número de partículas simuladas.

A rede neural aplicada tem as seguintes características:

- É do tipo *feed-forward backpropagation network*;
- O número de entradas é igual ao numero de variáveis otimizadas;
- Apresenta uma saída que da a nota do circuito;
- Tem uma camada escondida com (numero de variáveis)/2 neurônios;
- usa uma função radial para os neurônios.

O esquemático com os passos do PSON está demonstrado na **Figura 12**, a seguir.

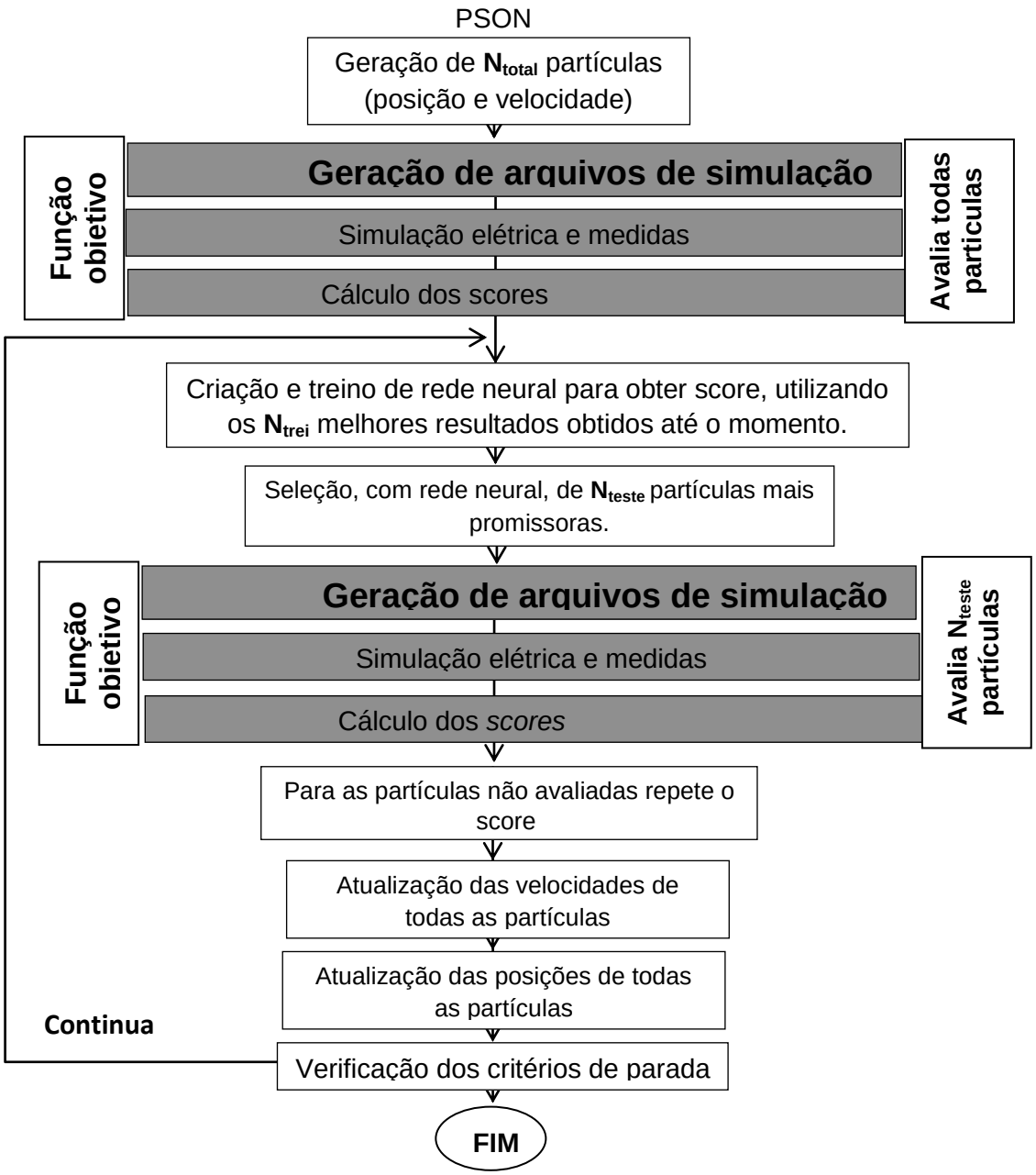

Figura 12. Esquemático de uma execução do algoritmo PSON.

Para o PSON, o valor de $\mathbf{N_{total}}$ foi de 4, 6 e 8 vezes o número de variáveis, o valor de $\mathbf{N_{trei}}$ foi de 100, 200, 300 e 400, formando um total de 12 configurações. O Valor de N_{teste} foi mantido igual ao número de variáveis.

2.6 Função de *Fitness*

Para avaliar os circuitos e dar a eles uma nota existe a função *Fitness*, ou função objetivo. Essa função é a mesma para ambos os algoritmos e realiza as seguintes operações: gera o arquivo, com comandos e valores das variáveis, chamado *param*, executa as simulações com um simulador elétrico Hspice, lê os resultados gerados e, por fim, calcula a nota do circuito. Caso tenha havido algum problema nas simulações, por exemplo, a não convergência, a nota infinita é atribuída. Ainda, a função *Fitness* mantém uma cópia do arquivo *param* do melhor circuito já encontrado, *paramop*, e que será a solução final. Para múltiplas execuções de otimização, o arquivo *paramopT* terá o resultado final. Todos os parâmetros obtidos e as notas atribuídas são exibidos na tela durante as otimizações, possibilitando o acompanhamento do processo. A descrição completa da função *Fitness* se encontra no Apêndice E.

Para a realização das simulações são especificadas as condições de funcionamento do circuito, como tensões mínima e máxima de alimentação e variação de temperatura. Para cálculo da nota são especificados os parâmetros de desempenho, como tensão de saída, coeficiente de temperatura, regulação de linha, consumo de potencia, área e estado de operação de alguns transistores, e os pesos que serão aplicados a cada um desses parâmetros. Através das simulações são determinados os valores da tensão de saída, V_{RefM}, do coeficiente de temperatura, CT_M, da regulação de linha, RL_M, da corrente consumida, I_{CONS}, dos índices de forte inversão para alguns transistores, I_{Stri}, e dos índices de fraca inversão para alguns transistores, I_{Weaki}. Com esses resultados são avaliados os termos V_{RefW}, TC_W, LR_W, P_W, que indicam quanto a tensão de saída, o coeficiente de temperatura, a regulação de linha, e a potência consumida estão longe dos valores desejados, e os valores de I_{StrW} e I_{WeakW}, que indicam quanto certos transistores estão fora da região de operação desejada. Estes termos são aplicados no cálculo final de uma nota.

Para obter o termo V_{RefW} é avaliada a seguinte expressão

$$V_{RefW} = \begin{cases} \dfrac{|V_{RefM} - V_{RefE}|}{\min(V_{RefM}, V_{RefE})} & se\ maior\ que\ Err \\ 0{,}0 & se\ menor\ que\ Err \end{cases} \tag{39}$$

onde,

V_{RefW} = contribuição da tensão de saída para o cálculo da nota;

V_{RefE} = V_{Ref} especificado;

Err = variação da tensão de saída permitida (erro relativo) e

Min{ , } = é a função mínimo

Para obter o termo TC_W é avaliada a seguinte expressão

$$TC_W = \begin{cases} \frac{CT_M - TC_E}{TC_E}, & se\ CT_M > TC_E \\ 0{,}0 & se\ CT_M \leq TC_E \end{cases} \quad (40)$$

onde,

TC_W = contribuição do TC para o cálculo da nota e

TC_E = TC especificado.

Para obter termo LR_W é avaliada a seguinte expressão

$$LR_W = \begin{cases} \frac{RL_M - LR_E}{LR_E}, & se\ RL_M > LR_E \\ 0,0 & se\ RL_M \leq LR_E \end{cases} \quad (41)$$

onde,

LR_W = contribuição do LR para o cálculo da nota e

LR_E = LR especificado.

Para obter P_W foi avaliada a seguinte expressão

$$P_W = \left(\frac{V_{Max} + V_{Min}}{2}\right)\frac{I_{CONS}}{P_R} \quad (42)$$

onde,

P_W = contribuição da potência média para o cálculo da nota;

P_R = potência de referência para o circuito (constante) e

V_{Max}, V_{Min} = máxima e mínima tensão de alimentação.

Para obter I_{StrW} foi avaliada a seguinte expressão

$$I_{StrW} = \sum_i 10.(0{,}1 - I_{Stri}) \quad \text{para} \quad I_{Stri} < 0{,}1 \quad (43)$$

onde,

I_{StrW} = contribuição do estado do transistor para o cálculo da nota e

i = índices dos transistores em que se deseja forte inversão.

O valor de I_{Stri} é calculado através da expressão:

$$I_{Stri} = (V_{gsi} - V_t) \tag{44}$$

Para obter I_{WeakW} foi avaliada a seguinte expressão

$$I_{WeakW} = \sum_i 10.(I_{Weaki}) \quad \text{para } I_{Weaki} > 0{,}12 \tag{45}$$

onde,

I_{WeakW} = contribuição do estado do transistor para o cálculo da nota e

i = índices dos transistores em que se deseja fraca inversão.

O valor de I_{Weaki} é calculado através da expressão:

$$I_{Weaki} = \frac{\frac{|I_{Di}|}{U_T(1+\frac{g_{mbi}}{g_{mi}})} - g_{mi}}{g_{mi}} \tag{46}$$

onde I_{Di} é a corrente de dreno, g_{mi} é a transcondutância e g_{mbi} é a transcondutância de efeito de corpo do iésimo transistor.

Um termo adicional é A_W, que indica quanto a área está longe de um valor de referência, é avaliado através da seguinte expressão

$$A_W = \frac{Área\ dos\ transistores + Área\ dos\ resistores}{A_R} \tag{47}$$

onde,

A_W = contribuição da área estimada para o cálculo da nota e

A_R = área de referência para o circuito (constante).

Por fim, o valor da nota é dada por

$$nota = \left(p1.V_{RefW} + p2.TC_W + p3.RL_W + p4.P_W + p5.I_{WeakW} + p6.I_{StrW} + p7.A_W\right)^2 \tag{48}$$

onde *p1*, *p2*, .. *p7* são pesos especificados pelo usuário.

3 Simulações e resultados

Para realizar as simulações, foi utilizado o software CirOp de projeto e otimização. Este software está sendo desenvolvido pelo professor Navarro, utilizando como plataforma o Matlab e alguns pacotes deste.

No programa, primeiramente, escolhe-se o bloco que se deseja projeta e otimizar (**Figura 13**). Em seguida, escolhe-se qual topologia do bloco será simulada, ajustando-se os valores dos parâmetros desejados para o desempenho circuito e os intervalos possíveis para as variáveis (**Figura 14**). Por último, escolhe-se a quantidade de vezes que se deseja otimizar a mesma configuração, o número máximo de simulações por execução, o nome do arquivo contendo os resultados e se as execuções vão ter um circuito inicial aleatório ou não (**Figura 15**). O arquivo contendo os resultados será armazenado na pasta “results”, dentro da pasta da topologia escolhida. A seguir, imagens dos menus de escolha para o usuário:

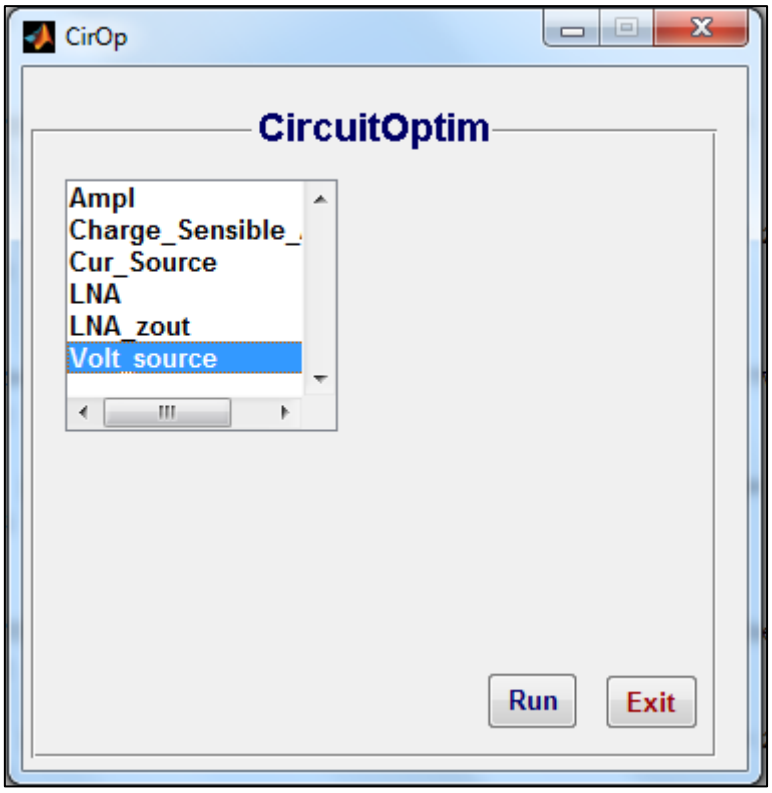

Figura 13. Menu para escolha do tipo de circuito a ser simulado.

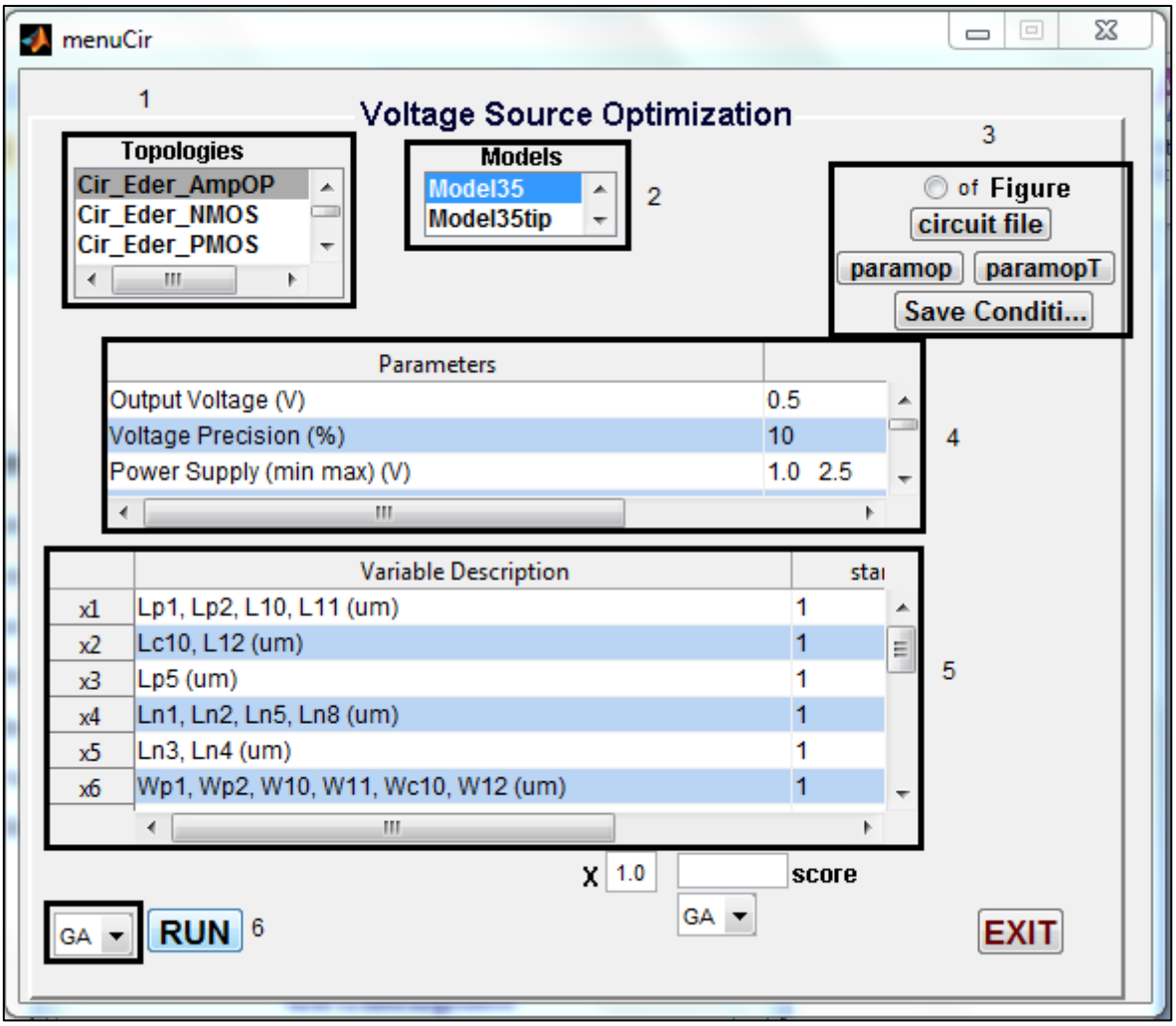

Figura 14. Menu de escolha do tipo de topologia (1), escolha dos parâmetros desejados (4), intervalo de valores das variáveis (5) e algoritmo desejado (6).

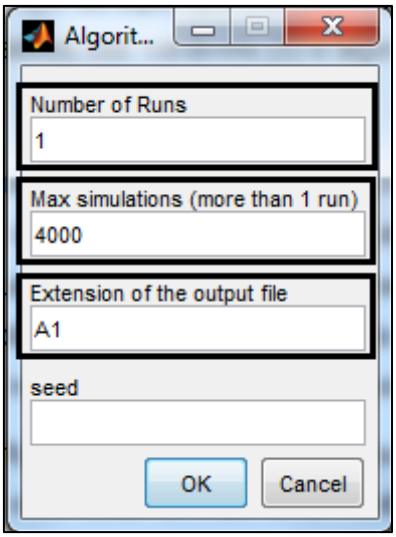

Figura 15. Menu de escolha de quantas execuções serão realizadas com a mesma configuração (1), número máximo de simulações por execução (4000) e nome do arquivo de saída (A1).

Os parâmetros de funcionamento de cada circuito foram mantidos os mesmos para todos os algoritmos executados. A seguir, a **Tabela 5**, **Tabela 6** e **Tabela 7** apresentam as condições de funcionamento do circuito, especificados os parâmetros de desempenho e os pesos utilizados na função de *Fitness*.

Tabela 5. Configuração dos parâmetros de funcionamento, parâmetros de desempenho e pesos utilizados para o Circuito Ishibe.

Parâmetros	**Valores**
Tensão de saída(V)	0,5
Variação da tensão de saída (%)	10
Tensão de alimentação (min. máx.) (V)	1,5 2,5
Temperatura (min. máx.) (Celsius)	-10 90
Coeficiente de temperatura (ppm/C)	15
Regulação de linha (ppm)	500
Consumo (uW)	5
Área (um x um)	
Transistor em fraca inversão	Mn1
Transistor em forte inversão	
Constantes	3,0
Pesos (Vref, TC, RL, Pot, Winv, Stinv, Area)	1,0 10 10 10 1,0 0,1

Na **Tabela 5**, observa-se os parâmetros para o circuito Ishibe, onde os pesos da última linha estão relacionados com a tensão de referência, Vref, o coeficiente de temperatura, TC, a regulação de linha, RL, a potência consumida pelo circuito, Pot, os

transistores em fraca e forte inversão, respectivamente Winv e Sinv, e à área do circuito, Área. A constante com valor 3,0 representa quantos transistores iguais ao transistor M_{p1} são usados em paralelo, ou seja, quantas vezes o transistor resultante é maior que o M_{p2}. Isso é determinado na parametrização do circuito, encontrada no Apêndice A.

Tabela 6. Configuração dos parâmetros de funcionamento, parâmetros de desempenho e pesos utilizados para o Circuito Ueno completo.

