Proposta de Pesquisa

Algoritmos Evolutivos Multi-objetivo Aplicado ao Design de Hardware

Tiago Moreira Trocoli da Cunha

Orientador: Prof. Alexandre Cláudio Botazzo Delbem

Resumo: Este projeto tem como objetivo o desenvolvimento de uma meta-heurística multi-objectiva aplicada ao design de *hardware*.

Palavras-Chave: Otimização Multi-objetiva, MOEA, MOSPO, *Hardware* Evolutivo, *Particle Swarm Optimization*, Algoritmos Evolutivos.

The content of this report are sole responsibility of the authors.
O conteúdo deste relatório é de única responsabilidade dos autores.

Departamento de Sistemas de Computação
Instituto de Ciências Matemáticas e de Computação | Universidade de São Paulo |
Av. Trabalhador São-carlense, 400 · Centro · São Carlos/SP · CEP 13566-590 · Brasil ·
www.icmc.usp.br

Sumário

1. Introdução

O desenvolvimento de dispositivos eletrônicos é significante para a industria. Cada vez mais vem aumentando o grau de sofisticação que esta sendo demandado no setor industrial, impulsionado pela incorporação crescente da tecnologia eletrônica-digital em um número imensurável de dispositivos. No entanto, o desenvolvimento de métodos tradicionais que dependem de regras criadas há décadas juntamente com mão-de-obra humana pra desenvolver sistemas com uma complexidade crescente vem se tornando um verdadeiro gargalo de desenvolvimento, pra tanto existem dois caminhos possíveis.

O primeiro, é contratar mais desenvolvedores, mais engenheiros, físicos e mais *experts* no assunto. O que é custoso. O segundo é simplificar o circuito ao impor mais e mais abstrações no *design*, um exemplo disso são as linguagens de descrição de *hardware*. Isto resulta em uma gradual perda de potencial no comportamento do circuito [26].

Mas, recentemente uma nova área que aplica técnicas de algoritmos evolutivos para *hardware* vem emergindo. Estas técnicas mostram uma terceira opção - usar meta-heurísticas evolutivas para desenvolver hardwares, diminuindo a dependência da mão-de-obra humana e abrindo espaço para uma nova era de desenvolvimento. Esta nova área de pesquisa é chamada de hardware evolutivo

O uso de algoritmos evolutivos para design de *hardware* traz uma série de benefícios, permitindo automação de projetos(design) e inovação em várias áreas de aplicação. Algumas dessas importantes áreas onde *hardware* evolutivo é aplicada incluem [27]:

- Automação de projeto de baixo custo.
- Desenvolvimento de sistemas adaptativos
- Desenvolvimento de *hardware* de alta complexidade
- Desenvolvimento de sistemas tolerante á falhas
- Inovação em projetos de *hardware*

Vale ressaltar que o presente documento não tem a intenção de fornecer detalhes sobre as áreas de computação evolutiva e inteligência computacional, mas sim, prover de forma resumida alguns dos objetivos de estudo da pesquisa.

2. Revisão Bibliográfica

Essa sessão apresenta uma breve revisão bibliográfica sobre conceitos de algoritmos evolutivos, inteligência de enxame que pode ser considerada um área da computação evolutiva, otimização multi-objetiva e *hardware* evolutivo nesta ordem.

2.1 Algoritmos Evolutivos

Podemos entender evolução como um processo de otimização em que o objetivo é otimizar o organismo a sobreviver em ambientes dinâmicos e competitivos [13]. Esta afirmação se baseia na Teoria da Evolução das Espécies, de Chales Darwin. Segundo esta teoria, indivíduos que melhor se adaptam ao ambiente têm maiores chances de sobreviverem e se reproduzirem, isto é chamado de seleção natural. Enquanto existir algumas variações entre eles, irá haver uma seleção dos indivíduos que têm as variações mais vantajosas para sobreviverem. Se tais variações são hereditárias, então a reprodução irá levar há uma progressiva evolução desta população [8].

No entanto, Chales Darwin não sabia como se transmitia a hereditariedade. Posteriormente, com trabalhos de outros cientistas foi criado a Teoria Neo-darwiniana ou Teoria da Síntese Evolutiva Moderna, que inclui os conhecimentos da genética [9]. Basicamente, essa teoria faz referência a quatro principais conclusões. Primeiro, a evolução pode ser elucidada pelas mutações e pela recombinação gênica, norteadas pelo processo de seleção natural. Segundo, os fenômenos evolutivos fundamentam-se nos mecanismos genéticos. Terceiro, a teoria se baseia focada na população em vez de um indivíduo. Quarto, a variabilidade genética é o principal fator da evolução [10].

