INTEGRAÇÃO DE PLATAFORMAS COMPUTACIONAIS PARA A SOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO EM ENGENHARIA

Tales Lima Fonseca

Afonso Celso de Castro Lemonge

Heder Soares Bernardino

taleslimaf@gmail.com

afonso.lemonge@ufjf.edu.br

hedersb@gmail.com

Curso de Engenharia Computacional, Universidade Federal de Juiz de Fora

36036-900, Juiz de Fora, Minas Gerais, Brasil

Resumo. *Problemas de otimização são comuns em diversas áreas. Na engenharia estrutural, por exemplo, estes estão sendo modelados considerando complexidades que, até então, eram descartadas. Isso, com o intuito de simplificar as simulações durante o processo de busca das soluções ótimas. Os softwares comerciais, boa parte deles, consolidados no mercado passam a ser alternativas atraentes nas análises dos problemas de otimização que demandam análises mais complexas. O uso dos mesmos evita a implementação de códigos computacionais que suportam modelagens complexas como as requeridas em vários problemas de otimização estrutural, por exemplo. Pretende-se neste trabalho a apresentação de um estudo preliminar da integração de plataformas computacionais para a solução de problemas de otimização estrutural utilizando o softwares comercial Abaqus como ferramenta de simulação computacional e a linguagem de programação Python. Um algoritmo genético implementado é usado como ferramenta de busca. Os experimentos iniciais são realizados em problemas de otimização estrutural comumente encontrados na literatura mostrando a integração desejada. Com isso, espera-se estudar os problemas de otimização estrutural de maior complexidade, que é o objetivo principal deste trabalho.*

Palavras-chave: *Otimização Estrutural, Algoritmos Genéticos, Python, Abaqus*

1 INTRODUÇÃO

Engenheiros estruturais normalmente buscam criar projetos com desempenho máximo e/ou um custo mínimo. Estes requisitos podem ser alcançados modelando-se os projetos através de problemas de otimização. A procura por projetos otimizados se torna desejável em virtude da possibilidade de minimização dos gastos com materiais sem que, com isso, haja perda de sua segurança. Além disso, mais recentemente tem crescido o interesse na diminuição do uso excessivo de matérias primas cada vez mais escassas. Finalmente, é importante que a estrutura projetada possua um bom desempenho durante toda a sua vida útil, independente do seu grau de complexidade, o que torna altamente recomendado a busca por técnicas robustas de otimização. Uma das técnicas que se mostrou capaz de resolver problemas de otimização estrutural obtendo bons resultados foram os algoritmos genéticos (Deb, 1991).

Em meio aos problemas encontrados na otimização estrutural, existem os que apresentam uma grande complexidade para se alcançar boas soluções. Tal complexidade pode acabar desinteressando o desenvolvimento, a implementação e o uso de plataformas computacionais domésticas, de modo que plataformas robustas (muitas vezes comerciais) são requeridas. Contudo, nem sempre os módulos de otimização desejados estão disponíveis nestas plataformas, o que não impede o seu uso na avaliações de projetos candidatos.

A integração de plataformas computacionais para a solução de problemas de otimização estrutural utilizando o *software* comercial Abaqus® combinado com e a linguagem de programação Python é investida aqui. Um algoritmo genético foi implementado em Python enquanto que as avaliações estruturais necessárias para a busca são realizadas via Abaqus®.

2 ALGORITMOS GENÉTICOS

Os algoritmos genéticos (AGs) são técnicas de busca estocásticas inspirados pela teoria da evolução das espécies (Holland, 1975) e têm sido largamente utilizados para resolver os problemas da engenharia estrutural. Os AGs não requerem continuidade ou diferenciabilidade da função objetivo, bem como um ponto de partida factível. Também, não possuem um domínio de aplicação restrito e podem resolver problemas sobre variáveis de projeto mistas. Como pode ser visto no Algoritmo 1, que apresenta um pseudocódigo de um AG, avaliações das soluções candidatas são necessárias a cada passo. Essa avaliação depende, no caso abordado aqui, de (i) um simulador (Abaqus®) e (ii) uma técnica de penalização (Seção 3).