Parâmetros	**Valores**
Tensão de saída(V)	0,6
Variação da tensão de saída(%)	40
Tensão de alimentação (min. máx.)(V)	1,0 2,5
Temperatura (min. máx.) (Celsius)	-10 90
Coeficiente de temperatura (ppm/C)	20
Regulação de linha (ppm)	80
Consumo (uW)	2,0
Área (um x um)	
Transistor em fraca inversão	Mn2
Transistor em forte inversão	
Constantes	3,0 1,0 10 0,1
Pesos (Vref, TC, RL, Pot, Winv, Stinv, Area)	1,0 10 10 0,1 1,0 1,0 0,1

Na **Tabela 6**, observa-se os parâmetros para o circuito Ueno completo, onde a constante com valor 3,0 tem a mesma função que no circuito Ishibe, e as constantes com valores 1,0, 10 e 0,1 representam, respectivamente, quantos transistores iguais à M_{p3}, M_{p4} e M_{p5} são usados em paralelo, M2, o tamanho da largura dos transistores M_{n3}, M_{n4}, M_{n5} e M_{n6}, M3, e quantos transistores iguais ao M_{n12} são usados em paralelo. Isso é determinado na parametrização do circuito, encontrada no Apêndice BApêndice A.

Tabela 7. Configuração dos parâmetros de funcionamento, parâmetros de desempenho e pesos utilizados para o Circuito Ueno modificado.

Parâmetros	**Valores**
Tensão de saída(V)	0,7
Variação da tensão de saída(%)	20
Tensão de alimentação (min. máx.)(V)	1,0 2,5
Temperatura (min. máx.) (Celsius)	-10 90
Coeficiente de temperatura (ppm/C)	20
Regulação de linha (ppm)	100
Consumo (uW)	2,0
Área (um x um)	
Transistor em fraca inversão	Mn1
Transistor em forte inversão	
Constantes	3,0 1,0 10 0,1

Pesos (Vref, TC, RL, Pot, Winv, Stinv, Area)	10 100 10 1,0 1,0 1,0 0,1

Na **Tabela 7**, as constantes referentes aos pesos 3,0, 1,0 e 10 tem a mesma função que no Ueno completo, e a última constante, 0,1, mostra quantos transistores M_{n10} e M_{n11} são usados em paralelo. Isso é determinado na parametrização do circuito, encontrada no Apêndice CApêndice A.

Como foram utilizados algoritmos metaheurísticos, baseados em escolhas randômicas, existe a chance de o algoritmos convergir para soluções que não são as melhores possíveis. Para tentar aumentar a probabilidade de obter uma boa solução é normal serem realizadas múltiplas otimizações. Em nosso trabalho foram realizadas, para cada circuito, algoritmo e configuração do algoritmo, 15 execuções.

Como queremos visualizar o desempenho de cada configuração dos algoritmos, precisamos condensar os resultados das 15 execuções o melhor possível. Trabalhamos sempre com a média geométrica das 15 otimizações. Essa média versus a interação onde ela é obtida é exibida em um gráfico. Os gráficos foram separados por circuito e por configuração, de modo que cada gráfico mostre todas as configurações de um único.

Para todos os algoritmos, foram limitados os possíveis valores que as variáveis poderiam assumir, de acordo com a **Tabela 8**, **Tabela 9** e **Tabela 10**.

Tabela 8. Valores das parametrizações das variáveis do Circuito Ishibe.

Variáveis	**Mínimo**	**Máximo**
X1 (um)	1,0	20
X2 (um)	1,0	20
X3 (um)	1,0	30
X4 (um)	1,0	20
X5 (um)	1,0	30
X6 (um)	1,0	20
X7 (um)	1,0	400
X8 (um)	1,0	100
X9 (um)	1,0	30
X10 (um)	1,0	100
X11 (um)	1,0	100
X12 (um)	1,0	100
X13 (kOhm)	50	200
X14	0,6	3,0
X15	0,6	1,6
X16	1,0	6,0

X17	1,0	6,0

Tabela 9. Valores das parametrizações das variáveis do Circuito Ueno completo.

Variáveis	Mínimo	Máximo
X1 (um)	1,0	70
X2 (um)	10	100
X3 (um)	1,0	20
X4 (um)	1,0	100
X5 (um)	1,0	20
X6 (um)	1,0	200
X7 (um)	1,0	5,0
X8	1,0	50
X9	1,0	20
X10 (um)	1,0	100
X11 (um)	1,0	100
X12 (um)	0,5	20
X13 (um)	1,0	100
X14 (um)	1,0	50
X15 (um)	2,0	4,0

Tabela 10. Valores das parametrizações das variáveis do Circuito Ueno modificado.

Variáveis	Mínimo	Máximo
X1 (um)	1,0	40
X2 (um)	10	100
X3 (um)	1,0	20
X4 (um)	1,0	200
X5 (um)	1,0	20
X6 (um)	1,0	200
X7 (um)	1,0	5.0
X8	1,0	50
X9	1,0	20
X10 (um)	1,0	100
X11 (um)	1,0	60
X12 (um)	2,0	4.0

Após todas as simulações, verificando os arquivos de resultados globais, paramopT, pode-se ver qual foi o melhor resultado obtido para as dimensões dos componentes, bem como quais os parâmetros de funcionamento que o circuito obteve.

Circuito Ishibe:

```
.param Vref = 0.50
```

```
.ParamX1= 15.083  X2= 4.325  X3= 8.762  X4= 5.156  X5= 4.389  X6= 7.025
.ParamX7= 64.769  X8= 35.373  X9= 2.567  X10= 91.907  X11= 21.292  X12=
1.168  X13= 179.542
.Param   X14= 1.060  X15= 1.063  X16= 4.176  X17= 1.352
*Score=0.0035  TC= 7.6ppm  (FTC= 0)  RL = 471ppm  (FRL = 0) Vref =0.53V
(FVref =0.00) Pot.=2.9uW (Fpot= 0.59)
```

Circuito Ueno complete:

```
.param Vref = 0.60
.ParamX1= 6.126  X2= 21.726  X3= 17.502  X4= 10.522  X5= 1.344  X6= 183.307
.ParamX7= 3.049  X8= 13.998  X9= 1.616  X10= 1.000  X11= 1.000  X12= 12.687
X13= 48.114
.ParamX14= 35.171  X15= 2.013
*Score=1.7e+002TC= 40ppm  (FTC= 0.98)  RL = 106ppm  (FRL = 0.33) Vref
=0.77V (FVref =0.00) Pot.=0.96uW (Fpot= 0.48)
```

Circuito Ueno modificado:

```
.param Vref = 0.70
.ParamX1= 17.613  X2= 46.544  X3= 8.759  X4= 37.711  X5= 9.570  X6= 43.432
.ParamX7= 4.578  X8= 27.764  X9= 3.607  X10= 3.337  X11= 21.073  X12= 3.528
*Score=1e+002TC= 19ppm  (FTC= 0)  RL = 158ppm  (FRL = 0.58) Vref =0.78V
(FVref =0.00) Pot.=3.5uW (Fpot= 1.7)
```

3.1 Algoritmo Genético

3.1.1 Circuito Ishibe

Os resultados da aplicação de Algoritmos Genéticos no circuito Ishibe são apresentados na **Figura 16**, **Figura 17** e na **Figura 18**.

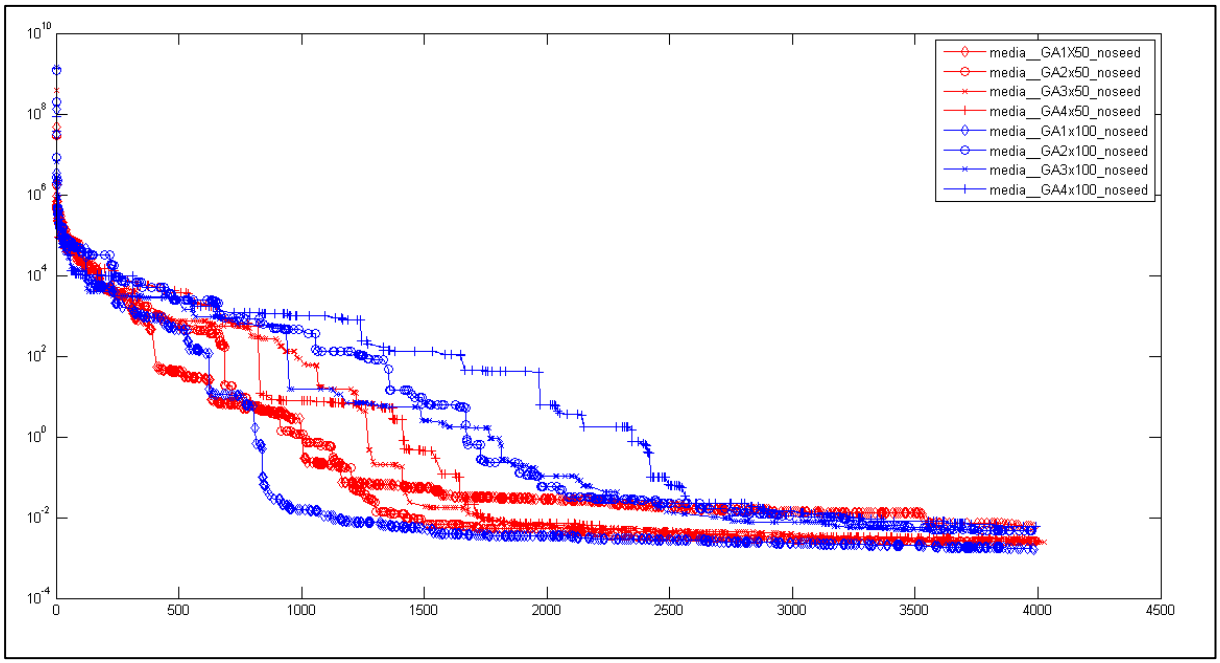

Figura 16. Gráfico da nota média versus simulações para o Algoritmo Genético aplicado ao circuito Ishibe. As configurações testadas são: 1, 2, 3 e 4 populações com 50 indivíduos em cada população, **GA1x50**, **GA2x50**, **GA3x50** e **GA4x50**; 1, 2, 3 e 4 populações com 100 indivíduos em cada população, **GA1x100**, **GA2x100**, **GA3x100** e **GA4x100**.

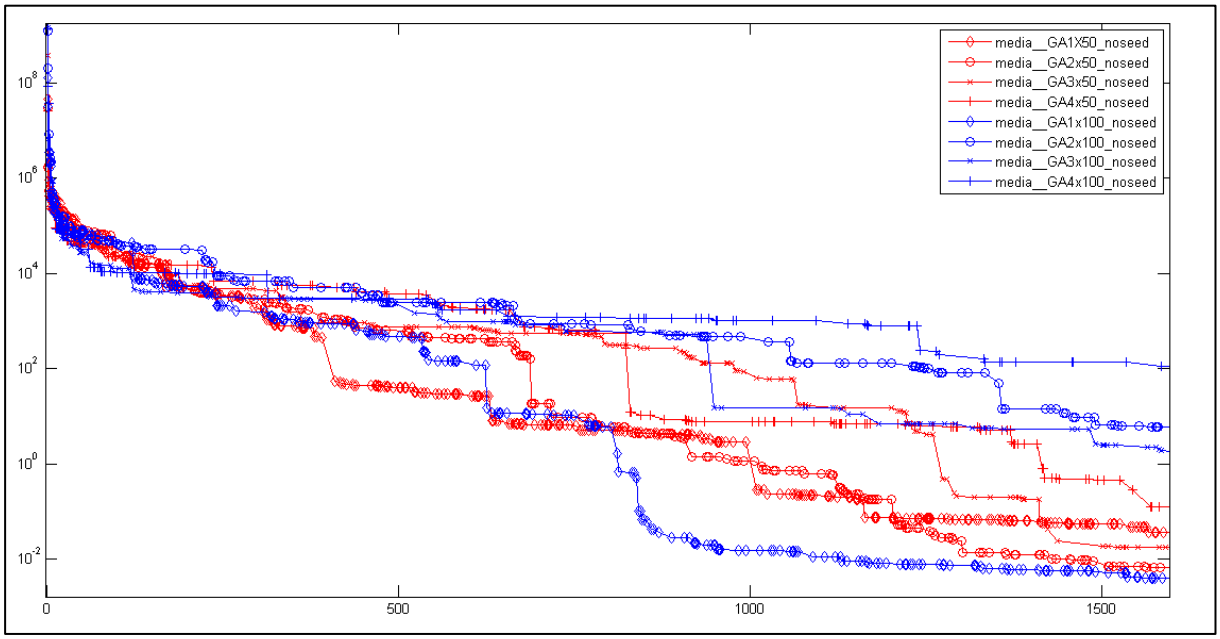

Figura 17. Gráfico da nota versus simulações para o Algoritmo Genético, aplicado ao circuito Ishibe. São mostrados em detalhes os resultados obtidos ao início das simulações.

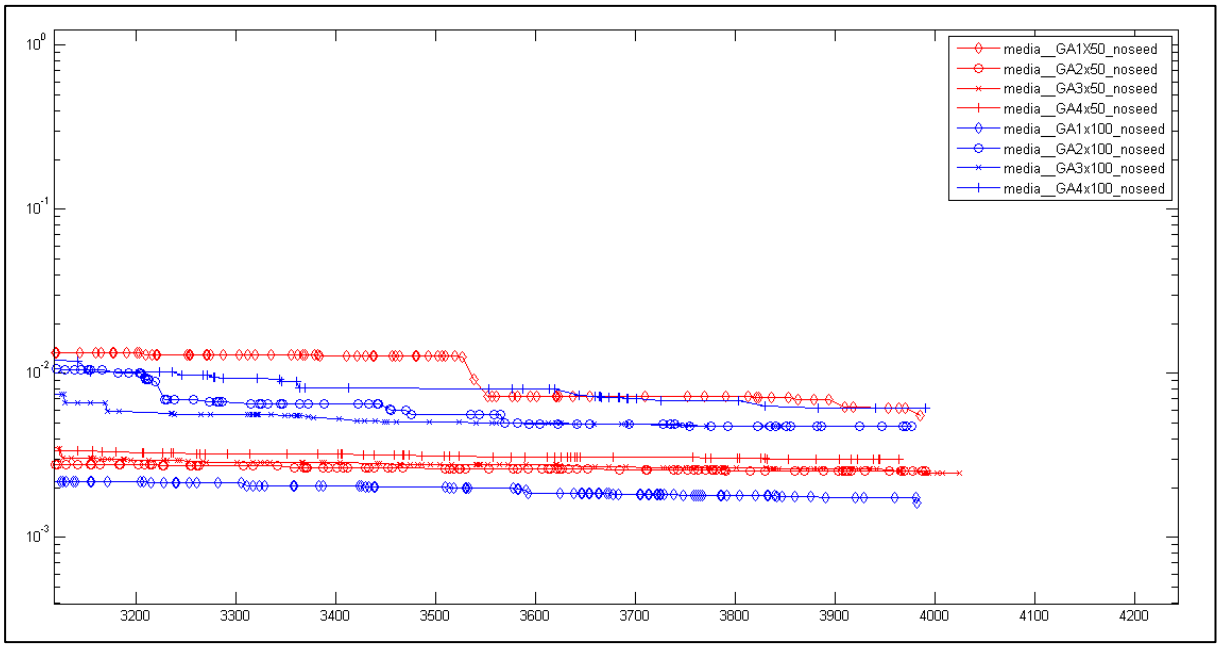

Figura 18. Gráfico da nota versus simulações para o Algoritmo Genético, aplicado ao circuito Ishibe. São mostrados em detalhes os resultados obtidos ao fim das simulações.

Analisando os gráficos, podemos perceber que o melhor resultado foi obtido utilizando a configuração com uma população com 100 indivíduos, que também foi a configuração que convergiu mais rápido para um resultado aceitável.

3.1.2 Circuito Ueno completo

Os resultados da aplicação de Algoritmos Genéticos no circuito Ueno completo são apresentados na **Figura 19**, **Figura 20** e **Figura 21**.

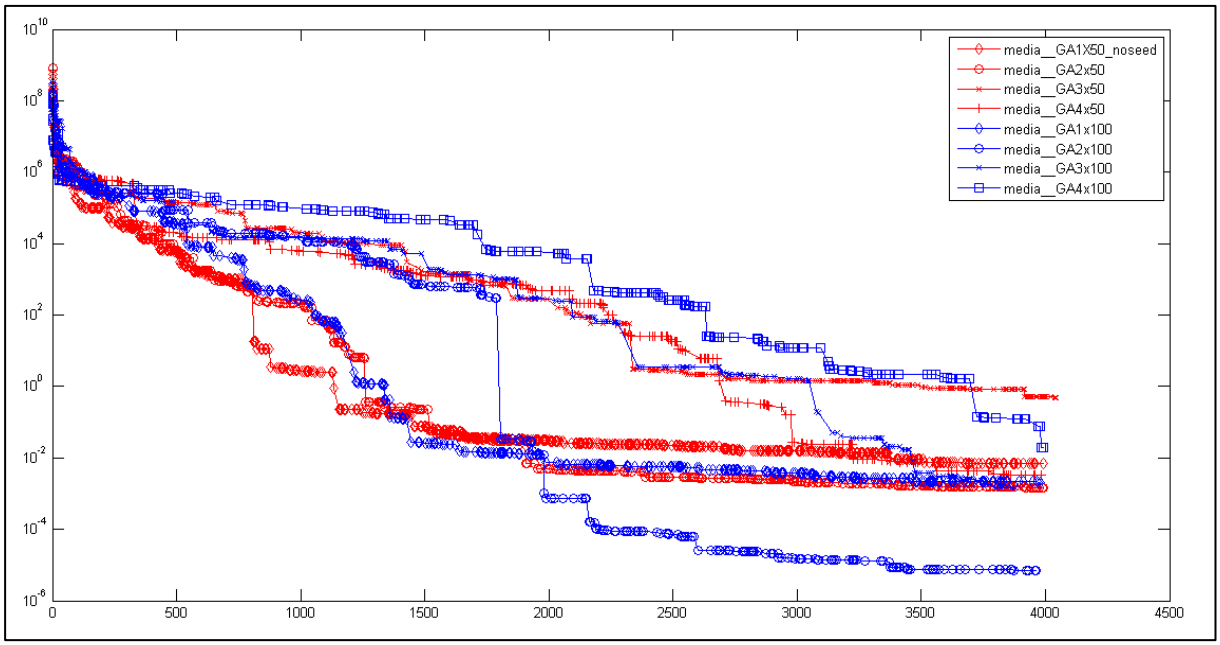

Figura 19. Gráfico da nota média versus simulações para o Algoritmo Genético aplicado ao circuito Ueno completo. As configurações testadas são: 1, 2, 3 e 4 populações com 50 indivíduos em cada população, **GA1x50_noseed**, **GA2x50**, **GA3x50** e **GA4x50**; 1, 2, 3 e 4 populações com 100 indivíduos em cada população, **GA1x100**, **GA2x100**, **GA3x100** e **GA4x100**.

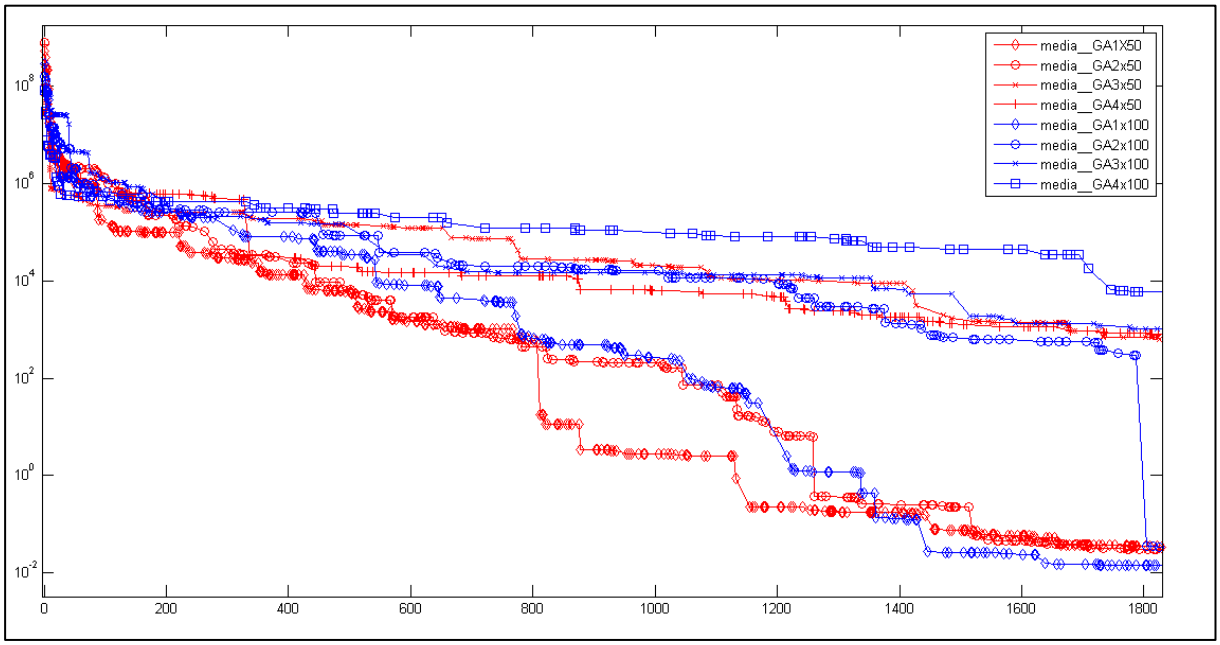

Figura 20. Gráfico da nota versus simulações para o Algoritmo Genético, aplicado ao circuito Ueno completo. São mostrados em detalhes os resultados obtidos ao início das simulações.