A partir disso, algoritmos evolutivos(AE) é definido como meta-heurísticas estocástica que simulam no computador os processos de evolução para resolver problemas. Existem vários algoritmos evolutivos na literatura, no entanto, sua forma genérica é elucidada a seguir [14] :

Algoritmo Evolutivo Genérico

Seja t = 0, o contador de gerações;
Crie e inicialize a população P(t) consistindo de n indivíduos de m dimensões;
Enquanto *as condições de parada forem falsas* faça:
    Avalie o *fitness*, F(x_i(t)), de cada indivíduo do vetor x_i;
    Realize a reprodução para gerar descendentes;
    Selecione uma nova população, P(t+1);
    Avance para um nova geração, t = t + 1;
Fim

2.2 Inteligência de Enxame

Inteligência de Enxame ou SI, do inglês *Swarm Intelligence* originou de estudos de colônias ou de enxames de organismos sociais [17]. Estudos de comportamentos socias de organismos de exames incentivou o desenvolvimento de algoritmos de optimização(PSO) e de clusterização. Por exemplo, a simulação da coreografia de revoadas de pássaros resultou no desenvolvimento do algoritmo de otimização por enxame de partículas(PSO) , e estudos de comportamentos coletivo de formigas resultou na otimização por colônia de formigas.

Pode-se formalizar um exame como um grupo de agentes inteligentes que se comunicam entre si, direta ou indiretamente, através de suas ações no ambiente [16]. As interações entre agentes resultam em uma estratégia coletiva de resolução de problemas. Então, SI se refere a comportamentos de resolução de problemas que emergem das interações de agentes, e inteligência de exame computacional(CSI) se refere aos algoritmos que simulam tais comportamentos. Mais formalmente, SI é um propriedade de um sistema através dos quais comportamentos coletivos de agentes que interagem localmente entre si e o ambiente por regras simples, resulta no surgimento de um padrão global funcional e coerente [15]. O objetivo da inteligência de exame computacional é simular comportamentos simples de indivíduos e suas interações, com o intuito de se obter um comportamento mais complexo que pode ser usado em resoluções de problemas, principalmente em otimização.

Logo, PSO é um processo de busca baseada em população, onde indivíduos(ou agentes) são chamados de partículas e sua população de enxame. Neste algoritmo, cada partícula “voa” pelo espaço de busca multi-dimensional, ajustando sua posição no espaço de acordo com sua própria experiência e também das partículas vizinhas. Então, uma partícula, pode fazer uso da melhor posição encontrada por ela e pela sua vizinhança e assim ajustar sua posição em direção a solução ótima. O efeito disso é que, as partículas “voam” em direção ao ótimo, enquanto rondam uma certa área em volta da melhor solução atual. O algoritmo é descrito a seguir:

PSO

Crie e inicialize uma população com n agentes de m dimensões
repita
  para *cada agente* i = 1, …, n **faça**:
    //defina a melhor posição individual
    se f(xi) < f(yi) **então**:
      yi = xi;
    fim
    //defina a melhor posição da vizinhança(local)
    se f(yi) < f(ŷi) **então**:
      ŷ = yi;
    fim
  fim
  para *cada agente* i = 1, …, n **faça**:
    atualize a velocidade do agente pela equação (1);
    atualize a posição do agente pela equação (2);
  fim
até *a condição de parada for verdadeira*;

Resumidamente, cada partícula é tratada como um ponto de um espaço com m-dimensões. A partícula i é representada como $x_i = (x_{i1},x_{i2},...,x_{im})$. A melhor posição encontrada pela partícula i é recordada e representada por $y_i = (y_{i1},y_{i2},...,y_{im})$. A melhor partícula entre a vizinhança de i é $\hat{y}_i = (\hat{y}_{i1},\hat{y}_{i2},...,\hat{y}_{im})$. E a velocidade da partícula i é representada por $v_i = (v_{i1},v_{i2},...,v_{im})$. A partícula é manipulada de acordo com a equação abaixo:

$$v_{id} = w_i * v_{id} + c_1 * rand() * (y_{id} - x_{id}) + c_2 * rand() * (\hat{y}_{id} - x_{id}) \quad (1)$$

$x_{id} = x_{id} + v_{id}$ (2)

onde d é a dimensão (1<=d<=m), c_1 e c_2 são constantes positivas chamadas de aceleração e rand() é uma função aleatória no intervalode de 0 a 1, e w é o peso inercial.