Algoritmo 1: Pseudocódigo AG

```
t = 1;
inicializa( Pop(t=1) );
avalia( Pop(t=1) );
repita
    selPop(t) = seleção(Pop(t));
    novaPop(t) = reprodução(selPop(t));
    mutação(novaPop(t));
    avalia(novaPop(t));
    Pop(t+1) = novaGeração(novaPop(t),Pop(t));
    t=t+1;
até t = NGEN;
```

3 UMA TÉCNICA DE PENALIZAÇÃO ADAPTATIVA

O projeto de otimização estrutural tratado aqui visa a minimização do peso da estrutura (funções objetivo) sujeito às restrições de deslocamento dos nós e tensões das barras (restrições). Os AGs foram idealizados para resolver problemas de otimização sem restrições, de modo que é preciso incorporar técnicas para tratá-las. Uma classe de técnicas bastante empregadas para tratamento de restrições são as funções de penalização, em que os problemas com restrições são transformados em problemas irrestritos através da introdução de uma função de penalização. O ajuste de parâmetros da penalização estática dificulta sua utilização e, assim, sugere-se o uso de penalizações dinâmicas ou adaptativas.

Em (Barbosa and Lemonge, 2002), foi proposto um método de penalização adaptativa denominado APM (*Adaptive Penalty Method*). O APM adapta o valor dos coeficientes de penalização de cada restrição usando informações da população em cada geração. A função de aptidão é definida como

$$F(\vec{x}) = \begin{cases} f(\vec{x}), & \vec{x} \quad \text{factível} \\ \bar{f}(\vec{x}) + \sum_{j=1}^{m} k_j v_j(\vec{x}), & \text{caso contrário} \end{cases} \qquad \bar{f}(x) = \begin{cases} f(\vec{x}), & f(\vec{x}) > \left\langle f(\vec{x}) \right\rangle \\ \left\langle f(\vec{x}) \right\rangle, & f(\vec{x}) \leq \left\langle f(\vec{x}) \right\rangle \end{cases}$$

onde $\left\langle f(\vec{x}) \right\rangle$ é a média dos valores da função objetivo da população atual e k_j é definido como

$$k_j = |\left\langle f(\vec{x}) \right\rangle| \frac{\left\langle v_j(\vec{x}) \right\rangle}{\sum_{l=1}^{m} [\left\langle v_l(\vec{x}) \right\rangle]^2}$$

onde $\left\langle v_j(\vec{x}) \right\rangle$ é a média da violação $v_j(\vec{x})$.

4 INTEGRAÇÃO DAS PLATAFORMAS

O desenvolvimento de qualquer modelo utilizando o Abaqus® pode ser feito através de sua interface gráfica ou utilizando suas bibliotecas, implementadas em Python. Apesar dessas bibliotecas possuírem métodos que possibilitam a implementação de um modelo, seu uso direto é desmotivador, dado o grande trabalho de desenvolvimento requerido. Uma forma prática é utilizar a interface gráfica para construção de um modelo base que pode ser salvo como um *script* Python, podendo ser incorporado a outros códigos. Assim, é possível criar um modelo mais facilmente na interface gráfica, gerar um código em Python e incorporá-lo ao próprio código-fonte, variando apenas alguns de seus parâmetros.

As soluções candidatas do AG são passadas por parâmetro para o *script*, possibilitando assim a análise de diferentes soluções. Após a execução do *script* para uma determinada solução candidata, o Abaqus® escreve um arquivo no disco com extensão ".odb"(*output database*) contendo os dados de saída da simulação. Os dados deste arquivo de saída podem então ser lidos e seus valores utilizados para avaliar um projeto estrutural candidato.

5 EXPERIMENTO NUMÉRICO

Para validar a integração proposta aqui, foi realizado um experimento numérico com um exemplo clássico de otimização estrutural: a treliça de 10 barras (ilustrada na figura 1).

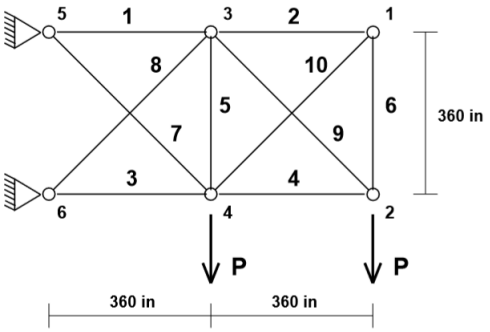

Figura 1: Treliça de 10 Barras

O problema de otimização da treliça de 10 barras consiste em encontrar o conjunto de áreas $A = \{A_1, \ldots, A_{10}\}$ que minimiza o peso da estrutura, ou seja:

$$W(A) = \sum_{i=1}^{10} \rho A_i L_i$$

sujeito às restrições de deslocamentos dos nós e de tensões das barras, respectivamente,

$$\frac{|u_j|}{u_{max}} - 1 \leq 0, \quad \forall\, j = 1, 2, \ldots, m \qquad \frac{\sigma_i}{\sigma_{max}} - 1 \leq 0, \quad \forall\, i = 1, 2, \ldots, n = 10$$

onde ρ é a densidade do material, L_i é o comprimento do i-ésimo membro da treliça, m é o número de graus de liberdade da treliça, n é o número de barras, $u_{max} = 25000.0$psi, $\sigma_{max} = 2.0$in, $A_i \in [0.1; 40.0]$in^2. Além disso, foram adotados: módulo de elasticidade igual a 10^7psi, massa específica igual a 0.1lb/in^3 e cargas concentradas com 100Kips.