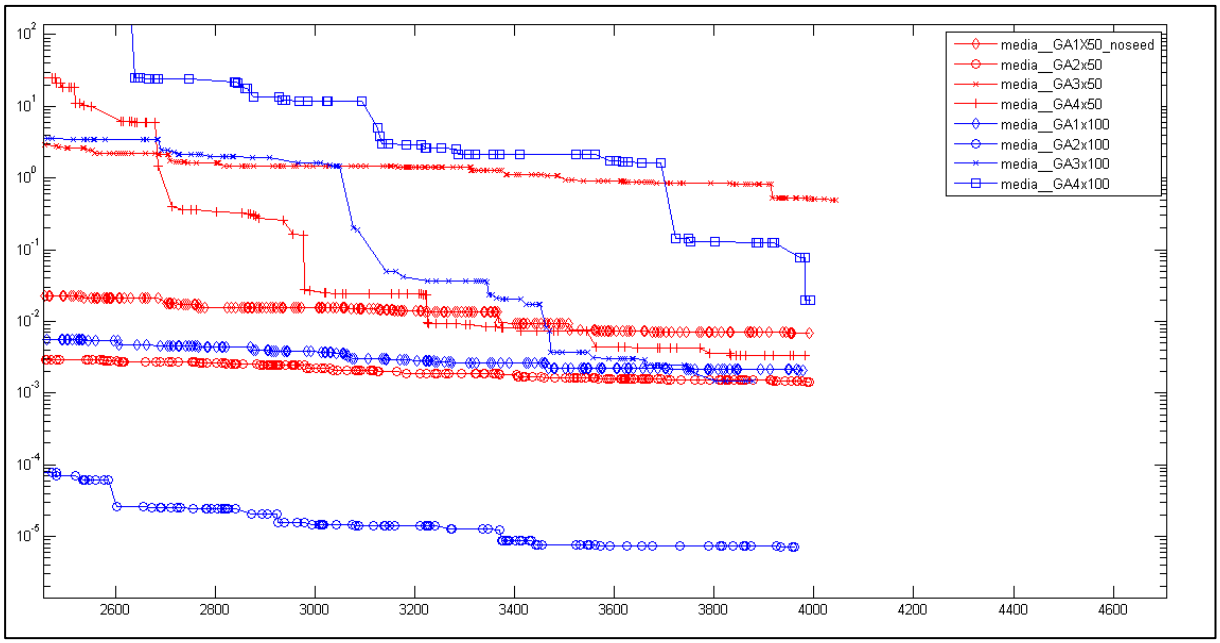

Figura 21. Gráfico da nota versus simulações para o Algoritmo Genético, aplicado ao circuito Ueno completo. São mostrados em detalhes os resultados obtidos ao fim das simulações.

Analisando os gráficos, podemos perceber que o melhor resultado foi obtido utilizando a configuração com duas populações com 100 indivíduos e a que convergiu mais rápido para um resultado aceitável foi com uma população de 100 indivíduos.

3.1.3 Circuito Ueno modificado

Os resultados da aplicação de Algoritmos Genéticos no circuito Ueno modificado são apresentados na **Figura 22**, **Figura 23** e **Figura 24**.

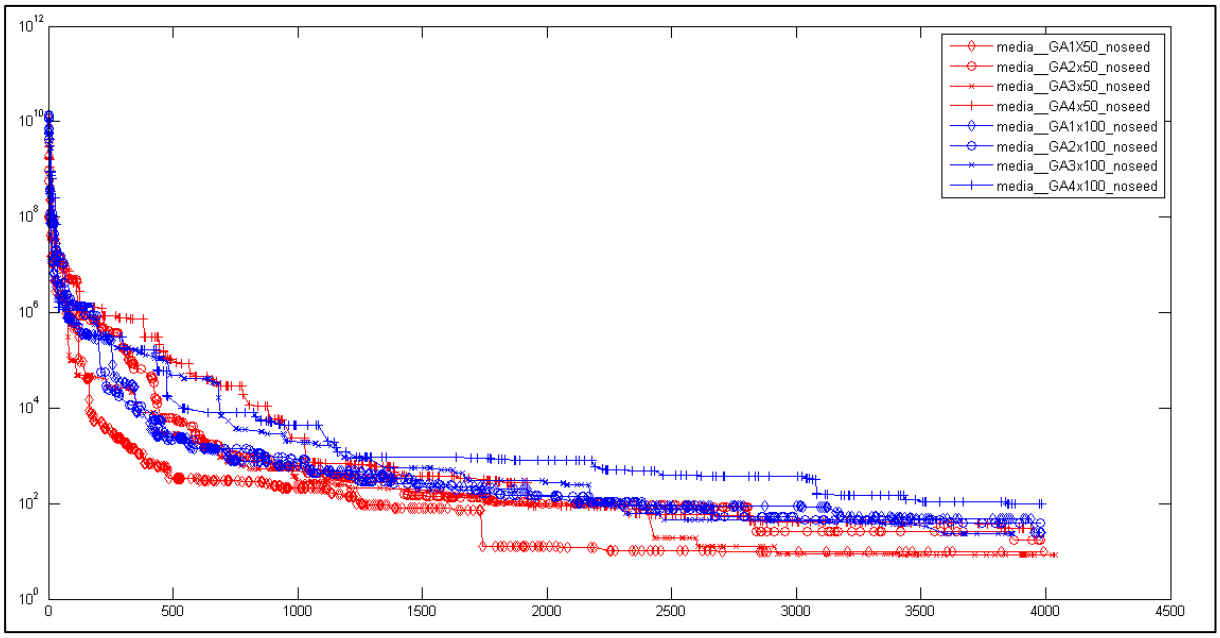

Figura 22. Gráfico da nota média versus simulações para o Algoritmo Genético aplicado ao circuito Ueno modificado. As configurações testadas são: 1, 2, 3 e 4 populações com 50 indivíduos em cada população, **GA1x50_noseed**, **GA2x50_noseed**, **GA3x50_noseed** e **GA4x50_noseed**; 1, 2, 3 e 4 populações com 100 indivíduos em cada população, **GA1x100_noseed**, **GA2x100_noseed**, **GA3x100_noseed** e **GA4x100_noseed**.

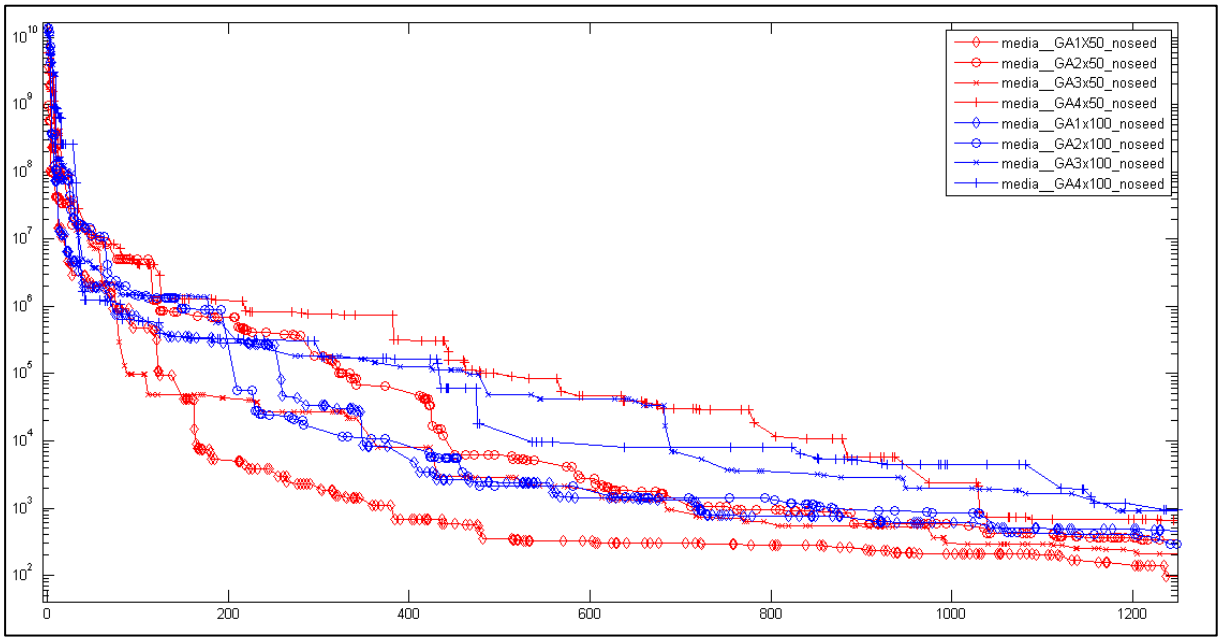

Figura 23. Gráfico da nota versus simulações para o Algoritmo Genético, aplicado ao circuito Ueno modificado. São mostrados em detalhes os resultados obtidos ao início das simulações.

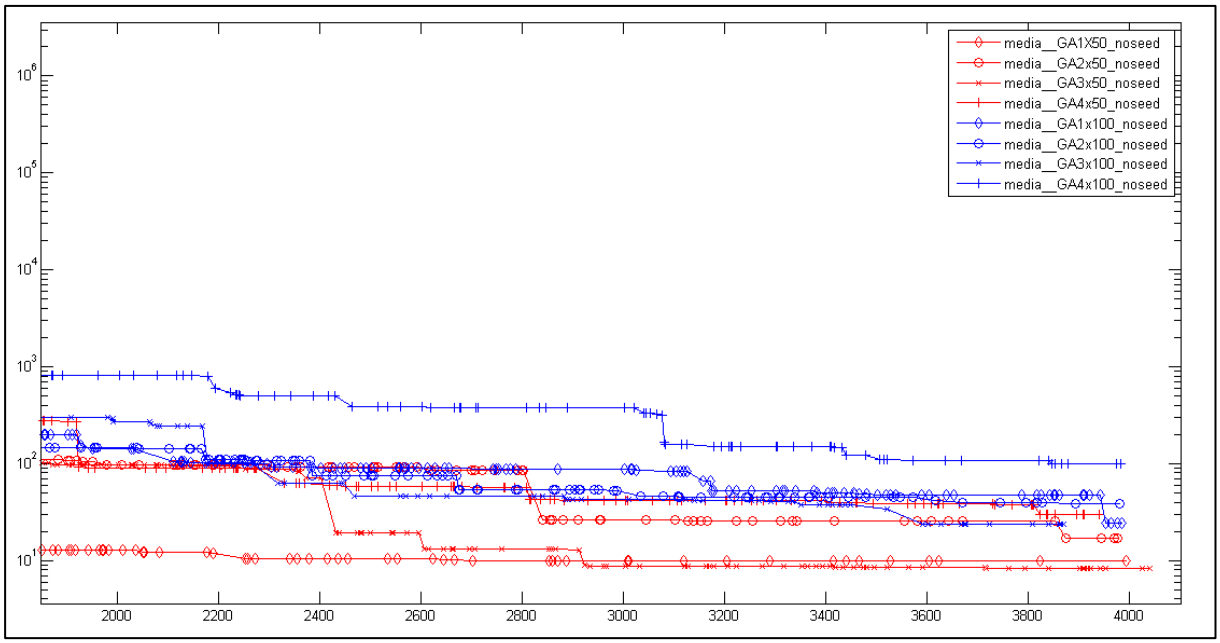

Figura 24. Gráfico da nota versus simulações para o Algoritmo Genético, aplicado ao circuito Ueno modificado. São mostrados em detalhes os resultados obtidos ao fim das simulações.

Analisando os gráficos, podemos perceber que o melhor resultado foi obtido utilizando a configuração com três populações com 50 indivíduos e a que convergiu mais rápido para um resultado aceitável foi com uma população de 50 indivíduos.

3.2 Simulated Annealing

3.2.1 Circuito Ishibe

Os resultados da aplicação de *Simulated Annealing* no circuito Ishibe são apresentados na **Figura 25**, **Figura 26** e **Figura 27**.

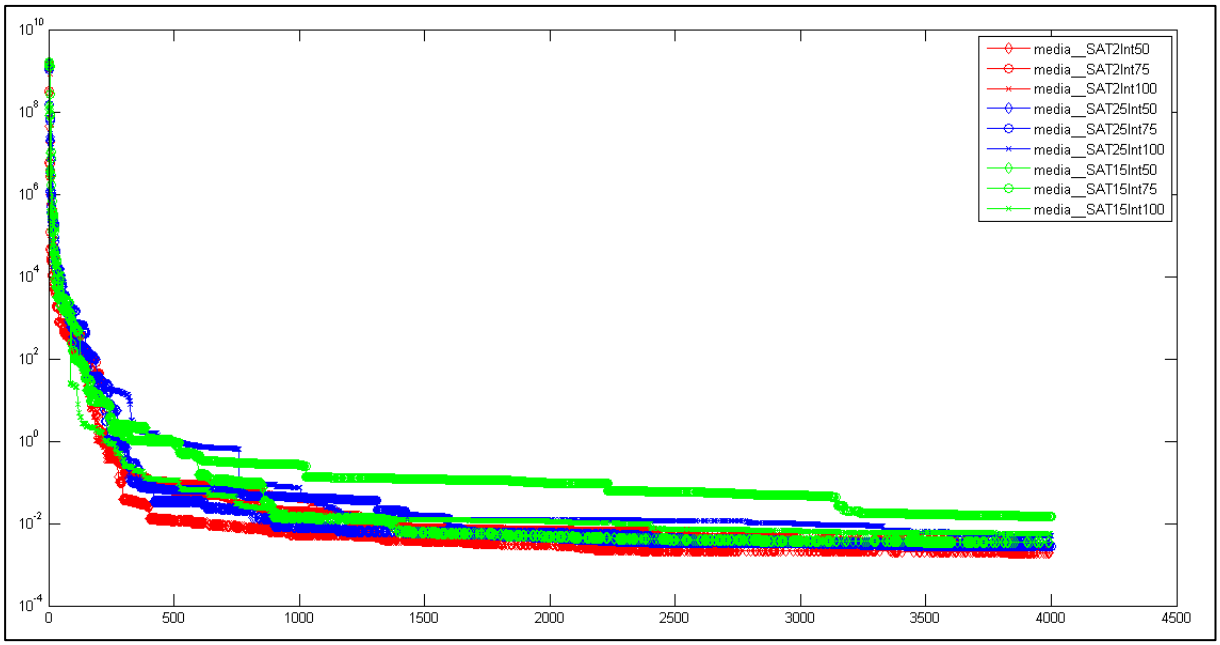

Figura 25. Gráfico da nota média versus simulações para o *Simulated Annealing* aplicado ao circuito Ishibe. As configurações testadas são: temperatura inicial 2 com 50, 75 e 100 trocas permitidas, **SAT2Int50**,**SAT2Int75** e **SAT2Int100**; temperatura inicial 2,5 com 50, 75 e 100 trocas permitidas, **SAT25Int50**, **SAT25Int75** e **SAT25Int100**; temperatura inicial 1,5 com 50, 75 e 100 trocas permitidas, **SAT15Int50**, **SAT15Int75** e **SAT15Int100**.

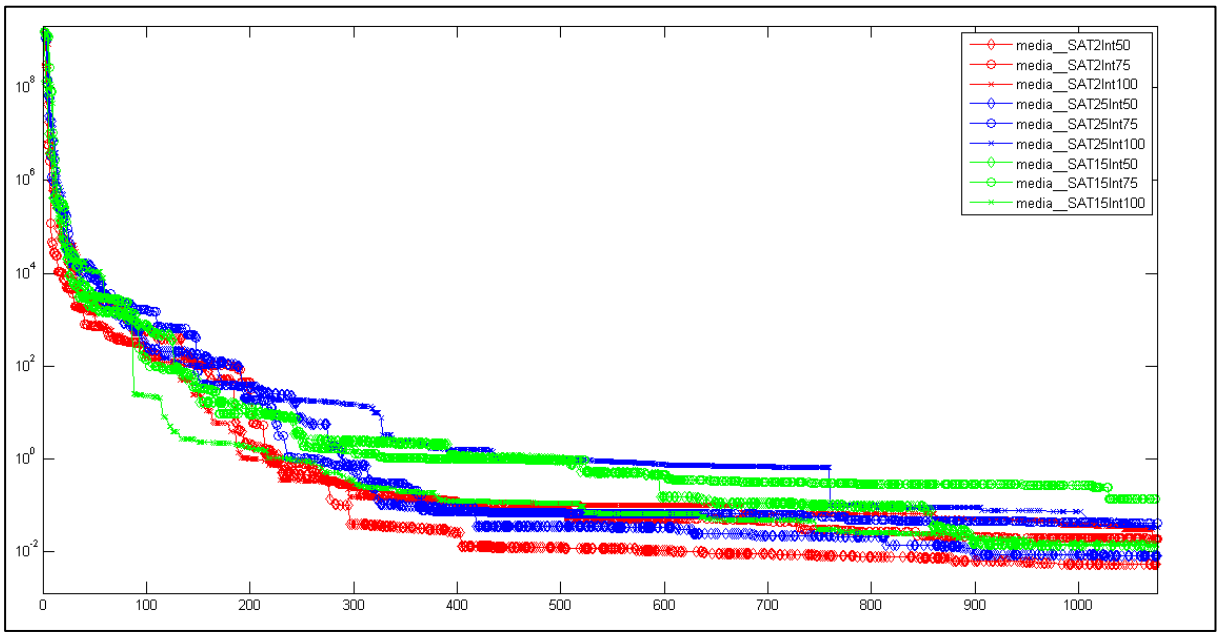

Figura 26. Gráfico da nota versus simulações para o *Simulated Annealing* aplicado ao circuito Ishibe. São mostrados em detalhes os resultados obtidos ao início das simulações.

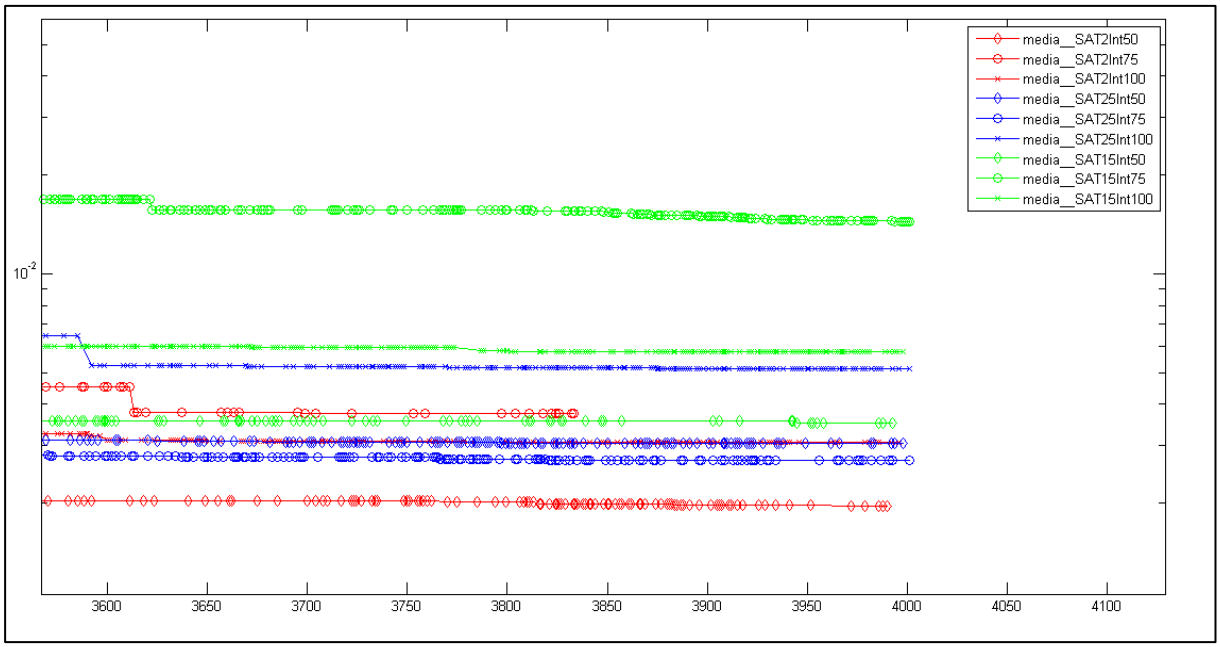

Figura 27. Gráfico da nota versus simulações para o *Simulated Annealing* aplicado ao circuito Ishibe. São mostrados em detalhes os resultados obtidos ao fim das simulações.

