Felipe Carraro

Otimização estrutural de pórticos planos utilizando o algoritmo *SGA*

Florianópolis

2015

Felipe Carraro

Otimização estrutural de pórticos planos utilizando o algoritmo *SGA*

Trabalho de conclusão de curso submetido ao Departamento de Engenharia Civil da Universidade Federal de Santa Catarina para a obtenção do título de Engenheiro Civil

Universidade Federal de Santa Catarina - UFSC

Centro Tecnológico

Curso de Engenharia Civil

Orientador: Rafael Holdorf Lopez

Florianópolis

2015

Felipe Carraro
Otimização estrutural de pórticos planos utilizando o algoritmo *SGA*/ Felipe Carraro. – Florianópolis, 2015-
94 p. : il. (algumas color.) ; 30 cm.

Orientador: Rafael Holdorf Lopez

Trabalho de Conclusão de Curso – Universidade Federal de Santa Catarina - UFSC
Centro Tecnológico
Curso de Engenharia Civil, 2015.

1. Engenharia Civil. 2. Otimização estrutural. 3. Pórticos. 4. *SGA*. I. Lopez, Rafael Holdorf. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Civil. III. Título

Felipe Carraro

Otimização estrutural de pórticos planos utilizando o algoritmo *SGA*

Trabalho de conclusão de curso submetido ao Departamento de Engenharia Civil da Universidade Federal de Santa Catarina para a obtenção do título de Engenheiro Civil

Prof. Rafael Holdorf Lopez
Orientador

Prof. Leandro Fleck Fadel Miguel
Universidade Federal de Santa Catarina

André Gustavo Carlon
Universidade Federal de Santa Catarina

Florianópolis
2015

Este trabalho é dedicado à todos que,
direta ou indiretamente, me ajudaram a completar
mais esta etapa da jornada.

Agradecimentos

Agradeço primeiramente a Deus, por me guiar, dar força e se mostrar presente nas horas mais necessárias.

Aos meus pais Fernando e Susana, que sempre me incentivaram a estudar e buscar meus objetivos.

À minha namorada Graziele, pelo apoio, motivação e compreensão nas minhas ausências.

Ao meu orientador Rafael, pela disponibilidade de sempre esclarecer meus questionamentos e por me instruir de modo a realizar um bom trabalho.

Ao meu amigo Matheus, pela criação do algoritmo que deu origem a este trabalho, bem como o esclarecimento de qualquer questão relacionada ao mesmo.

“Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

Resumo

Neste trabalho são abordados os conceitos de otimização de forma prática, aplicando-os à estruturas de pórtico plano, utilizando-se como material o aço. Será utilizado o algoritmo *SGA* (*Search Group Algorithm*), aplicando-o em três problemas de pórticos planos já estudados na literatura. Serão comparados os resultados obtidos com os resultados da literatura, verificando o desempenho do mesmo na solução deste tipo de problema não avaliado até então. Obteve-se um bom desempenho do algoritmo para todos os problemas estudados. No primeiro exemplo foi possível reproduzir o valor ótimo global com um número reduzido de iterações frente aos outros autores. Já no terceiro, que seria o mais complexo dos três estudados, o algoritmo conseguiu o melhor resultado em comparação direta para um número semelhante de avaliações em relação à literatura. No segundo problema, obteve-se um bom resultado, não sendo no entanto o melhor encontrado. Obteve-se o segundo melhor resultado dentre as comparações possíveis, com um número menor de avaliações da função (6.000) em relação ao melhor resultado da literatura (8.300). De todo modo, os resultados foram satisfatórios, de modo que o *SGA* se colocou como o melhor ou entre os melhores analisados, superando alguns autores utilizando algoritmos já consagrados. Desta maneira, classifica-se como competitiva a rotina utilizada para a busca de soluções.

Palavras-chaves: algoritmo de otimização, otimização estrutural, pórtico, *SGA*, aço.