2.3 Optimização Multi-Objetiva

A maioria das otimizações do mundo real são multi-objetivas, pelo menos implicitamente, se não, explicitamente [19], como é o caso de design de *hardware*. Isto significa que problemas reais têm vários objetivos, muitas vezes em conflito. Por exemplo:

- Quando se quer comprar um carro, se quer maximizar o conforto e minimizar o custo. O carro com maior conforto pode ser muito custoso, ao passo que, carro com menor custo pode ser desconfortável.
- Quando se quer construir uma ponte, se quer minimizar seu custo e maximizar sua resistência. Isto entra no mesmo esquema anterior, pode-se usar isopor para minimizar o custo, mas sua resistência é mínima. Ou usar titânio, mas seria muito custo.
- Quando se quer desenvolver um sistema de controle, se quer minimizar o tempo de subida e ao mesmo tempo, a distorção de um sinal. Um controlador cujo sinal tem tempo de resposta muito baixo teria uma distorção muito alta, por outro lado, se o mesmo tivesse uma distorção quase nula, o tempo de subida seria muito lento [18].

Optimização multi-objetiva, também chamada de optimização multi-critéria pode ser escrita como se segue [19]:

$$f : X \rightarrow \Re^n, \quad min\, f(x) \;=\; min\, [f1(x), f2(x), \ldots, fn(x)]^T$$

Sendo que, se alguma função é maximizada, é equivalente a minimizar sua forma negativa, ou seja, max [$f_k(x)$] = min [-$f_k(x)$]. O $n \in N$ e $n \geq 2$ é o número de funções objetivas e o X é o conjunto viável, o elemento $x^* \in X$ é chamado de solução viável, o vetor $z^* := f(x^*) \in R$ é chamado de vector objetivo ou vetor resultado.

Em otimização multi-objetiva, geralmente não existe uma solução viável que

minimiza todas as funções objetivo. Então, como pode-se avaliar as diferentes soluções dentro de um conjunto de mais de um objetivo? Se existisse, dois objetivos para serem minimizados e duas solução, podemos dizer que uma solução é “melhor” que a outra caso os resultados dos dois objetivos da primeira for menor que a da segunda. Mas, e se a primeira tiver um objetivo que é menor que a da outra e o segundo objetivo maior? As duas soluções estarão empatadas. O conceito de pareto foi criado com a ideia similar descrita anteriormente. Definindo matematicamente:

- Dominação: Uma solução viável x* domina uma outra x se as seguintes condições forem satisfeitas: $f_i(x^*) \leq f_i(x), \forall\ i \in [1,n]$ e $\exists\ j \in [1,n]$ tal que $f_i(x^*) < f_i(x)$. Neste caso, a notação matemática é $x^* \prec x$.
- Dominação fraca: Uma solução viável x* domina fracamente uma outra x se $f_i(x^*) \leq f_i(x), \forall\ i \in [1,n]$. Cuja a notação é $x^* \preccurlyeq x$.
- Não dominado: Uma solução x* se diz não dominada se não existe uma solução viável x, tal que, ela domina x*.
- Conjunto Ótimo de Pareto: É o conjunto de todo x* que não é dominado. Seja P_s tal conjunto, então:
  $P_s = \{x^* \in X : [\neg \exists\ x\ : (f_i(x^*) \leq f_i(x), \forall\ i \in [1,n]\ e\ \exists\ j \in [1,n]\ f_i(x^*) < f_i(x))]\}$
- *Pareto Front*: É o conjunto de todos os vetores f(x) correspondente ao conjunto ótimo de pareto. Seja, P_f o *pareto front*, então: $P_f = \{f(x^*)\ :\ x^* \in P_s\}$.