Analisando os gráficos, podemos perceber que o melhor resultado foi obtido utilizando a configuração com temperatura inicial 2, com o máximo de 50 trocas, e também foi a configuração que convergiu mais rápido para um resultado aceitável.

3.2.2 Circuito Ueno complete

Os resultados da aplicação de *Simulated Annealing* no circuito Ueno completo são apresentados na **Figura 28**, **Figura 29** e **Figura 30**.

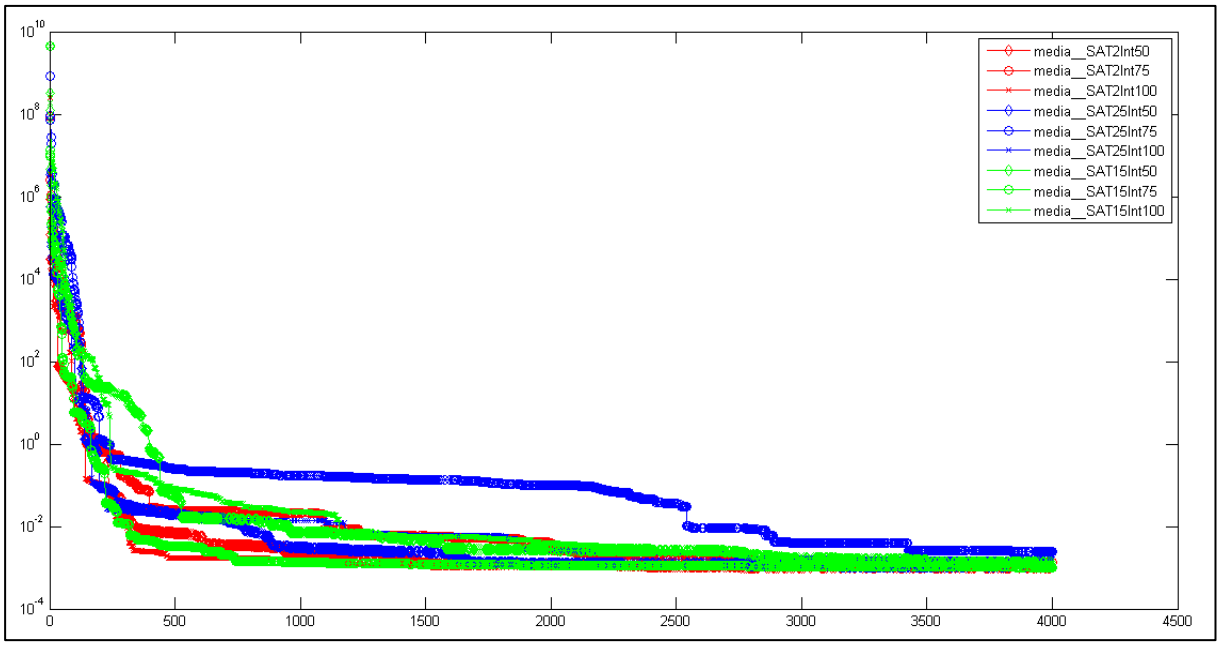

Figura 28. Gráfico da nota média versus simulações para o *Simulated Annealing* aplicado ao circuito Ueno completo. As configurações testadas são: temperatura inicial 2 com 50, 75 e 100 trocas permitidas, **SAT2Int50**, **SAT2Int75** e **SAT2Int100**; temperatura inicial 2,5 com 50, 75 e 100 trocas permitidas, **SAT25Int50**, **SAT25Int75** e **SAT25Int100**; temperatura inicial 1,5 com 50, 75 e 100 trocas permitidas, **SAT15Int50**, **SAT15Int75** e **SAT15Int100**.

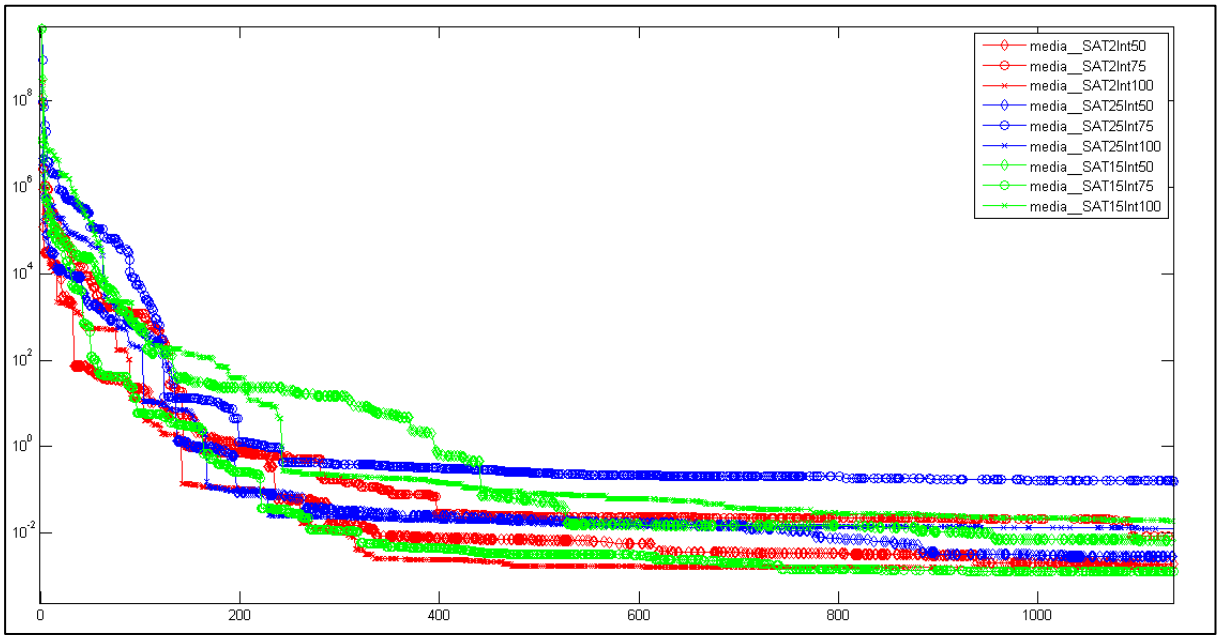

Figura 29. Gráfico da nota versus simulações para o *Simulated Annealing* aplicado ao circuito Ueno completo. São mostrados em detalhes os resultados obtidos ao início das simulações.

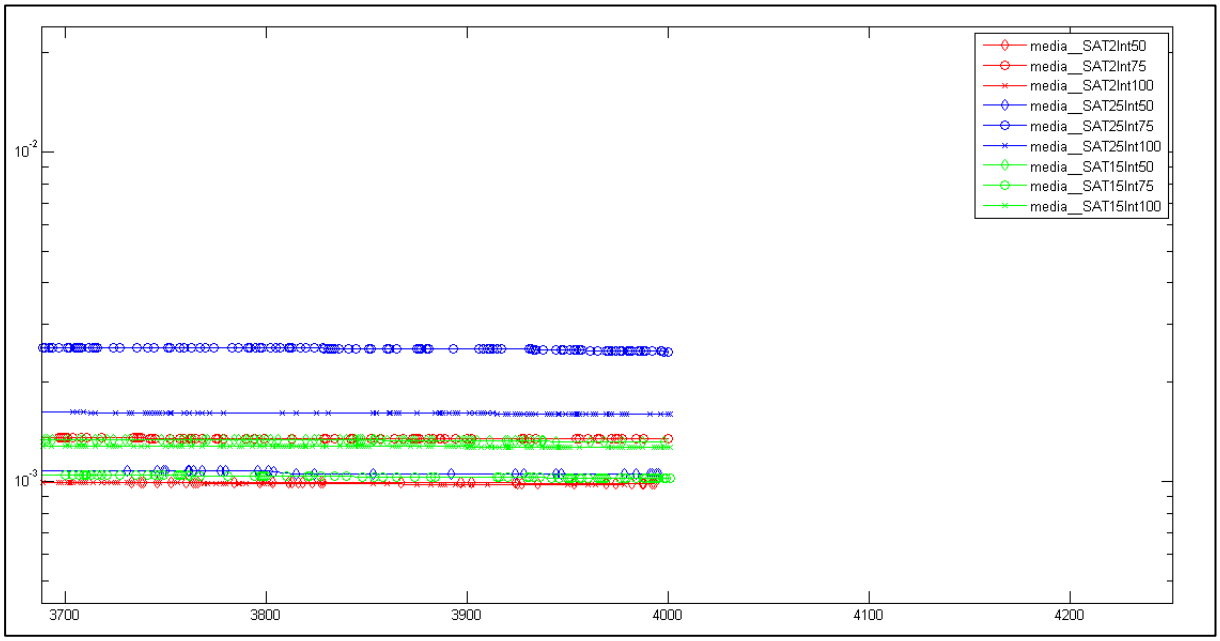

Figura 30. Gráfico da nota versus simulações para o *Simulated Annealing*, aplicado ao circuito Ueno completo. São mostrados em detalhes os resultados obtidos ao fim das simulações.

Analisando os gráficos, podemos perceber que o melhor resultado foi obtido utilizando a configuração com temperatura inicial 2,0 e máximo de trocas igual a 50, embora a configuração com temperatura inicial 2,0 máximo de trocas 100 tenha ficado bem próxima. A configuração que convergiu mais rápido para um resultado aceitável foi com temperatura inicial 2,0 e máximo de trocas igual a 100.

3.2.3 Circuito Ueno modificado

Os resultados da aplicação de *Simulated Annealing* no circuito Ueno modificado são apresentados na **Figura 31**, **Figura 32** e **Figura 33**.

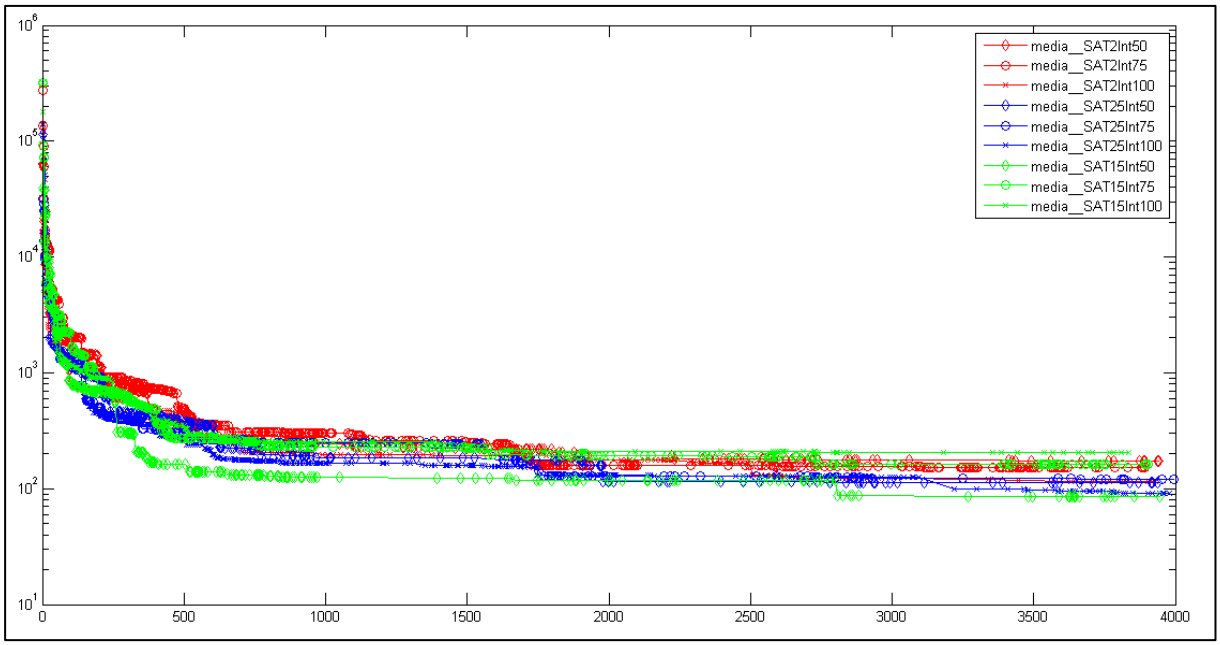

Figura 31. Gráfico da nota média versus simulações para o *Simulated Annealing* aplicado ao circuito Ueno modificado. As configurações testadas são: temperatura inicial 2 com 50, 75 e 100 trocas permitidas, **SAT2Int50**, **SAT2Int75** e **SAT2Int100**; temperatura inicial 2,5 com 50, 75 e 100 trocas permitidas, **SAT25Int50**, **SAT25Int75** e **SAT25Int100**; temperatura inicial 1,5 com 50, 75 e 100 trocas permitidas, **SAT15Int50**, **SAT15Int75** e **SAT15Int100**.

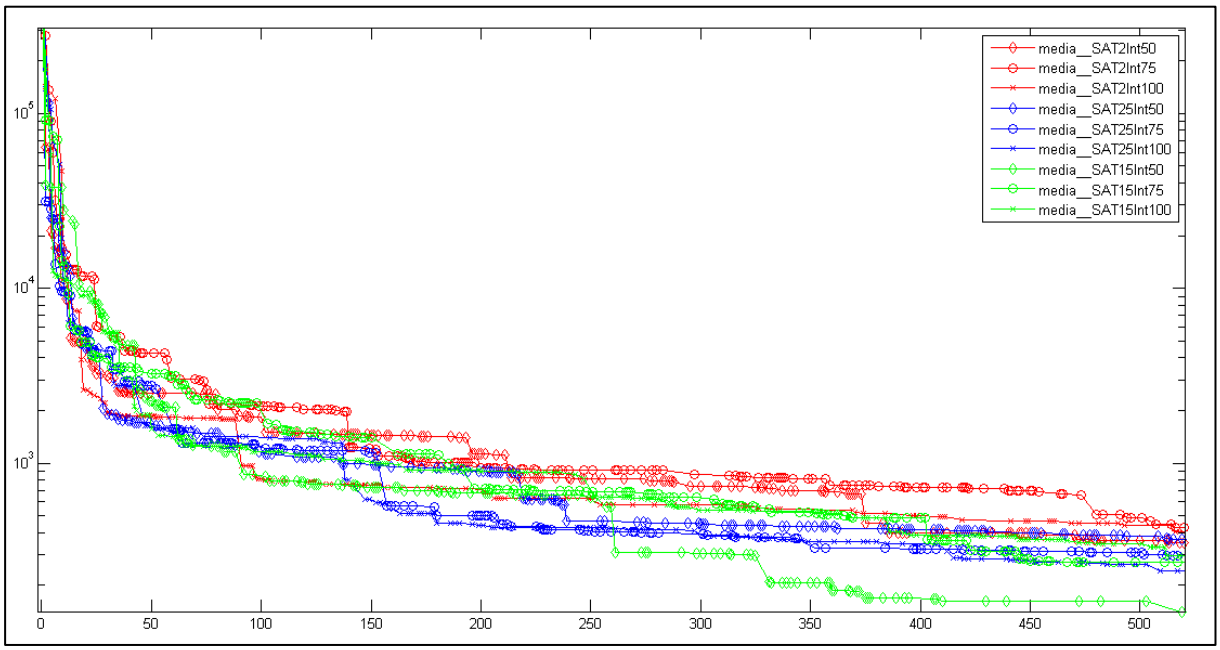

Figura 32. Gráfico da nota versus simulações para o *Simulated Annealing* aplicado ao circuito Ueno modificado. São mostrados em detalhes os resultados obtidos ao início das simulações.

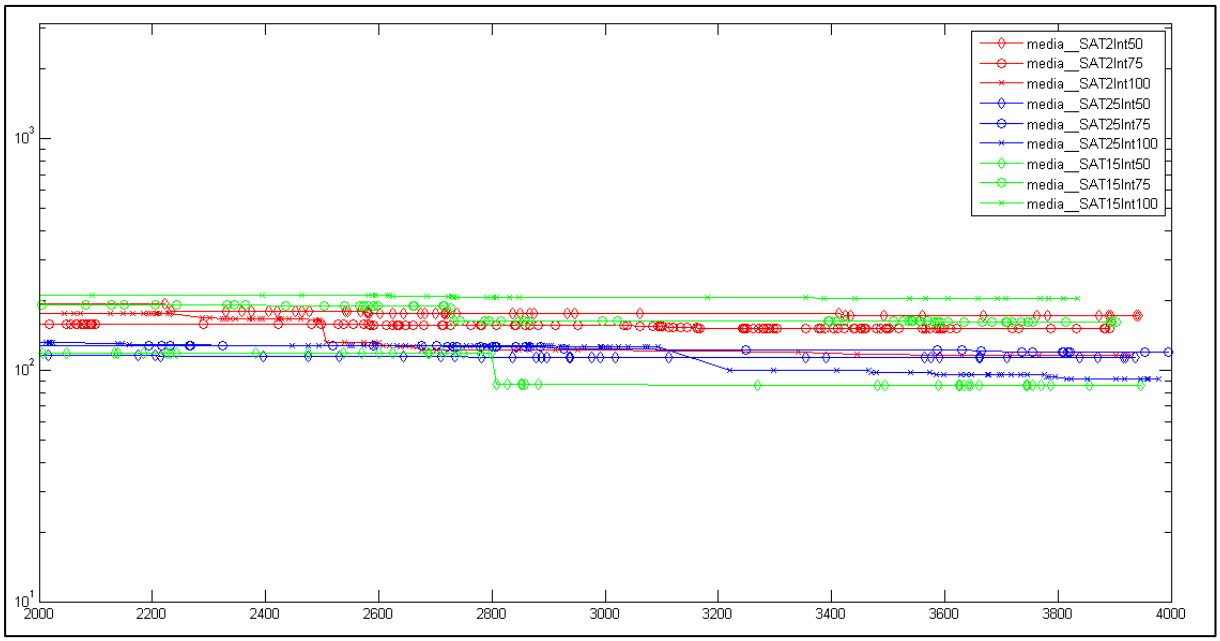

Figura 33. Gráfico da nota versus simulações para o *Simulated Annealing*, aplicado ao circuito Ueno modificado. São mostrados em detalhes os resultados obtidos ao fim das simulações.

Analisando os gráficos, podemos perceber que o melhor resultado foi obtido utilizando a configuração com temperatura inicial 1,5 e máximo de trocas igual a 50 e também foi a que convergiu mais rápido para um resultado aceitável.

3.3 Particle Swarm

3.3.1 Circuito Ishibe

Os resultados da aplicação de *Particle Swarm* no circuito Ishibe são apresentados na **Figura 34**, **Figura 35** e **Figura 36**.