Foi utilizado um AG geracional com codificação real, recombinação SBX e mutação Polinomial (Deb and Agrawal, 1995). Os valores adotados para a probabilidade de recombinação e mutação são, respectivamente, 80% e 5%, e as taxas são ambas iguais a 50%. Além disso, foi realizado um processo de elitismo na população, selecionando 5% dos melhores indivíduos para a geração seguinte e uma seleção de 80% dos melhores indivíduos para *crossover*. De forma a completar a quantidade de indivíduos para a geração seguinte, é realizado uma inserção de novos indivíduos gerados aleatoriamente dentro do espaço de busca. Finalmente, a população foi formada por 100 indivíduos, o processo evolutivo iterou por 200 gerações.

A Figura 2 apresenta um gráfico da convergência da melhor solução em cada geração, que possui peso final igual a 5159.9009 lb. Suas variáveis de projeto são apresentadas na Tabela 1. Apesar do orçamento computacional reduzido, a integração computacional obteve resultado satisfatório, convergindo para uma solução próxima da melhor solução conhecida, de peso igual à 5060.8544 lb (Silva et al., 2008). Apesar de viabilizar a otimização em projetos complexos, a utilização do Abaqus® requer mais tempo de processamento, pois as trocas de dados entre processo de busca e simulador são realizados via escrita/leitura de arquivos em disco.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi proposto a integração do *software* Abaqus® com um Algoritmo Genético (AG) para a resolução de problemas de otimização estrutural. O tratamento das restrições do problema foi feito via o acoplamento de um método de penalização adaptativa (APM) ao AG.

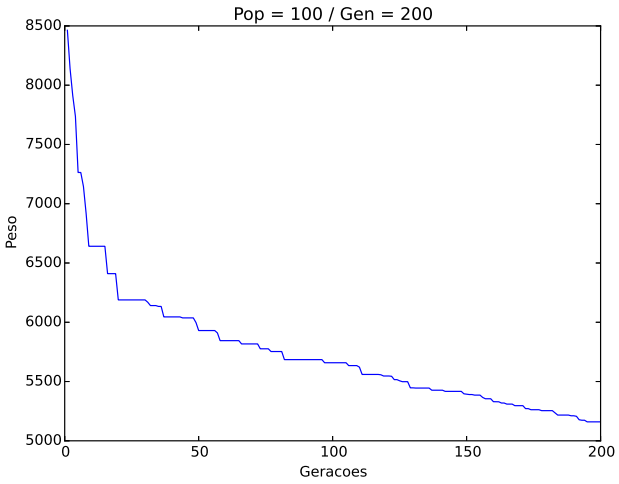

Figura 2: População com 100 Indivíduos em 200 Gerações

Tabela 1: Comparação de Resultado: Variáveis de Projeto e Pesos Finais

—	naval	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	Peso (lb)
Artigo	20000	31.7064	0.1946	27.0424	16.0071	0.1060	0.3731	7.9972	18.6270	21.0473	0.3417	5159.9009
Ref.	280000	30.5208	0.1000	23.1962	15.2335	0.1000	0.5494	7.4579	21.0399	21.5213	0.1000	5060.8544

Para avaliar a viabilidade da integração, foi utilizado um problema clássico de otimização estrutural. O resultado apresentado valida a integração entre as plataformas. O algoritmo obteve um resultado satisfatório, apesar da solução ainda poder ser melhorada (ao comparar o resultado com outros disponíveis na literatura).

Como trabalho futuro sugere-se a implementação e validação da integração proposta aqui na resolução de outros problemas, bem como a melhora do método de busca.

Agradecimentos Os autores agradecem os apoios do PGMC/UFJF, CNPq (305099/2014-0) e FAPEMIG (TEC PPM 528/11 e TEC PPM 388/14).

REFERÊNCIAS

Barbosa, H. J. C. and Lemonge, A. C. C. (2002). An adaptive penalty scheme in genetic algorithms for constrained optimiazation problems. pages 287–294. ACM.

Deb, K. (1991). Optimal design of a welded beam via genetic algorithms. *AIAA Journal*, pages 2013–2015.

Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, pages 115–148.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

Silva, E. K. d., Barbosa, H. J. C., and Lemonge, A. C. C. (2008). An adaptive constraint handling technique for differential evolution in engineering optimization. In *International Conference on Engineering Optimization*, page 46.