No caso de optimização com uma função, só existe um objetivo, que é encontrar a solução ótima global. Mesmo quando tal optimização é multi-modal muitos algoritmos só procuram uma solução ótima [21]. No caso de optimização multi-objetiva, existem várias soluções ótimas, então o objetivo de procurar uma única solução não é adequado. O objetivo se torna dois, de maximizar o número de soluções não dominadas e maximizar a diversidade delas.

2.4 Hardware Evolutivo

Hardware Evolutivo é um novo campo que usa algoritmos evolutivos (EA) para desenvolver circuitos eletrônico-digital sem a necessidade de mão-de-obra humana. Reunindo o campo de hardware reconfigurável , inteligência artificial, tolerância a falhas e sistemas autônomos.

Para otimizar um circuito lógico, é crucial antes em pensar numa forma de mapear uma cadeia de números inteiros ou reais em um grafo tal que seus vértices

representam portas lógicas ou módulos(como *multiplexer* e ALU) e suas arestas os fios. Um dos modelos que esta pesquisa usará é descrito por Julian F. Miller [24]. Tal representação, fornece características interessantes listadas a seguir [25]:

- Tanto circuito combinacional tanto sequencial pode ser mapeado.
- Esta representação é flexível, permitindo implementações de uma grande variedade de funções sequenciais e combinacionais.
- Se referindo ao aspecto prático, este modelo pode ser facilmente transferido para *hardwares* reconfiguráveis.
- Este modelo ainda não foi bem sucedido em evoluir circuitos combinacionais de grande escala.
- Esta representação é mais promissora no design de circuitos que não podem ser projetados por heurísticas dos software de CAD(*Computer Aided Design*).

No entanto, a pesquisa não ficará restrita a somente o modelo de Julian F. Miller, mas outros também propostos por outros autores. Este projeto usará também técnicas para complementar o modelo e analisando-as em relação a eficiência para o problema proposto. Pode-se citar a técnica cujo o tamanho da representação é dinâmica, o paradigma de memória, a especiação e a técnica da auto-modificação evolutiva.

3. Proposta de Pesquisa

Este projeto aplicará *multi-objective particle swarm optimization,* um tipo de algoritmo evolutivo, para o desenvolvimento de *hardware*. Tal algoritmo, implementado pelo autor da pesquisa e testado em vários *benchmarks*, utiliza o conceito de pareto anteriormente mencionado.

No primeiro momento, a pesquisa concentrará em desenvolver circuitos lógicos combinacionais clássicos de baixa complexidade, como um somador completo, multiplicador até mesmo uma ULA(Unidade Lógica Aritmética). O intuito do projeto, então, será aumentar a dificuldade paulatinamente, no aspecto da complexidade do circuito, no tamanho e na quantidade de objetivos.

No quesito objetivo, *a priori* o objetivo será apenas desenvolver um hardware que exiba um comportamento desejado, ou seja, seu comportamento tem que ser de acordo com aquele especificado anteriormente no projeto. Depois, vai adicionando outros objetivos, como consumo de energia, velocidade, nível de paralelismo e custo de produção.

No quesito complexidade, no início, como já foi dito, o objetivo será desenvolver circuitos combinacionais clássicos, depois aumentar sua escala, podendo adicionar registradores, memória até chegar no nível de desenvolver um circuito sequencial ou seja, um processador completo.

A análise dos resultados se baseará na efetividade das solução de pareto, ou seja, se o comportamento do dispositivo é realmente aquele esperado e na diversidade das soluções.

4. Plano de Trabalho

Mês	Tarefas
Agosto/2015 - Janeiro/2016	Aquisição de Conhecimento
Janeiro - Março	Aplicação do MOPSO em desenvolvimento de circuitos combinacionais clássicos.
Março - Abril	Aplicação do MOPSO em desenvolvimento de circuitos combinacionais mais complexos e sobre mais de um objetivo.
Abril - Maio	Aplicação do MOPSO em desenvolvimento de circuitos sequenciais.
Maio - Junho	Entrega do relatório final e desenvolvimento de experimentos complementares.

5. Bibliografia

[1] David Corne.; Marco Dorigo.; Fred Glover. *New Ideas in Optimization* McGraw-Hill, London, UK, 1999.

[2] Bianchi, Leonora; Marco Dorigo; Luca Maria Gambardella; Walter J. Gutjahr. "A survey on metaheuristics for stochastic combinatorial optimization". *Natural Computing: an international journal* 8. 2009. pp. 239–287.