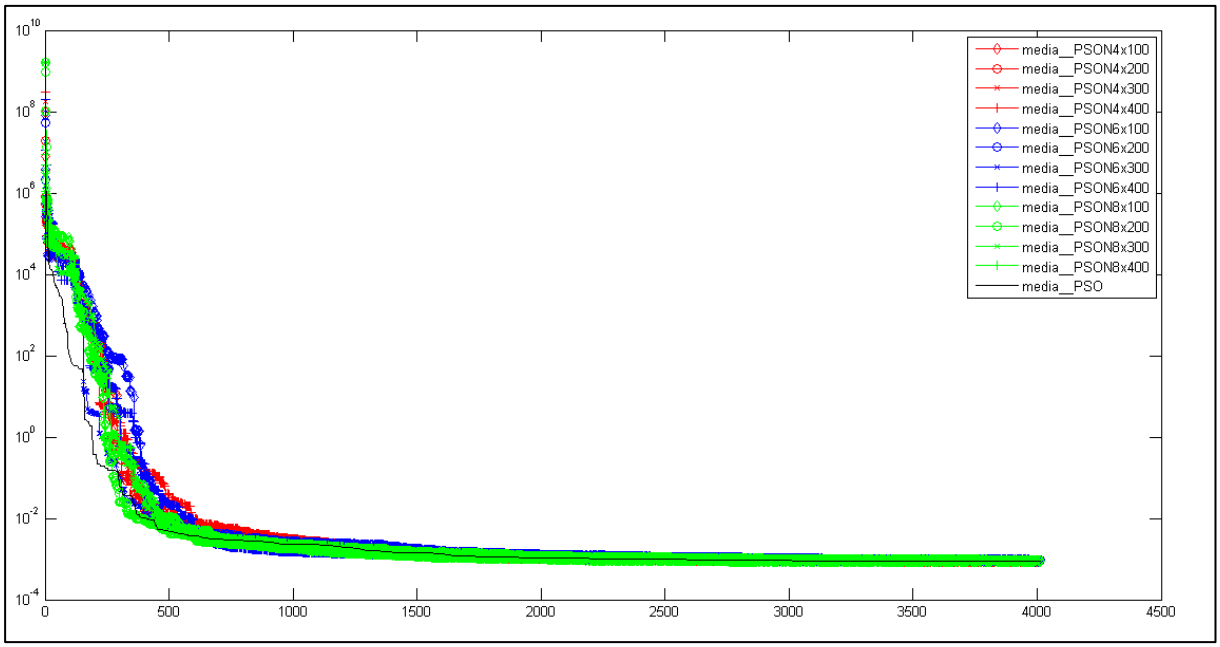

Figura 34. Gráfico da nota média versus simulações para o *Particle Swarm* aplicado ao circuito Ishibe. As configurações testadas são: rede escolhendo número de partículas igual 4vezes número de variáveis, utilizando os 100, 200, 300 e 400 melhores resultados, **PSON4X100**, **PSON4X200**, **PSON4X300** e **PSON4X400**; 6 vezes o número de variáveis, utilizando os 100, 200, 300 e 400 melhores resultados, **PSON6X100**, **PSON6X200**, **PSON6X300** e **PSON6X400**; 8 vezes o número de variáveis, utilizando os 100, 200, 300 e 400 melhores resultados, **PSON8X100**, **PSON8X200**, **PSON8X300** e **PSON8X400**; PSO simples, **PSO**.

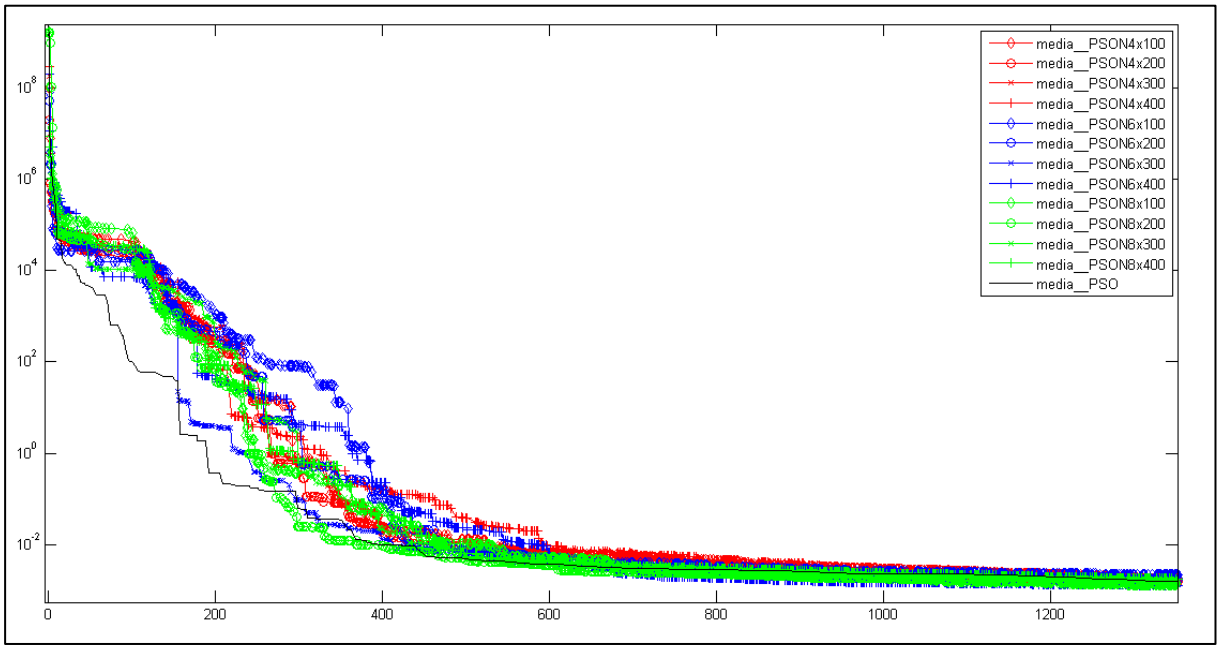

Figura 35. Gráfico da nota versus simulações para o *Particle Swarm*, aplicado ao circuito Ishibe. São mostrados em detalhes os resultados obtidos ao início das simulações.

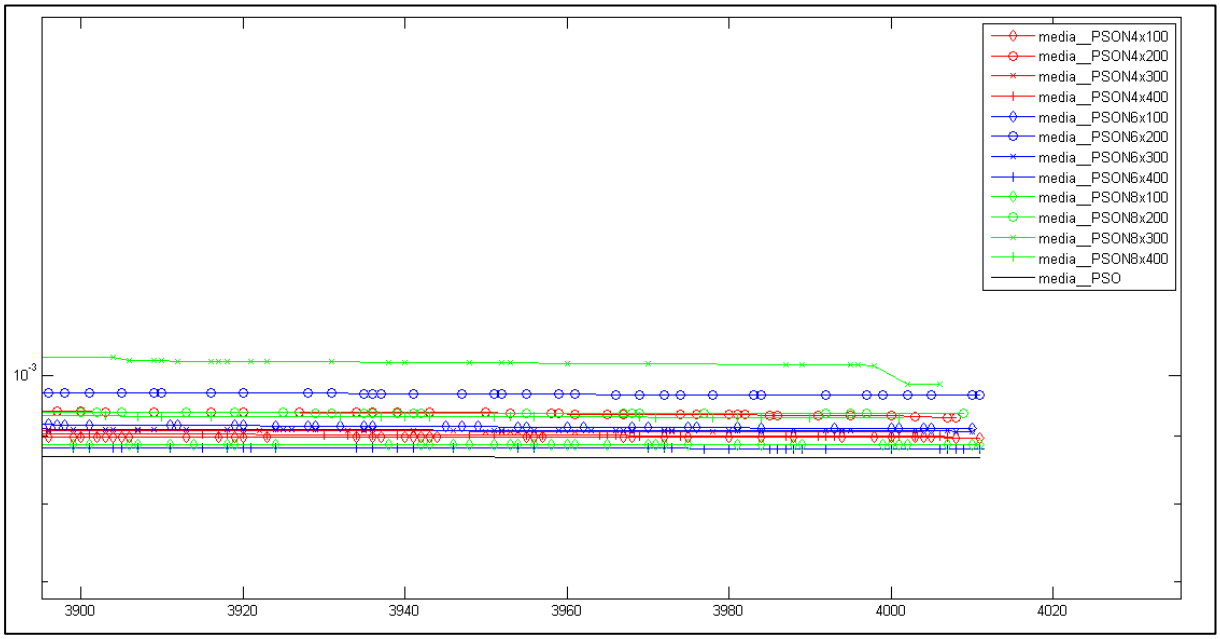

Figura 36. Gráfico da nota versus simulações para o *Particle Swarm*, aplicado ao circuito Ishibe. São mostrados em detalhes os resultados obtidos ao fim das simulações.

Analisando os gráficos, podemos perceber que o melhor resultado foi obtido pelo algoritmo PSO simples. As configurações que convergiram mais rápido para um resultado aceitável foi o próprio PSO simples e a rede neural avaliando 6 vezes o número de variáveis e utilizando 200 circuitos para retreinar a rede.

3.3.2 Circuito Ueno complete

Os resultados da aplicação de *Particle Swarm* no circuito Ueno completo são apresentados na **Figura 37**, **Figura 38** e **Figura 39**

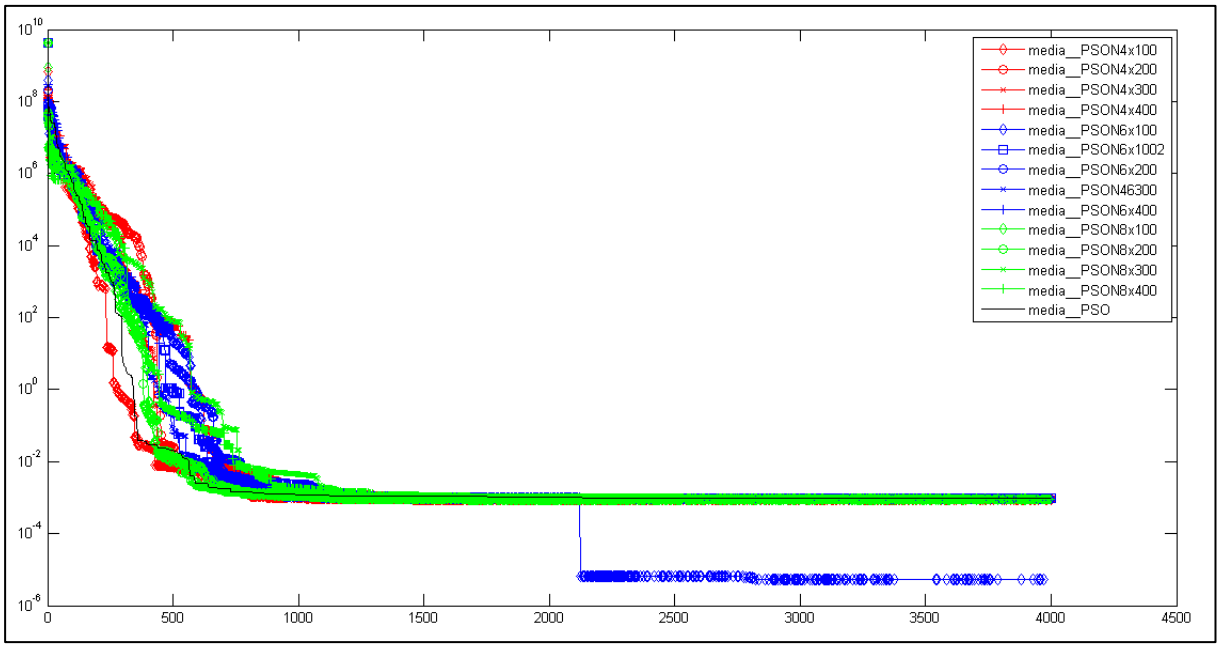

Figura 37. Gráfico da nota média versus simulações para o *Particle Swarm* aplicado ao circuito Ueno completo. As configurações testadas são: rede escolhendo número de partículas igual 4 vezes número de variáveis, utilizando os 100, 200, 300 e 400 melhores resultados, **PSON4X100**, **PSON4X200**, **PSON4X300** e **PSON4X400**; 6 vezes o número de variáveis, utilizando os 100, 200, 300 e 400 melhores resultados, **PSON6X100**, **PSON6X200**, **PSON6X300** e **PSON6X400**; 8 vezes o número de variáveis, utilizando os 100, 200, 300 e 400 melhores resultados, **PSON8X100**, **PSON8X200**, **PSON8X300** e **PSON8X400**; PSO simples, **PSO**.

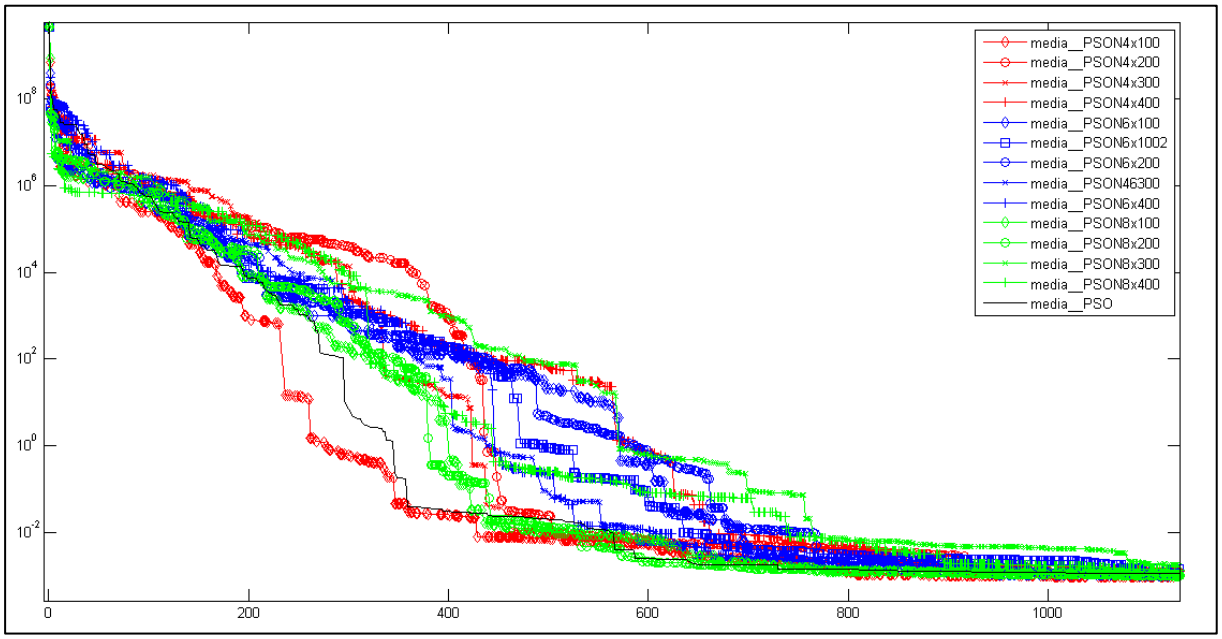

Figura 38. Gráfico da nota versus simulações para o *Particle Swarm*, aplicado ao circuito Ueno completo. São mostrados em detalhes os resultados obtidos ao início das simulações.

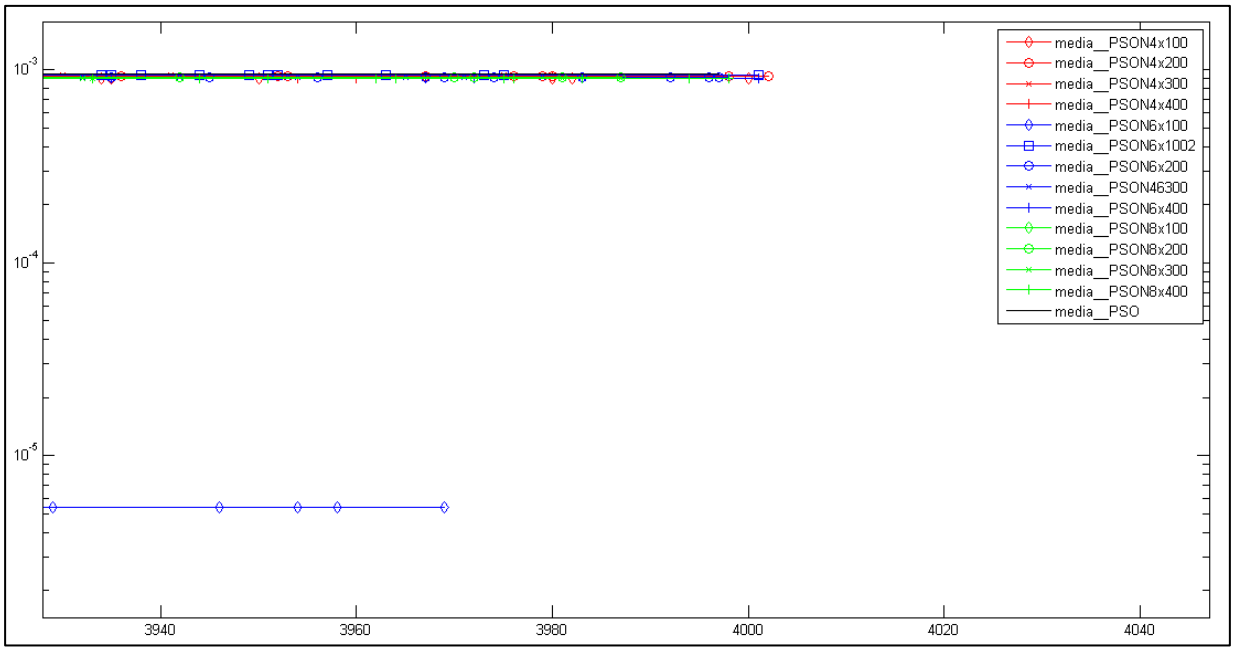

Figura 39. Gráfico da nota versus simulações para o *Particle Swarm*, aplicado ao circuito Ueno completo. São mostrados em detalhes os resultados obtidos ao fim das simulações.

Analisando os gráficos, podemos perceber que o melhor resultado foi obtido pela configuração avaliando 6 vezes o número de variáveis e utilizando 100 melhores resultados para re-treinar a rede. As configurações que convergiram mais rápido para um resultado aceitável foi o PSO simples e a rede neural avaliando 4 vezes o número de variáveis e utilizando 100 circuitos para retreinar a rede.

3.3.3 Circuito Ueno modificado

Os resultados da aplicação de *Particle Swarm* no circuito Ueno modificado são apresentados na **Figura 40**, **Figura 41** e **Figura 42**.

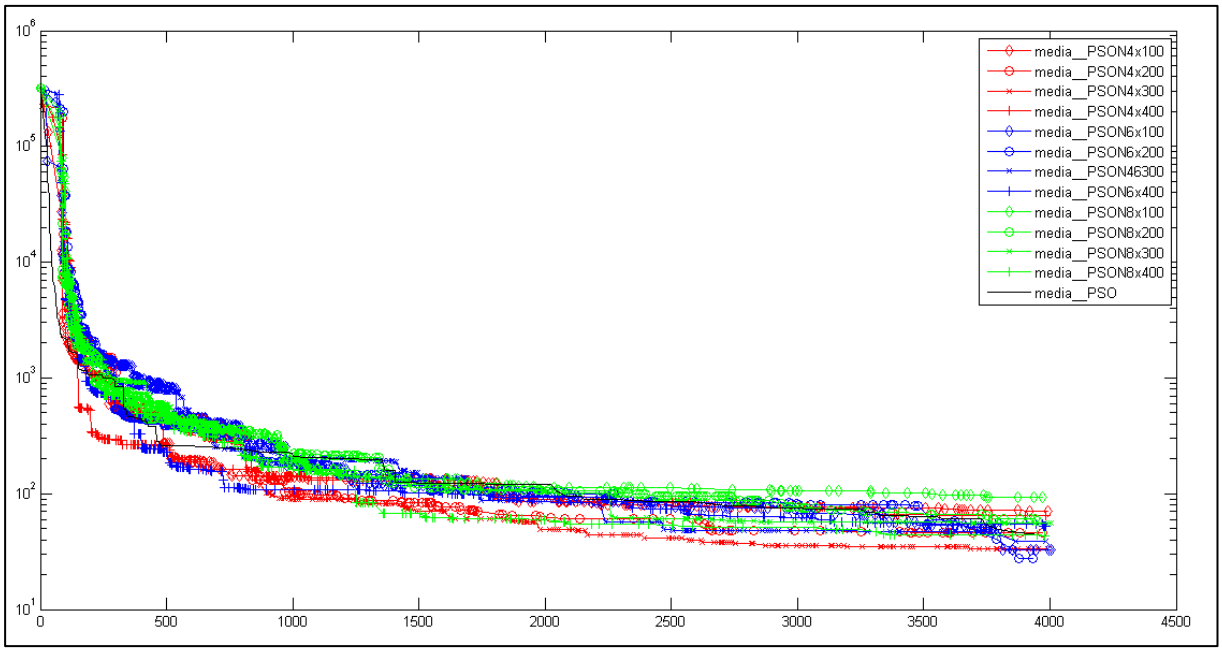

Figura 40. Gráfico da nota média versus simulações para o *Particle Swarm* aplicado ao circuito Ueno modificado. As configurações testadas são: rede escolhendo número de partículas igual 4 vezes número de variáveis, utilizando os 100, 200, 300 e 400 melhores resultados, **PSON4X100**, **PSON4X200**, **PSON4X300** e **PSON4X400**; 6 vezes o número de variáveis, utilizando os 100, 200, 300 e 400 melhores resultados, **PSON6X100**, **PSON6X200**, **PSON6X300** e **PSON6X400**; 8 vezes o número de variáveis, utilizando os 100, 200, 300 e 400 melhores resultados, **PSON8X100**, **PSON8X200**, **PSON8X300** e **PSON8X400**; PSO simples, **PSO**.