Abstract

In this document, optimization conecpts are adressed in a practical way, aplying them to plane frame steel structures. It will be used the *SGA* (*Search Group Algorithm*), developed inside this university, aplying it on three plane frame problems already studied in the literature. The results obtained will be compared with the results from the literatura, verifying the algorithm performance on the solution of this kind of problem, unexplored until now. As a result, a good performance was obtained on all problems studied. On the first example, it was possible to reproduce the global optimum with a reduced number of iterations compared to the other authors. On the third example, which was the most complex of the three, the algorithm managed to obtain the best result where direct comparatives were avaiable, in a similar number of evaluations compared to the literature. On the second problem, a mostly good result was obtained, although a better result was found on the literature. It was the second best result where comparatives were possible, with a reasonable number of evaluations (6.000) against the author with the best result (8.300). The results obtained were satisfactory, as *SGA* achieved some of the best results with a performance better than some authors using already established algorithms. This way, the optimal results searching routine can be classified as competitive.

Key-words: optimization algorithm. structural optimization. frame. *SGA*. steel.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Apresentação

No desenvolvimento de projetos de engenharia, muitas vezes se observa um processo de tentativa e erro em busca de um projeto adequado. Existem diversas variáveis e restrições que se inter-relacionam, formando um grande grupo de soluções possíveis. Em termos de projetos estruturais, a ideia geral se baseia em um ciclo de pré-dimensionamento do problema, seguido de uma análise da solução proposta e da correção de eventuais inadequações da primeira proposição.

Com a popularização dos computadores, tornou-se mais fácil e viável realizar simulações computacionais de estruturas, de modo a avaliar a resposta de uma dada estrutura mediante a uma ou mais combinações de carregamento, geometria, seção transversal, etc.

No caso dos pórticos de aço, que serão os objetos de estudo deste trabalho, durante o processo de concepção do projeto, torna-se necessário encontrar perfis com propriedades que atendam a resistência requerida para suportar os esforços externos e cujos deslocamentos sejam tais que atendam as prescrições normativas.

Para automatizar e acelerar este ciclo surge a proposta de otimização estrutural, onde busca-se a melhor solução possível para o problema, dadas as variáveis e restrições do projeto. Para isso são utilizados algoritmos de otimização, que usam as informações de cada análise para buscar e aprimorar uma dada solução encontrada de modo iterativo.

1.2 Motivação

A motivação principal deste trabalho é a possibilidade da obtenção de conhecimentos aprofundados acerca do tema otimização. Tal assunto, apesar de ter grande aplicação prática na engenharia, como será visto na sequência, além da aplicação a muitas outras áreas do conhecimento, não é abordado durante a graduação.

A intenção do trabalho é dar continuidade à alguns outros estudos relacionados ao mesmo assunto já apresentados anteriormente por alunos da graduação de Engenharia Civil. Estudos anteriores, como Carlon (2013), Barbaresco (2014), e Ribeiro (2014), tratavam da otimização geral ou de estruturas treliçadas. Este trabalho visa dar um passo adiante, aplicando os conceitos de otimização à estruturas de pórtico.

1.3 Objetivo

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é aplicar o método de otimização "algoritmo do grupo de busca" no projeto ótimo de pórticos planos de aço.

1.3.2 Objetivos Específicos

Como objetivos específicos, cita-se:

(i) a familiarização inicial com as características dos problemas de otimização e suas particularidades;

(ii) a elaboração de rotinas computacionais de modo a verificar os elementos segundo a norma de aço americana;

(iii) a adaptação do algoritmo *SGA* à este tipo de problema;

(iv) a avaliação de problemas da literatura;

(v) a comparação dos resultados obtidos com os resultados da literatura.

2 Otimização

2.1 Elementos da otimização

O uso prático da otimização se dá primeiramente através da definição de pelo menos um objetivo, que representa a medida do desempenho de um sistema a ser analisado. Este objetivo depende de certas características do sistema, que são chamadas variáveis de projeto. A meta da otimização é encontrar valores para estas variáveis a fim de se obter o melhor valor para o objetivo. Algumas vezes estas variáveis estão limitadas à certos valores, situação que se caracteriza por um problema com restrições.

Deste modo, são três os elementos básicos de qualquer problema de otimização:

- Objetivo: esta é a função que associa os parâmetros do sistema analisado e mede um certo valor de desempenho. Ex.: $f_{obj} : \mathbb{R}^n \to \mathbb{R}$.
- Variáveis de projeto: São os parâmetros que permitem ao projetista modificar o sistema em questão de modo a melhorar o seu desempenho. Ex.: $\mathbf{x} = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}$.
- Espaço de busca: é o espaço contendo todas as possíveis variáveis de projeto limitadas pelas restrições impostas.