[3] Carlos A. Coello Coello. "*Twenty Years of Evolutionary Multi-Objective Optimization: A Historical View of the Field*".2005.

[4] "The Nature of Mathematical Programming," *Mathematical Programming Glossary*, INFORMS Computing Society.

[5] Carlos A. Coello Coello. "*Twenty Years of Evolutionary Multi-Objective Optimization: A Historical View of the Field*".2005.

[6] "Operational Research in the British Army 1939–1945, October 1947, Report C67/3/4/48, UK National Archives file WO291/1301 Quoted on the dust-jacket of: Morse, Philip M, and Kimball, George E, *Methods of Operations Research*, 1st Edition Revised, pub MIT Press & J Wiley, 5th printing, 1954.

[7] Simon, Dan. "*Evolutionary Optimization Algorithms: Biologically-Inspired and Population-Based Approaches to Computer Intelligence*".Wiley. 2013. pp 3.

[8] Darwin, Charles. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life* (1st ed.). London: John Murray. 1859.

[9] Mayr, Ernst . *What Evolution Is*. London: Phoenix. 2002.

[10] Huxley, Julian. *Evolution: The Modern Synthesis*. With a new foreword by Massimo Pigliucci and Gerd B. Müller. Cambridge, MA: MIT Press. 2010.

[11] K. Deb, A. Pratap, S. Agarwal T. Meyarivan. *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II*. IEEE Transactions on Evolutionary Computation, Vol 6, No. 2. 2002.

[12] C.A. Coello Coello and M.S. Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In Proceedings of the IEEE Congress on

Evolutionary Computation. volume 2. 2002. pp 1051–1056.

[13] Engelbrecht, Andries . *Computaional Intelligence: An Introduction* (2st ed.). Wiley. 2007.pp 127.

[14] Engelbrecht, Andries . *Computaional Intelligence: An Introduction* (2st ed.). Wiley. 2007. pp 128.

[15] V. Ramos, C. Fernandes.; A.C. Rosa. *Social Cognitive Maps, Swarm Collective Perception and Distributed Search on Dynamic Land-scapes.Technical report*, Instituto Superior Técnico, Lisboa, Portugal. 2005.

[16] J. Hoffmeyer. The Swarming Body. In I. Rauch and G.F. Carr, editors, Semi-otics Around the World, Proceedings of the Fifth Congress of the International Association for Semiotic Studies. 1994. pp. 937–940.

[17] J. Kennedy.; R.C. Eberhart. Particle Swarm Optimization. In Proceedings of the IEEE International Joint Conference on Neural Networks. 1995. pp. 1942–1948.

[18] Simon, Dan. Evolutionary Optimization Algorithm: Biologically Inspired and Population-Based Approaches to Computer Intelligence . Wiley. 2007. pp 518.

[19] Simon, Dan. Evolutionary Optimization Algorithm: Biologically Inspired and Population-Based Approaches to Computer Intelligence. Wileu. 2007. pp. 517.

[20] Ehrgott, Matthias. Multi-Criteria Optimization. 2005. Springer.

[21] Deb, K. Multi-objective optimization using evolutionary algorithms. Chichester, UK: Wiley. 2001. p. 516.

[22] Higuchi, T., Niwa, T., Tanaka, T., Iba, H., de Garis, H., and Furuya, T., “Evolving Hardware with Genetic Learning: A First Step Towards Building a Darwin Machine,” in [Proc. of the 2nd International Conference on Simulated Adaptive Behaviour]. MIT Press. 1993. pp. 417–424

[23] de Garis, H., “Evolvable Hardware – Genetic Programming of a Darwin Machine,” in [International Con-ference on Artificial Neural Networks and Genetic Algorithms ICANNGA’93]. 1993.

[24] Miller F., Julian. Cartesian Genetic Programming (Natural Computing Series). Springer. 2011. p 346.

[25] Zebulum Salem, Ricardo. Pacheco C. Aurélio, Marco. Vellasco B.R. Maria, Marley. Evolutionary Eletronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms. CRC. 2002.

[26] Gordon, G. W.; Timothy. Bentley, J. Peter. “On Envolvable Harware”. EH '02 Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware. 2002. pp. 1.

[27] Gordon, G. W.; Timothy. Bentley, J. Peter. “On Envolvable Harware”. EH '02 Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware. 2002. pp. 4.