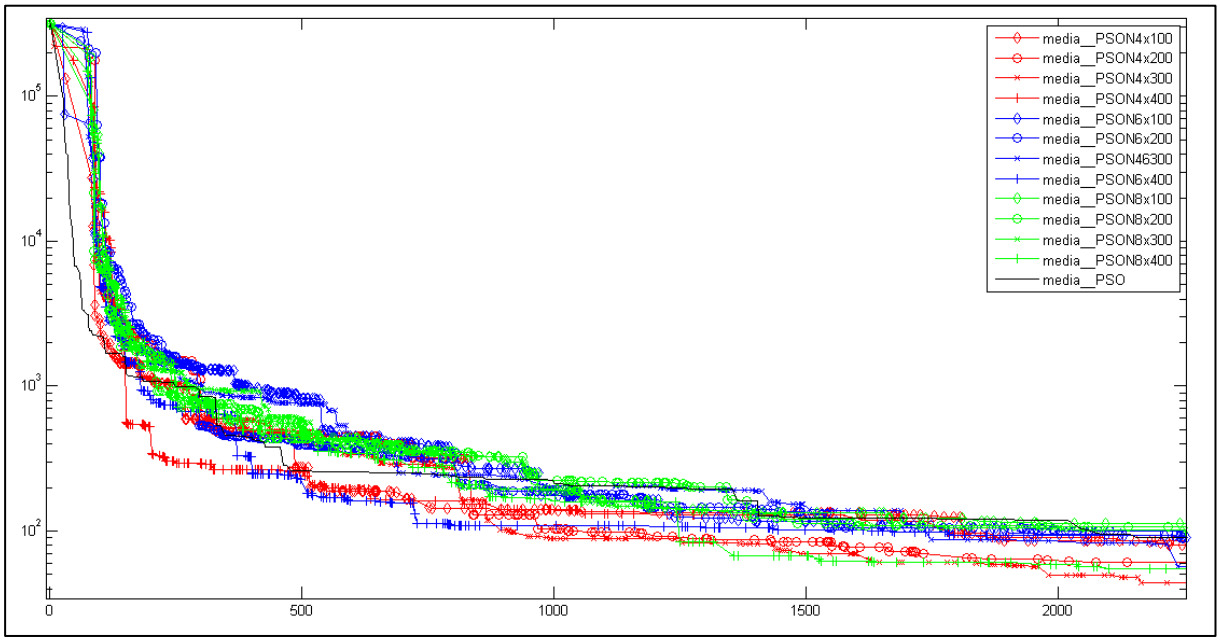

Figura 41. Gráfico da nota versus simulações para o *Particle Swarm*, aplicado ao circuito Ueno modificado. São mostrados em detalhes os resultados obtidos ao início das simulações.

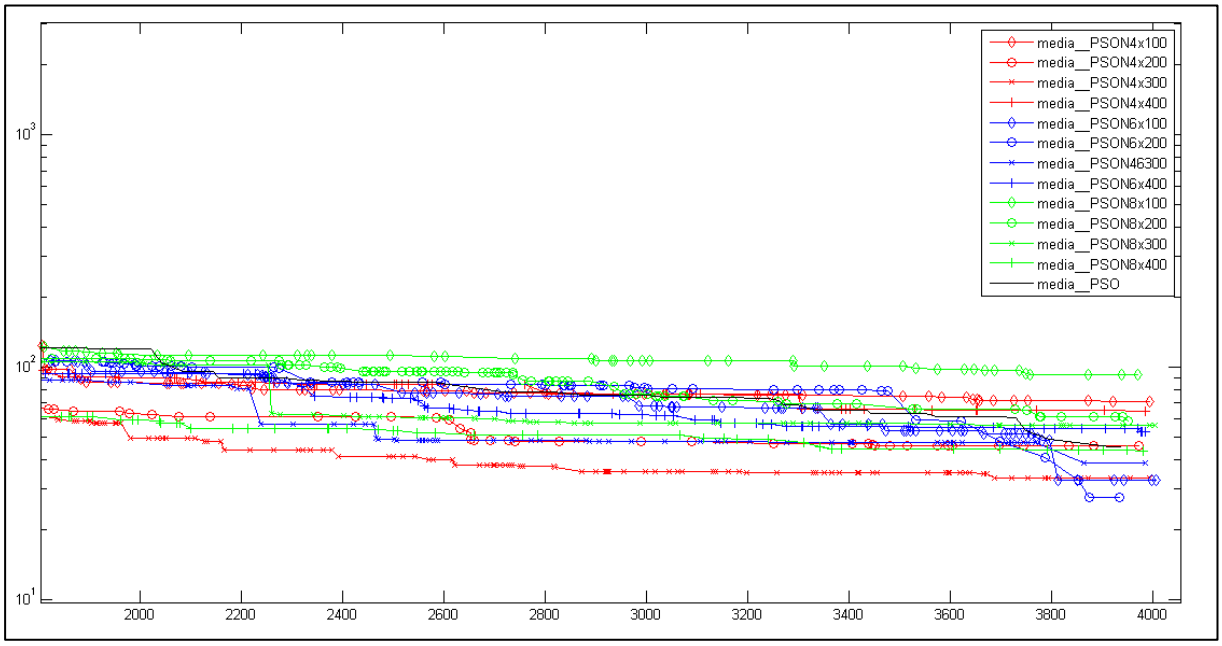

Figura 42. Gráfico da nota versus simulações para o *Particle Swarm*, aplicado ao circuito Ueno modificado. São mostrados em detalhes os resultados obtidos ao fim das simulações.

Analisando os gráficos, podemos perceber que o melhor resultado foi obtido pela configuração avaliando 6 vezes o número de variáveis e utilizando 200 circuitos para retreinar a rede. A configuração que convergiu mais rápido para um resultado aceitável foi a rede neural avaliando 4 vezes o número de variáveis e utilizando 400 circuitos para retreinar a rede.

Para comparação, a seguir são exibidos os gráficos com os melhores resultados de cada algoritmo para os circuitos Ishibe, Ueno completo e Ueno modificado, respectivamente na **Figura 43**, **Figura 44** e **Figura 45**.

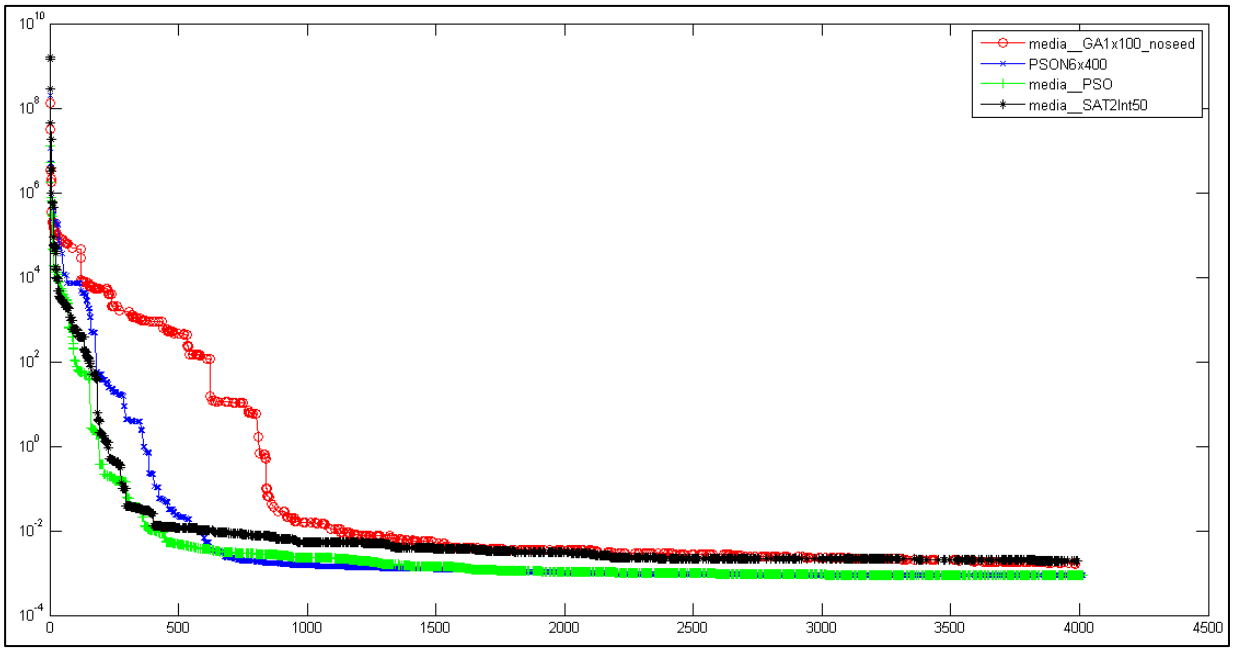

Figura 43. Gráfico com os melhores resultados de cada algoritmo para o circuito Ishibe. As configurações são: Algoritmo Genético com uma população de 100 indivíduos, **GA1x100**; *Particle Swarm* com rede neural avaliando 6 vezes o número de variáveis e Ntrei igual a 400, **PSON6x400**; PSO simples, **PSO**; *Simulated Annealing* com temperatura inicial 2 e máximo de alterações de 50, **SAT2Int50**.

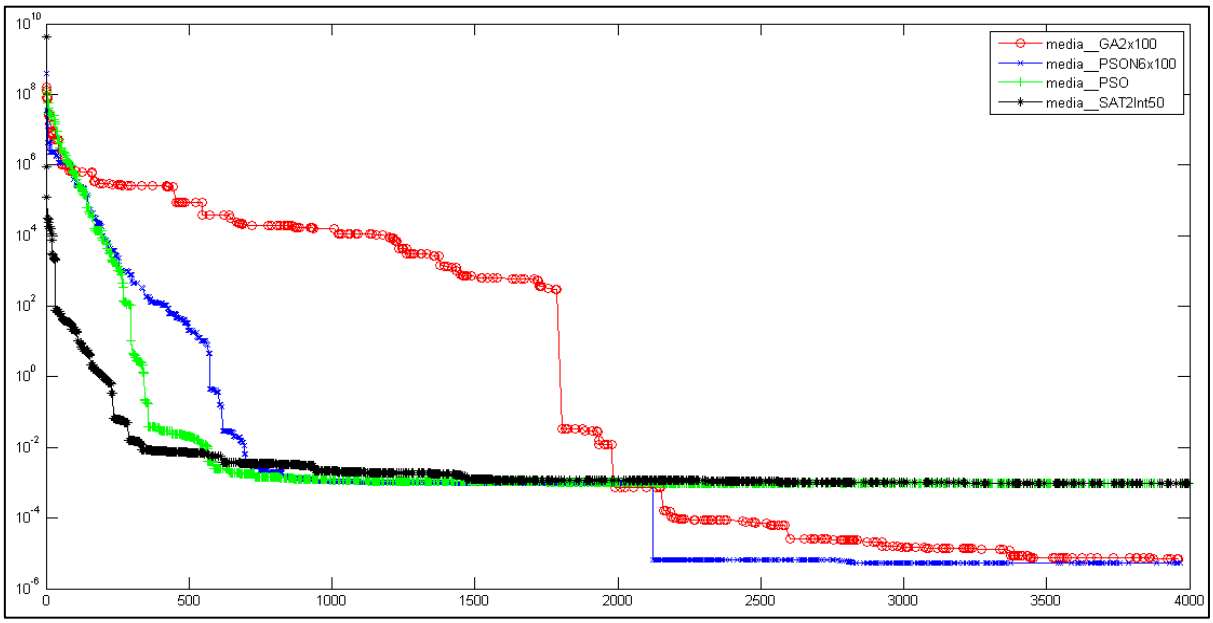

Figura 44. Gráfico com os melhores resultados de cada algoritmo para o circuito Ueno completo. As configurações são: Algoritmo Genético com duas populações de 100 indivíduos, **GA2x100**; *Particle Swarm* com rede neural avaliando 6 vezes o número de variáveis e Ntrei igual a 100, **PSON6x100**; PSO simples, **PSO**; *Simulated Annealing* com temperatura inicial 2 e máximo de alterações de 50, **SAT2Int50**.

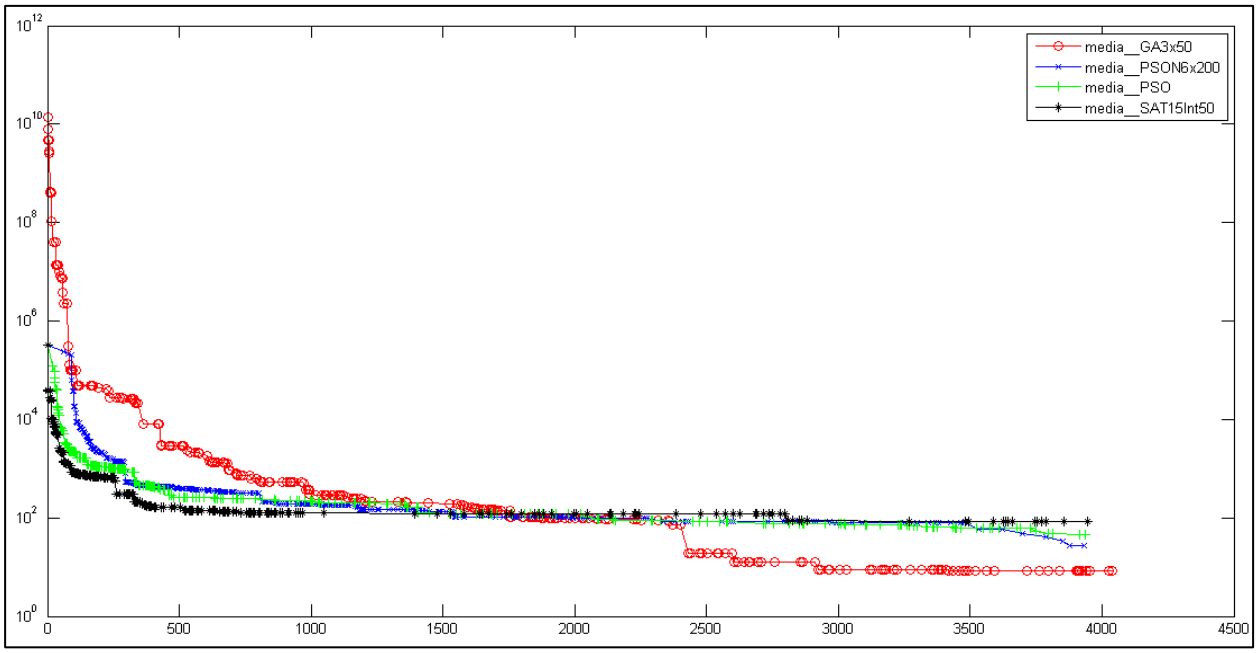

Figura 45. Gráfico com os melhores resultados de cada algoritmo para o circuito Ueno modificado. As configurações são: Algoritmo Genético com três populações de 50 indivíduos, **GA3x50**; *Particle Swarm* com rede neural avaliando 6 vezes o número de variáveis e Ntrei igual a 200, **PSON6x200**; PSO simples, **PSO**; *Simulated Annealing* com temperatura inicial 1,5 e máximo de alterações de 50, **SAT15Int50**.

4 Conclusão

Neste trabalho foi explicado o funcionamento geral de uma fonte de tensão de referência, o funcionamento específico de três topologias diferentes, o funcionamento de alguns algoritmos metaheurísticos e projetadas e otimizadas fontes de tensão aplicando algoritmos metaheuristicos. Ainda, a comparação de desempenho dos algoritmos foi realizada com base tanto na velocidade com que eles chegam a boas soluções, quanto na solução final apresentada.

Não obstante todos os algoritmos fornecerem soluções para as fontes, não foi possível determinar uma única configuração de algoritmo metaheuristico que seja, para todos os circuitos projetados, o melhor algoritmo. Como esperado do algoritmo genético, ele trouxe um resultado satisfatório para todos os circuitos, e caso seja possível realizar muitas avaliações, ele oferece bons resultados, porém ele foi o algoritmo mais lento de todos. O *Simulating Annealing* não alcançou os melhores resultados, mas foi rápido para fornecer bons resultados. O Algoritmo *Particle Swarm*, com rede neural, por sua vez, apresentou um desempenho bom, aparentando oferecer maior vantagem para circuitos mais complexos.

Para trabalhos futuros, seria interessante testar mais configurações dos algoritmos, como diminuir o máximo de trocas do *Simulated Annealing*, buscando realizar o *re-annealing* com mais frequência, e buscar melhorar a consistência dos testes do PSON para verificar o quanto ele é melhor em desempenho e resultado final.

Anexo

Modelo dos transistores utilizados para simulação.

```
.PARAM Wpad = 1.7um
.MODEL MODN NMOS LEVEL=49
* ---------------------------------------------------------------------
************************ SIMULATION PARAMETERS ************************
* ---------------------------------------------------------------------
* format    : HSPICE
* model     : MOS BSIM3v3
* process   : CS[ADFI]
* extracted : CSA C61417; 1998-10; ese(487)
* doc#      : 9933016 REV_N/C
* created   : 1999-01-12
* ---------------------------------------------------------------------
*                      TYPICAL MEAN CONDITION
* ---------------------------------------------------------------------
*       *** Flags ***
+MOBMOD =1.000e+00 CAPMOD =2.000e+00
*       *** Threshold voltage related model parameters ***
+K1     =6.044e-01
+K2     =2.945e-03 K3     =-1.72e+00 K3B    =6.325e-01
+NCH    =2.310e+17 VTH0   =4.655e-01
+VOFF   =-5.72e-02 DVT0   =2.227e+01 DVT1   =1.051e+00
+DVT2   =3.393e-03 KETA   =-6.21e-04
+PSCBE1 =2.756e+08 PSCBE2 =9.645e-06
+DVT0W  =0.000e+00 DVT1W  =0.000e+00 DVT2W  =0.000e+00
*       *** Mobility related model parameters ***
+UA     =1.000e-12 UB     =1.723e-18 UC     =5.756e-11
+U0     =4.035e+02
*       *** Subthreshold related parameters ***
+DSUB   =5.000e-01 ETA0   =3.085e-02 ETAB   =-3.95e-02
+NFACTOR=1.119e-01
*       *** Saturation related parameters ***
+EM     =4.100e+07 PCLM   =6.831e-01
+PDIBLC1=1.076e-01 PDIBLC2=1.453e-03 DROUT  =5.000e-01
+A0     =2.208e+00 A1     =0.000e+00 A2     =1.000e+00
+PVAG   =0.000e+00 VSAT   =1.178e+05 AGS    =2.490e-01
+B0     =-1.76e-08 B1     =0.000e+00 DELTA  =1.000e-02
+PDIBLCB=2.583e-01
*       *** Geometry modulation related parameters ***
+W0     =1.184e-07 DLC    =8.285e-09
+DWC    =2.676e-08 DWB    =0.000e+00 DWG    =0.000e+00
+LL     =0.000e+00 LW     =0.000e+00 LWL    =0.000e+00
+LLN    =1.000e+00 LWN    =1.000e+00 WL     =0.000e+00
+WW     =0.000e+00 WWL    =0.000e+00 WLN    =1.000e+00
+WWN    =1.000e+00
*       *** Temperature effect parameters ***
+AT     =3.300e+04 UTE    =-1.80e+00
+KT1    =-3.30e-01 KT2    =2.200e-02 KT1L   =0.000e+00
+UA1    =0.000e+00 UB1    =0.000e+00 UC1    =0.000e+00
```

```
+PRT    =0.000e+00
*       *** Overlap capacitance related and dynamic model parameters   ***
+CGDO   =2.100e-10 CGSO   =2.100e-10 CGBO   =1.100e-10
+CGDL   =0.000e+00 CGSL   =0.000e+00 CKAPPA =6.000e-01
+CF     =0.000e+00 ELM    =5.000e+00
+XPART  =1.000e+00 CLC    =1.000e-15 CLE    =6.000e-01
*       *** Parasitic resistance and capacitance related model parameters ***
+RDSW   =6.043e+02
+CDSC   =0.000e+00 CDSCB  =0.000e+00 CDSCD  =8.448e-05
+PRWB   =0.000e+00 PRWG   =0.000e+00 CIT    =1.000e-03
*       *** Process and parameters extraction related model parameters ***
+TOX    =7.700e-09 NGATE  =0.000e+00
+NLX    =1.918e-07
+XL     =5.000e-08 XW     =0.000e+00
*       *** Substrate current related model parameters ***
+ALPHA0 =0.000e+00 BETA0  =3.000e+01
*       *** Noise effect related model parameters ***
+AF     =1.400e+00 KF     =2.810e-27 EF     =1.000e+00
+NOIA   =1.000e+20 NOIB   =5.000e+04 NOIC   =-1.40e-12
+NLEV   =0
*       *** Common extrinsic model parameters ***
+ACM    =2
+RD     =0.000e+00 RS     =0.000e+00 RSH    =8.200e+01
+RDC    =0.000e+00 RSC    =0.000e+00
+LINT   =8.285e-09  WINT   =2.676e-08
+LDIF   =0.000e+00 HDIF   =6.000e-07 WMLT   =1.000e+00
+LMLT   =1.000e+00 XJ     =3.000e-07
+JS     =2.000e-05 JSW    =0.000e+00 IS     =0.000e+00
+N      =1.000e+00 NDS    =1000. VNDS   =-1.000e+00
+CBD    =0.000e+00 CBS    =0.000e+00 CJ     =9.300e-04
+CJSW   =2.800e-10 FC     =0.000e+00
+MJ     =3.100e-01 MJSW   =1.900e-01 TT     =0.000e+00
+PB     =6.900e-01 PHP    =9.400e-01
* ---------------------------------------------------------------------
.MODEL MODP PMOS LEVEL=49
* ---------------------------------------------------------------------
************************ SIMULATION PARAMETERS ************************
* ---------------------------------------------------------------------
* format    : HSPICE
* model     : MOS BSIM3v3
* process   : CS[ADFI]
* extracted : CSA C61417; 1998-10; ese(487)
* doc#      : 9933016 REV_N/C
* created   : 1999-01-12
* ---------------------------------------------------------------------
*                       TYPICAL MEAN CONDITION
* ---------------------------------------------------------------------
*
*       *** Flags ***
+MOBMOD =1.000e+00 CAPMOD =2.000e+00
*       *** Threshold voltage related model parameters ***
+K1     =5.675e-01
+K2     =-4.39e-02 K3     =4.540e+00 K3B    =-8.52e-01
```

```
+NCH    =1.032e+17 VTH0   =-6.17e-01
+VOFF   =-1.13e-01 DVT0   =1.482e+00 DVT1   =3.884e-01
+DVT2   =-1.15e-02 KETA   =-2.56e-02
+PSCBE1 =1.000e+09 PSCBE2 =1.000e-08
+DVT0W  =0.000e+00 DVT1W  =0.000e+00 DVT2W  =0.000e+00
*       *** Mobility related model parameters ***
+UA     =2.120e-10 UB     =8.290e-19 UC     =-5.28e-11
+U0     =1.296e+02
*       *** Subthreshold related parameters ***
+DSUB   =5.000e-01 ETA0   =2.293e-01 ETAB   =-3.92e-03
+NFACTOR=8.237e-01
*       *** Saturation related parameters ***
+EM     =4.100e+07 PCLM   =2.979e+00
+PDIBLC1=3.310e-02 PDIBLC2=1.000e-09 DROUT  =5.000e-01
+A0     =1.423e+00 A1     =0.000e+00 A2     =1.000e+00
+PVAG   =0.000e+00 VSAT   =2.000e+05 AGS    =3.482e-01
+B0     =2.719e-07 B1     =0.000e+00 DELTA  =1.000e-02
+PDIBLCB=-1.78e-02
*       *** Geometry modulation related parameters ***
+W0     =4.894e-08 DLC    =-5.64e-08
+DWC    =3.845e-08 DWB    =0.000e+00 DWG    =0.000e+00
+LL     =0.000e+00 LW     =0.000e+00 LWL    =0.000e+00
+LLN    =1.000e+00 LWN    =1.000e+00 WL     =0.000e+00
+WW     =0.000e+00 WWL    =0.000e+00 WLN    =1.000e+00
+WWN    =1.000e+00
*       *** Temperature effect parameters ***
+AT     =3.300e+04 UTE    =-1.35e+00
+KT1    =-5.70e-01 KT2    =2.200e-02 KT1L   =0.000e+00
+UA1    =0.000e+00 UB1    =0.000e+00 UC1    =0.000e+00
+PRT    =0.000e+00
*       *** Overlap capacitance related and dynamic model parameters   ***
+CGDO   =2.100e-10 CGSO   =2.100e-10 CGBO   =1.100e-10
+CGDL   =0.000e+00 CGSL   =0.000e+00 CKAPPA =6.000e-01
+CF     =0.000e+00 ELM    =5.000e+00
+XPART  =1.000e+00 CLC    =1.000e-15 CLE    =6.000e-01
*       *** Parasitic resistance and capacitance related model parameters ***
+RDSW   =1.853e+03
+CDSC   =6.994e-04 CDSCB  =2.943e-04 CDSCD  =1.970e-04
+PRWB   =0.000e+00 PRWG   =0.000e+00 CIT    =1.173e-04
*       *** Process and parameters extraction related model parameters ***
+TOX    =7.700e-09 NGATE  =0.000e+00
+NLX    =1.770e-07
+XL     =5.000e-08 XW     =0.000e+00
*       *** Substrate current related model parameters ***
+ALPHA0 =0.000e+00 BETA0  =3.000e+01
*       *** Noise effect related model parameters ***
+AF     =1.290e+00 KF     =1.090e-27 EF     =1.000e+00
+NOIA   =1.000e+20 NOIB   =5.000e+04 NOIC   =-1.40e-12
+NLEV   =0
*       *** Common extrinsic model parameters ***
+ACM    =2
+RD     =0.000e+00 RS     =0.000e+00 RSH    =1.560e+02
+RDC    =0.000e+00 RSC    =0.000e+00
```

```
+LINT   =-5.64e-08  WINT   =3.845e-08
+LDIF   =0.000e+00 HDIF   =6.000e-07 WMLT   =1.000e+00
+LMLT   =1.000e+00 XJ     =3.000e-07
+JS     =2.000e-05 JSW    =0.000e+00 IS     =0.000e+00
+N      =1.000e+00 NDS    =1000. VNDS   =-1.000e+00
+CBD    =0.000e+00 CBS    =0.000e+00 CJ     =1.420e-03
+CJSW   =3.800e-10 FC     =0.000e+00
+MJ     =5.500e-01 MJSW   =3.900e-01 TT     =0.000e+00
+PB     =1.020e+00 PHP    =9.400e-01
* ---------------------------------------------------------------------

.MODEL VERT10 PNP
* ---------------------------------------------------------------------
************************* SIMULATION PARAMETERS ************************
* ---------------------------------------------------------------------
* format    : ELDO, AccusimII, Continuum
* model     : BJT
* process   : C35[A-B][3-4][A-C][1-3]
* revision  : 2.0;
* extracted : C35[A-B][3-4][A-C][1-3] B11264.L2; 2002-11; hhl (5481)
* doc#      : Eng-182
* ---------------------------------------------------------------------
*                      TYPICAL MEAN CONDITION
* ---------------------------------------------------------------------
*
+IS     =2.3330e-17 IRB    =4.3770e-06
+IKF    =1.3760e-03 BF     =5.9810e+00 NF     =9.9250e-01
+ISE    =6.5290e-16 NE     =1.7760e+00 VAF    =1.9420e+02
+IKR    =1.9410e-04 BR     =9.8740e-02 NR     =9.9470e-01
+ISC    =2.8430e-14 NC     =1.1490e+00 VAR    =1.0320e+01
+RBM    =1.0000e+00
+RB     =2.1380e+02
+RE     =9.7360e+00
+RC     =4.5400e+01
+TF     =6.4800e-10
+
+EG     =1.1150e+00 XTI    =5.5300e+00 XTB    =2.2500e+00
+CJE    =1.4880e-13 VJE    =1.0200e+00 MJE    =5.4882e-01
+CJC    =4.3387e-14 VJC    =5.3000e-01 MJC    =3.1214e-01
+
* ---------------------------------------------------------------------
```

Referências

[1] SEDRA, A. S.; SMITH, K.C. “Microeletrônica”, 4ª ed., 2005

[2] Austriamicrosystems. 0,35μm CMOS process technology. Disponível em: http://www.ams.com/eng/Products/Full-Service-Foundry/Process-Technology/CMOS

[3] ENGELBRECHT, R.M. “Projeto de fontes de referência de tensão em tecnologia CMOS”. São Carlos, 2007. Trabalho de Conclusão de Curso – Departamento de Engenharia Elétrica, Escola de Engenharia de São Carlos, USP.

[4] SILVA, E.S.C. “Projeto de fontes de referência de baixa tensão em tecnologia CMOS”. São Carlos, 2008. Trabalho de Conclusão de Curso – Departamento de Engenharia Elétrica, Escola de Engenharia de São Carlos, USP.

[5] SASSI, M.M.F.S. “Projeto de fontes de tensão de referência através de metaheurísticas”. São Carlos, 2013. Trabalho de pós Graduação - Departamento de Engenharia Elétrica, Escola de Engenharia de São Carlos, USP.

[6] NAVARRO, J.; ISHIBE, E.I. A sub-1v reference voltage supplier. In: IEEE International Symposium on Circuits and Systems (ISCAS), 2011, Rio de Janeiro. Proceedings… Piscataway, NJ: IEEE, 2011

[7] UENO, K. “A 300 nW, 15ppm/ºC, 20ppm/V CMOS Voltage Reference Circuit Consisting of Subthreshold MOSFETs”. **IEEE Journal of Solid-State Circuits,** Piscataway, NJ, v. 44, n. 7, July 2009.

[8] LUKE, S. “Essentials of Metaheuristics”, 2 ed., June 2013

[9] MATHWORKS. Matlab. Disponível em: http://www.mathworks.com/products/matlab/

[10] MATHWORKS. Global Optimization Toolbox. Disponível em: http://www.mathworks.com/products/global-optimization/

[11] SYNOPSYS. Hspice. Disponível em: <http://www.synopsys.com/tools/verification/amsverification/circuitsimulation/hspice/Pages/default.aspx>

Apêndice A

Parametrização do Circuito Ishibe

```
.include param
.options statfl=1 NXX Noelck=1 MEASFILE=1 NOTOP RUNLVL=1 dcon=1
Mp1    vd     b      f      vd     MODP W='X7*1u'    L='X1*1u'     M='M1'
Mp2    vd     b      b      vd     MODP W='X7*1u'    L='X1*1u'
Mp3    f      a      d      vd     MODP W='X7*1u'    L='X2*1u'     M='M1'
Mp4    vd     b      c      vd   MODP   W='X8*1u'    L='X1*1u'
Mp5    a      a      vd     vd     MODP W='X9*1u'    L='X3*1u'
Mn1    0      d      d      0      MODN W='X10*1u'   L='X4*1u'
Mn2    j      d      e      0      MODN W='X10*1u'   L='X4*1u'
Mn3    e      c      b      0      MODN W='X10*1u'   L='X5*1u'
Mn4    0      d      a      0      MODN W='X11*1u'   L='X4*1u'
Mn5    0      c      c      0      MODN W='X12*1u'   L='X6*1u'
R1     j      0      'X13*1k'
M10    vd     b      t      vd     MODP W='X7*1u'    L='X1*1u'     M='int(X17)'
M11    vd     b      y      vd     MODP W='X7*1u'    L='X1*1u'     M='int(X16)'
Mc10   s      a      t      vd     MODP W='X7*1u'    L='X2*1u'     M='int(X17)'
M12    u      a      y      vd     MODP W='X7*1u'    L='X2*1u'     M='int(X16)'
.param R2 = 'X14*X13*1k*0.5/(26m*int(X17)*log(M1))'
R2     u      s      R2
R3     s      0      'R2*X15*Vref/(1.12-Vref)'
Q1     0      0      u      Vert10
Vdd    vd     0      1.5
.end
```

Arquivo param, utilizado

```
.include ..\..\Models\Model35
.param Vref = 0.50
.Param   X1= 10.255  X2= 10.952  X3= 18.050  X4= 1.173  X5= 13.870  X6= 10.751
.Param   X7= 163.967  X8= 95.912  X9= 10.406  X10= 38.404  X11= 31.292  X12= 1.011
X13= 87.137
.Param   X14= 0.988  X15= 1.343  X16= 3.286  X17= 3.323
** ( 10.255  10.952  18.050  1.173  13.870  10.751  163.967  95.912  10.406  38.404  31.292
1.011  87.137  0.988  1.343  3.286  3.323  )
.Param   M1= 3.00
.DC  TEMP   -10.0   90.0   4.0
.MEASURE DC    VreftPP   PP    V(s)
.MEASURE DC    VrefAV    AVG  V(s)
.DC   Vdd   0.50V   2.50V  0.10V
.MEASURE DC   VrefvPP   PP  v(s) from=1.0 to=2.5
.MEASURE DC   Vdd_cur  find  I(Vdd) when v(Vd) = 1.75
.MEASURE   DC   Mn1_weak   find      par('((abs(I1(Mn1))/(26m*(1+Lx9(Mn1)/Lx7(Mn1)))   -
Lx7(Mn1))/Lx7(Mn1))')  when v(Vd) = 1.75
```

Apêndice B

Parametrização do Circuito Ueno Completo

```
.include param
.options statfl=1 NXX Noelck=1 MEASFILE=1 NOTOP RUNLVL=1 dcon=1
Mp1   d8    d11   vd    vd    MODP W='X2*1u'    L='X1*1u'    M=M1
Mp2   d9    d11   vd    vd    MODP W='X2*1u'    L='X1*1u'
Mp3   d3    d11   vd    vd    MODP W='X2*1u'    L='X1*1u'    M=M2
Mp4   d5    d11   vd    vd    MODP W='X2*1u'    L='X1*1u'    M=M2
Mp5   s     d11   vd    vd    MODP W='X2*1u'    L='X1*1u'    M=M2
Mp6   d10   d10   vd    vd    MODP W='X4*1u'    L= 'X3*1u'
Mp7   d11   d10   vd    vd    MODP W='X4*1u'    L='X3*1u'
Mn1   g1    g1    0     0     MODN W='X6*1u'    L='X5*1u'
Mn2   d2    g1    s2    0     MODN W='X6*1u'    L='X5*1u'
Mn3   d3    d3    d4    0     MODN W='M3*1u'    L='X7*1u'    M='int(X8)'
Mn4   d4    d3    0     0     MODN W='M3*1u'    L='X7*1u'    M='int(X9)'
Mn5   d5    d5    d6    0     MODN W='M3*1u'    L='X7*1u'    M='int(X8)'
Mn6   d6    d5    d4    0     MODN W='M3*1u'    L='X7*1u'    M='int(X9)'
Mn7   s     s     d6    0     MODN W='X10*1u'   L='X7*1u'
Mn8   d8    d8    g1    0     MODN W='X6*1u'    L='X11*1u'
Mn9   d9    d8    d2    0     MODN W='X6*1u'    L='X11*1u'
Mn10  d10   d9    d14   0     MODN W='X13*1u'   L='X12*1u'
Mn11  d11   d8    d14   0     MODN W='X13*1u'   L='X12*1u'
Mn12  d14   g1    0     0     MODN W='X6*1u'    L='X5*1u'    M= M4
MR1   s2    s     0     0     MODN W='X15*1u'   L='X14*1u'
Vdd   vd    0     1.5
.end
```

Arquivo param utilizado

```
.include ..\..\Models\Model35
.param Vref = 0.60
.Param   X1= 22.441  X2= 72.123  X3= 7.794  X4= 10.522  X5= 1.344  X6= 140.774
.Param   X7= 3.049  X8= 21.932  X9= 1.157  X10= 8.356  X11= 9.407  X12= 12.687  X13=
48.114
.Param   X14= 35.171  X15= 2.519
** ( 22.441  72.123  7.794  10.522  1.344  140.774  3.049  21.932  1.157  8.356  9.407
12.687  48.114  35.171  2.519  )
.Param   M1= 3.00  M2= 1.00  M3= 10.00  M4= 0.10
.DC  TEMP   -10.0   90.0   4.0
.MEASURE DC    VreftPP   PP    V(s)
.MEASURE DC    VrefAV     AVG  V(s)
.DC   Vdd   0.50V   2.50V  0.10V
.MEASURE DC   VrefvPP   PP  v(s) from=1.0 to=2.5
.MEASURE DC   Vdd_cur  find  I(Vdd) when v(Vd) = 1.75
.MEASURE   DC   Mn2_weak   find      par('((abs(I1(Mn2))/(26m*(1+Lx9(Mn2)/Lx7(Mn2)))   -
Lx7(Mn2))/Lx7(Mn2))')  when v(Vd) = 1.75
```

Apêndice C

Parametrização do Circuito Ueno Modificado

```
.include param
.options statfl=1 NXX Noelck=1 MEASFILE=1 NOTOP RUNLVL=1 dcon=1
Mp1    g1     d11    vd     vd     MODP W='X2*1u'     L='X1*1u'     M=M1
Mp2    d2     d11    vd     vd     MODP W='X2*1u'     L='X1*1u'
Mp3    d3     d11    vd     vd     MODP W='X2*1u'     L='X1*1u'     M=M2
Mp4    d5     d11    vd     vd     MODP W='X2*1u'     L='X1*1u'     M=M2
Mp5    s      d11    vd     vd     MODP W='X2*1u'     L='X1*1u'     M=M2
Mp6    d10    d10    vd     vd     MODP W='X4*1u'     L='X3*1u'
Mp7    d11    d10    vd     vd     MODP W='X4*1u'     L='X3*1u'
Mn1    g1     g1     0      0      MODN W='X6*1u'     L='X5*1u'
Mn2    d2     g1     s2     0      MODN W='X6*1u'     L='X5*1u'
Mn3    d3     d3     d4     0      MODN W='M3*1u'     L='X7*1u'     M='int(X8)'
Mn4    d4     d3     0      0      MODN W='M3*1u'     L='X7*1u'     M='int(X9)'
Mn5    d5     d5     d6     0      MODN W='M3*1u'     L='X7*1u'     M='int(X8)'
Mn6    d6     d5     d4     0      MODN W='M3*1u'     L='X7*1u'     M='int(X9)'
Mn7    s      s      d6     0      MODN W='X10*1u'    L='X7*1u'
Mn10   d10    d2     0      0      MODN W='X6*1u'     L='X5*1u'     M=M4
Mn11   d11    g1     0      0      MODN W='X6*1u'     L='X5*1u'     M=M4
MR1    s2     s      0      0      MODN W='X12*1u'    L='X11*1u'
Vdd    vd     0      1.5
.MEASURE  DC  m6_weakp2  find  par('((abs(I1(Mn1))/(26m*(1+Lx9(Mn1)/Lx7(Mn1)))  -
Lx7(Mn1))/Lx7(Mn1))') when v(Vd) = 1.75
.end
```

Arquivo param utilizado

```
.include ..\..\Models\Model35
.param Vref = 0.70
.Param   X1= 1.154  X2= 54.479  X3= 19.495  X4= 76.827  X5= 7.124  X6= 88.159
.Param   X7= 1.747  X8= 6.696  X9= 10.301  X10= 1.746  X11= 9.627  X12= 3.562
** ( 1.154  54.479  19.495  76.827  7.124  88.159  1.747  6.696  10.301  1.746  9.627  3.562
)
.Param   M1= 3.00  M2= 1.00  M3= 10.00  M4= 0.10
.DC  TEMP   -10.0   90.0   4.0
.MEASURE DC    VreftPP   PP    V(s)
.MEASURE DC    VrefAV     AVG  V(s)
.DC   Vdd   0.50V   2.50V  0.10V
.MEASURE DC   VrefvPP   PP  v(s) from=1.0 to=2.5
.MEASURE DC   Vdd_cur  find  I(Vdd) when v(Vd) = 1.75
.MEASURE  DC  Mn1_weak  find  par('((abs(I1(Mn1))/(26m*(1+Lx9(Mn1)/Lx7(Mn1)))  -
Lx7(Mn1))/Lx7(Mn1))')  when v(Vd) = 1.75
```

Apêndice D

Função Metaheu, responsável pela chamada das condições, de acordo com o algoritmo escolhido.

```
global modo;
global nvars;
global genesLB;
global inicialSol;
global dif;
global modoind;
global namext;
global cont;
global circuito;
global slash;

namext = '';
prompt={'Number of Runs','Max simulations (more than 1 run)','Extension of
the output file', 'seed'};
name='Algoritm Running';
numlines=1;
defaultanswer={'1','4000', 'A1', ''};
entrada=inputdlg(prompt,name,numlines,defaultanswer);
namext = entrada{3};
system(['del ' circuito slash 'results' slash 'optimos.' modo namext]);

profile on;
switch modo
case'GA'
    modoind=1;
    load optimtool_GA.mat;
    optimproblem.lb = zeros(1, nvars);
    optimproblem.ub = ones(1, nvars);
    optimproblem.nvars= nvars;
    optimproblem.options.PopInitRange = [optimproblem.lb; optimproblem.ub];
    optimproblem.options.PopulationSize=[100 100];

optimproblem.options.Generations=round(str2num(entrada{2})/sum(optimproblem
.options.PopulationSize))-1;
% inicializa os geradores para se repetir resultados
if ~isempty(entrada{4})
         rand('seed', str2double(entrada{4}));
         randn('seed', strdouble(entrada{4}));
end;

    Best.scoreT = Inf('double');
if str2double(entrada{1}) == 1
        optimtool(optimproblem);
elsefor i=1:str2double(entrada{1});
fprintf('\n_____________________  inicio da %1.0f otimizacao   Melhor score:
%3g  _____________________\n\n ', i, Best.scoreT);
cont = 0; Best.score = Inf('double'); Best.scoreT = Inf('double');
           ga(optimproblem);
end;
end;
```

```
case'SA'
    modoind=2;
    load optimtool_SA.mat;
    optimproblem.lb = zeros(1, nvars);
    optimproblem.ub = ones(1, nvars);
    optimproblem.x0 = zeros(1, nvars);
    optimproblem.options.InitialTemperature=1.5;
    optimproblem.options.ReannealInterval= 75;
    optimproblem.options.MaxFunEvals=[str2num(entrada{2})];
    j=1;
for i=1:length(genesLB)
if (dif(i) ~= 0)
        optimproblem.x0(j)= (inicialSol(i) - genesLB(i))/dif(i);
        j=j+1;
end;
end;

% gera condicoes iniciais
if ~isempty(entrada{4})
        samples  = RandStream('mt19937ar', 'Seed', str2double(entrada{4}));
% this stream is used to generate the initial conditions
        rand('seed', str2double(entrada{4}));
        optimproblem.x0 = rand(samples, 1,nvars);
end;

    Best.scoreT = Inf('double');
if str2double(entrada{1}) == 1
        optimtool(optimproblem);
elsefor i=1:str2double(entrada{1});
fprintf('\n_____________________  inicio da %1.0f otimizacao   Melhor score:
%3g  _____________________\n\n ', i, Best.scoreT);
cont = 0; Best.score = Inf('double');
           simulannealbnd(optimproblem);
if ~isempty(entrada{4})
            optimproblem.x0 = rand(samples, 1,nvars);
end;
end;
end;

case'PS'
    modoind=3;
    load optimtool_PS.mat;
    optimproblem.lb = zeros(1, nvars);
    optimproblem.ub = ones(1, nvars);
    optimproblem.x0 = zeros(1, nvars);
    optimproblem.options.MaxFunEvals = str2num(entrada{2});
    j=1;
for i=1:length(genesLB)
if (dif(i) ~= 0)
          optimproblem.x0(j)= (inicialSol(i) - genesLB(i))/dif(i);
          j=j+1;
end;
end;

% gera condicoes iniciais
if ~isempty(entrada{4})
```

```
samples  = RandStream('mt19937ar', 'Seed', str2double(entrada{4})); % this
stream is used to generate the initial conditions
         rand('seed', str2double(entrada{4}));
         optimproblem.x0 = rand(samples, 1,nvars);
end;

     Best.scoreT = Inf('double');
if str2double(entrada{1}) == 1
        optimtool(optimproblem);
elsefor i=1:str2double(entrada{1});
fprintf('\n___________________  inicio da %1.0f otimizacao   Melhor score:
%3g  ___________________\n\n ', i, Best.scoreT);
cont = 0; Best.score = Inf('double');
           patternsearch(optimproblem);
if ~isempty(entrada{4})
            optimproblem.x0 = rand(samples, 1,nvars);
end;
end;
end;

case'MM'
    load optimtool_MM.mat;
    optimproblem.lb = zeros(1, nvars);
    optimproblem.ub = ones(1, nvars);
    optimproblem.x0 = zeros(1, nvars);
    j=1;
for i=1:length(genesLB)
if (dif(i) ~= 0)
          optimproblem.x0(j)= (inicialSol(i) - genesLB(i))/dif(i);
          j=j+1;
end;
end;
    modoind=4;
    optimtool(optimproblem);

case'SAM'
     modoind=5;
%set the options
    optimproblem = struct(...
'fitnessfcn', @fitness,...
'Display', 'crossovers',...  % it can be 'final', 'iter', 'crossovers', or
'none'
'TolFun', 1e-6,...
'ObjectiveLimit', -1e+20,...
'TolCon', 1e-6,...
'CoolSched',@(T) (.8*T),...
'InitTemp',20,...                % initial temperature
'MaxTriesWithoutBest',10000,...  %max number of attempts without a new best
before finishing
'MaxSuccess',200,...
'MaxTries',300,...
'StopTemp',1e-8,...
'StopVal',-Inf,...
'MaxtoLocal',25,...
'TimeLimit',inf,...     % time in seconds
'LockOn',1,...          % lock to variables that are showing progress
'Crossover',1,...       % able or disable crossovers
'MaxSim',3000,...
```

```
'Initial_Sigma',1);

    optimproblem.lb = zeros(1, nvars);
    optimproblem.ub = ones(1, nvars);
    optimproblem.x0 = zeros(1, nvars);
   optimproblem.MaxSim = str2num(entrada{2});
j=1;
for i=1:length(genesLB)
if (dif(i) ~= 0)
        optimproblem.x0(j)= (inicialSol(i) - genesLB(i))/dif(i);
        j=j+1;
end;
end;
if ~isempty(entrada{4})
        samples  = RandStream('mt19937ar', 'Seed', str2double(entrada{4}));
% this stream is used to generate the initial conditions
        rand('seed', str2double(entrada{4}));
        optimproblem.x0 = rand(samples, 1, nvars);
end;

    Best.scoreT = Inf('double');
for i=1:str2double(entrada{1});
      fprintf('\n_________________  inicio da %1.0f otimizacao   Melhor
score: %3g  _________________\n\n ', i, Best.scoreT);
cont = 0; Best.score = Inf('double');
      annealwithcrossovers(optimproblem);
if ~isempty(entrada{4})
        optimproblem.x0 = rand(samples, 1,nvars);
end;
end;
% SAM;

case'SCE'
   optimproblem.lb = zeros(1, nvars);
   optimproblem.ub = ones(1, nvars);
   optimproblem.x0 = zeros(1, nvars);
   j=1;
for i=1:length(genesLB)
if (dif(i) ~= 0)
         optimproblem.x0(j)= (inicialSol(i) - genesLB(i))/dif(i);
         j=j+1;
end;
end;
   fun = 'fitness';
   modoind=6;
   saida= SCE(fun, optimproblem.x0, optimproblem.lb, optimproblem.ub);

case'PSO'
   modoind=7;
   optimproblem.lb = zeros(1, nvars);
   optimproblem.ub = ones(1, nvars);
   optimproblem.x0 = zeros(1, nvars);
   fun = 'fitness';
   j=1;
for i=1:length(genesLB)
if (dif(i) ~= 0)
       optimproblem.x0(j)= (inicialSol(i) - genesLB(i))/dif(i);
       j=j+1;
```

```
end;
end;
if ~isempty(entrada{4})
        samples  = RandStream('mt19937ar', 'Seed', str2double(entrada{4}));
% this stream is used to generate the initial conditions
        rand('seed', str2double(entrada{4}));
        optimproblem.x0 = rand(samples, 1, nvars);
end;

   Best.scoreT = Inf('double');
for i=1:str2double(entrada{1});
      fprintf('\n____________________  inicio da %1.0f otimizacao   Melhor
score: %3g  ____________________\n\n ', i, Best.scoreT);
cont = 0; Best.score = Inf('double');
      saida= PSO(fun, optimproblem.x0, optimproblem.lb, optimproblem.ub,
str2double(entrada{2}), length(optimproblem.x0));
%saida= PSON2(fun, optimproblem.x0, optimproblem.lb, optimproblem.ub,
6*length(optimproblem.x0), length(optimproblem.x0), 300, 'radbas',
round(nvars/2), str2double(entrada{2}));
if ~isempty(entrada{4})
         optimproblem.x0 = rand(samples, 1,nvars);
end;
end;

case'DE'
   optimproblem.lb = zeros(1, nvars);
   optimproblem.ub = ones(1, nvars);
   optimproblem.x0 = zeros(1, nvars);
    j=1;
for i=1:length(genesLB)
if (dif(i) ~= 0)
          optimproblem.x0(j)= (inicialSol(i) - genesLB(i))/dif(i);
          j=j+1;
end;
end;
   fun = 'fitness'; VTR = 1.e-8;
   modoind=8;
   Best.scoreT = Inf('double');
Rnvars = length(optimproblem.x0);

for i=1:str2num(entrada{1});
    fprintf('\n____________________  inicio da %1.0f otimizacao   Melhor
score: %3g  ____________________\n\n ', i, Best.scoreT);
cont = 0; Best.score = Inf('double');
    saida= devec3(fun, VTR, Rnvars, optimproblem.lb, optimproblem.ub, [],
6*Rnvars, round(str2num(entrada{2})/(6*Rnvars)));
end;

case'EvN'
   modoind=9;
   fun = 'fitness';
optimproblem.x0 = zeros(1, nvars);
   j=1;
for i=1:length(genesLB)
if (dif(i) ~= 0)
       optimproblem.x0(j)= (inicialSol(i) - genesLB(i))/dif(i);
       j=j+1;
end;
```

```
end;

if ~isempty(entrada{4})
        samples  = RandStream('mt19937ar', 'Seed', str2double(entrada{4}));
% this stream is used to generate the initial conditions
        rand('seed', str2double(entrada{4}));
        optimproblem.x0 = rand(samples, 1, nvars);
end;

%saida = EvNeu (fun,  optimproblem.x0, nvars, 200, 200, 40, 300, 'radbas',
[nvars, nvars]);
   Best.scoreT = Inf('double');
for i=1:str2double(entrada{1})
fprintf('\n_____________________  inicio da %1.0f otimizacao   Melhor score:
%3g  _____________________\n\n ', i, Best.scoreT);
cont = 0; Best.score = Inf('double');
     saida = EvNeu16_4(fun,  optimproblem.x0, nvars, 95, 200, 40, 300,
'radbas', round(nvars/2));
if ~isempty(entrada{4})
         optimproblem.x0 = rand(samples, 1,nvars);
end;
end;

case'PSON'
   modoind=10;
   optimproblem.lb = zeros(1, nvars);
   optimproblem.ub = ones(1, nvars);
   optimproblem.x0 = zeros(1, nvars);
   fun = 'fitness';
   j=1;
for i=1:length(genesLB)
if (dif(i) ~= 0)
       optimproblem.x0(j)= (inicialSol(i) - genesLB(i))/dif(i);
       j=j+1;
end;
end;
if ~isempty(entrada{4})
        samples  = RandStream('mt19937ar', 'Seed', str2double(entrada{4}));
% this stream is used to generate the initial conditions
rand('seed', str2double(entrada{4}));
        optimproblem.x0 = rand(samples, 1, nvars);
end;

   Best.scoreT = Inf('double');
for i=1:str2double(entrada{1});
      fprintf('\n_____________________  inicio da %1.0f otimizacao   Melhor
score: %3g  _____________________\n\n ', i, Best.scoreT);
cont = 0; Best.score = Inf('double');
      saida= PSON2(fun, optimproblem.x0, optimproblem.lb, optimproblem.ub,
4*length(optimproblem.x0), length(optimproblem.x0), 400, 'radbas',
round(nvars/2), str2double(entrada{2}));
if ~isempty(entrada{4})
         optimproblem.x0 = rand(samples, 1,nvars);
end;
end;

end
```

Apêndice E

Função *Fitness*, utilizada para calcular scores dos circuitos

```
function [sc, sci] = fitness(x)
% sc da o scores total e sci sao os valores parciais

global slash;
global simulador;
global circuito;
global ParDados
global genesLB;

global V_alvo ;
global precisao;
global Vmin;
global Vmax;
global Tmin;
global Tmax;
global TC;
global RL;
global pPot;
global AreaCir;
global weakTrans;
global strTrans;
global const;
global pesos;

global Best
global BestRes;

global modoind;
global modo;
global dif;
global cont;
global namext;


V_alvo = str2num(ParDados{1, 2});
precisao = str2num(ParDados{2, 2})/100;
vet = str2num(ParDados{3, 2});  Vmin = vet(1);  Vmax= vet(2);
vet = str2num(ParDados{4, 2});  Tmin = vet(1);  Tmax= vet(2);
TC = str2num(ParDados{5, 2});
RL = str2num(ParDados{6, 2});
pPot = str2num(ParDados{7, 2});
AreaCir = str2num(ParDados{8, 2});
weakTrans = strread(ParDados{9, 2}, '%s');
strTrans = strread(ParDados{10, 2}, '%s');
const = str2num(ParDados{11, 2});
pesos = str2num(ParDados{12, 2});

deltaT = Tmax - Tmin;
deltaV = Vmax - Vmin;
```

```
% os parametros x (gerados) vao de 0 a 1;  a partir deles sao gerados os
parametros xr para o arquivo de simulacao
%xr = x.*dif+genesLB;

j=1;
for i=1:length(genesLB)
if (dif(i) ~= 0)
        xr(i)= (x(j)*dif(i)+genesLB(i));
        j=j+1;
else xr(i)=genesLB(i);
end;
end;

% calculo de area do circuito .. nao ta sendo usado
cond = [circuito slash 'AreaCirMea.m'];
if exist(cond)
   eval(['cd ', circuito]);
   AreaMed = AreaCirMea(xr);
   eval('cd ..');
else AreaMed = 0;
end;

% arquivo param tem os comando para simulacao
arq = fopen([circuito slash 'param'],'w');
param_mod(arq, xr);
fclose(arq);

disp _____________________________

cont=cont +1;
fprintf('simulação = %d\n', cont);

% executa a simulacao
[~, b] = system([simulador circuito slash 'circuito.sp']);
% [~, b] = system (['START /Realtime/wait/min C:\synopsys\Hspice_A-
2008.03\BIN\hspice_mt.exe ' circuito '\circuito.sp']);

% read the simulation results
arq = fopen('circuito.ms0','r');
Meas=LeMeas(arq, 3);
fclose(arq);

% verifica se a simulacao gerou todos resultados
if length(Meas) > (7+ length(weakTrans) + length(strTrans))

% calculate the circuit performance
%TC
if (Meas(2) <= 0)  FTC = Inf('double'); TCm = Inf('double');
else
    TCm = (Meas(1)/Meas(2))*(1/deltaT)*(1e6);
    FTC = (TCm-TC)/TC; %FTCn = FTC;
if(TCm < TC) FTC = 0;
end
end

%Vref
if (Meas(2) <= 0)  FVref = Inf('double');
```

```
else FVref = abs((Meas(2)-V_alvo)/min(V_alvo, Meas(2))); %FVrefn = FVref;
if (FVref <= precisao) FVref = 0;
end;
end

%RL
if (Meas(5) <= 0)  FRL = Inf('double'); RLm = Inf('double');
else
    RLm = (abs(Meas(5))/deltaV)*(1/Meas(2))*1e6;
    FRL = (RLm-RL)/RL;  %FRLn = FRL;
if (RLm < RL)  FRL = 0;
end
end

%Power consumption
Fpot= -1.0e6*Meas(6)*(Vmax+Vmin)/(pPot*2);

% Area
Farea = AreaMed/AreaCir;

% verify if transistors are in weak invertion
Weakin=0;

for i= 1: length(weakTrans);
     j= 6+i;
    Meas(j)= abs(Meas(j));
if Meas(j)> 0.12 Weakin=10*Meas(j) +Weakin; end;
end;

Strin=0;
for i= 1: length(strTrans);
     j= i+6+length(weakTrans);
if Meas(j) < 0.1 Strin=10*abs(Meas(j)-0.1) + Strin;  end;
end;

%final score
sc = ( pesos(1)*FVref + pesos(2)*FTC  + pesos(3)*FRL + pesos(4)*Fpot +
pesos(5)*Weakin + pesos(6)*Strin + pesos(7)*Farea)^2;
sci = [FRL   FTC   FVref  Fpot  Weakin Strin];
%sci = [FRLn   FTCn   FVrefn  Fpot  Weakin Strin];
% save the best solution
if(Best.score > sc)
    arq = fopen([circuito slash 'results' slash 'optimos.' modo
namext],'a+');
    fprintf(arq,'%d  %.3g\n', cont, sc);
    fclose(arq);
    beep;
    Best.score = sc;
    fprintf('*>');
    arq = fopen([circuito slash 'paramop'],'w');
    param_mod(arq,xr);
    fprintf(arq,'*Score=%.2g  TC= %.2gppm  (FTC= %.2g)  RL = %.3gppm  (FRL
= %.2g) Vref =%.2fV (FVref =%.2f) Pot.=%.2guW (Fpot= %.2g) \n*Weakin= %.2g
Strin= %.2g Area=%.2gum2 (Farea= %.2g)\n\n',sc, TCm, FTC, RLm, FRL,
Meas(2), FVref, -1.0e6*Meas(6)*(Vmax+Vmin)/2, Fpot, Weakin, Strin, AreaMed,
Farea);
    fclose(arq);
```

```
if(Best.scoreT > sc)
      Best.scoreT = sc;
      Best.parameters = xr;
      arq = fopen([circuito slash 'paramopT'],'w');
param_mod(arq,xr);

      fprintf(arq,'*Score=%.2g  TC= %.2gppm  (FTC= %.2g)  RL = %.3gppm
(FRL = %.2g) Vref =%.2fV (FVref =%.2f) Pot.=%.2guW (Fpot= %.2g) \n*Weakin=
%.2g  Strin= %.2g Area=%.2gum2 (Farea= %.2g)\n\n',sc, TCm, FTC, RLm, FRL,
Meas(2), FVref, -1.0e6*Meas(6)*(Vmax+Vmin)/2, Fpot, Weakin, Strin, AreaMed,
Farea);
fclose(arq);
      BestRes{modoind} = Best;
end;

end;

% performance results
fprintf('Score=%.3g  TC= %.2gppm  (FTC= %.2g)  RL = %.3gppm  (FRL = %.2g)
Vref =%.2fV (FVref =%.2f) Pot.=%.2guW (Fpot= %.2g)\nWeakin= %.2g  Strin=
%.2g Area=%.2gum2 (Farea= %.2g)\n\n',sc, TCm, FTC, RLm, FRL, Meas(2),
FVref, -1.0e6*Meas(6)*(Vmax+Vmin)/2, Fpot, Weakin, Strin, AreaMed, Farea);

% problemas na simulacao
else sc = inf('double');
    sci = [ inf('double')  inf('double')  inf('double')  inf('double')
inf('double')  inf('double')];
    fprintf('Score=%.3g  \n\n',sc);
end;

% used parameters
for i = 1:length(xr)
if (mod(i, 10) == 0) fprintf('\n');
end
    fprintf('X%i= %1.1f   ', i, xr(i));
end;
fprintf('\n');

end
```