Assim, pode-se descrever um problema de otimização da seguinte maneira:

Encontrar um vetor x:

$$\mathbf{x} = \{x_1, x_2, \ldots, x_n\} \tag{2.1}$$

que minimize a função:

$$f_{obj} = f(\mathbf{x}) \tag{2.2}$$

sujeito às restrições:

$$g_i(\mathbf{x}) \leq 0; \quad i = 1, \ldots, n_{ri} \tag{2.3}$$

$$h_j(\mathbf{x}) = 0; \quad j = 1, \ldots, n_{rd} \tag{2.4}$$

onde n_{ri} é o número de restrições de igualdade e n_{rd} é o número de restrições de desigualdade.

2.2 Escolha do algoritmo

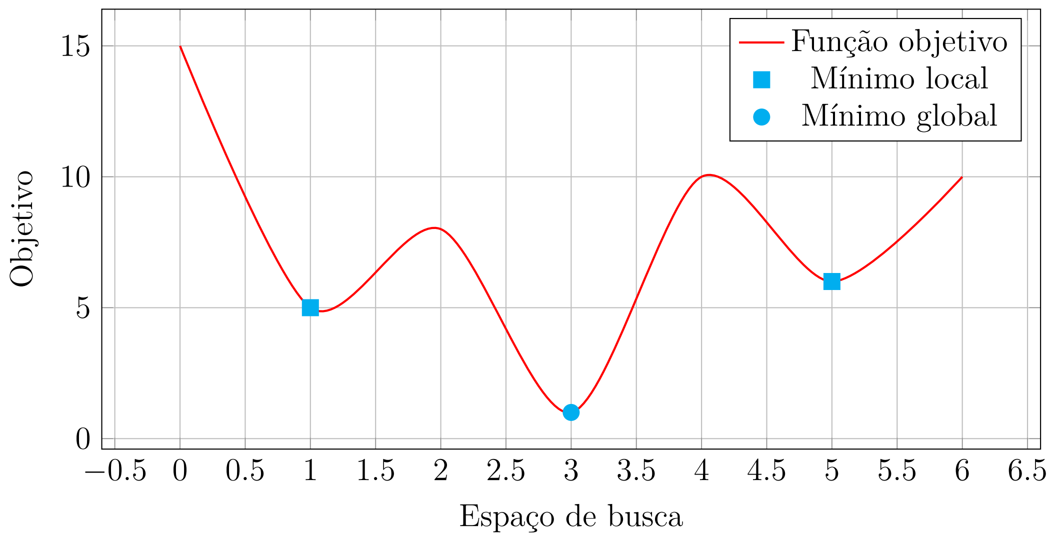
Para a solução de problemas de otimização, podem ser empregados diversos métodos ou algoritmos, dependendo das características do problema estudado.

Algoritmos de otimização clássicos geralmente necessitam de mais informações da função objetivo, como a derivada da função. Problemas de engenharia, como os abordados neste trabalho, são geralmente complexos. Nestes problemas, podem existir uma alta não-linearidade ou pontos que não são diferenciáveis, justificando assim o uso de algoritmos metaheurísticos. Estes, por sua vez, não necessitam de nenhuma informação sobre a função, buscando melhorar o resultado até então obtido iterativamente a partir da informações do desempenho do sistema em interações anteriores.

Além disso, tem-se que algoritmos clássicos apresentam geralmente um funcionamento determinístico, ou seja, seguem a mesma sequência de passos de modo a encontrar uma solução e fornecem sempre o mesmo resultado para um dada entrada. Isto pode representar uma dificuldade, já que o método empregado pode não encontrar o valor ótimo global, ficando restringido a uma solução local, ou seja, a melhor solução encontrada dentro de uma certa vizinhança, como ilustra a Figura 1. Problemas mais complexos tendem a ter diversos mínimos locais, o que prejudica os resultados obtidos por esta classe de métodos.

Algoritmos metaheurísticos, no entanto, apresentam um caráter estocástico, de modo que possuem aleatoriedade em sua formulação. Esta aleatoriedade faz com que os resultados apresentem variações a cada execução do mesmo, mas contribuem para dar condições ao algoritmo de explorar o domínio, escapando de mínimos locais e encontrando o ótimo global ou uma boa solução próximo a ele.

Figura 1: Mínimos locais e globais - Exemplo



Fonte: Autor

3 Formulação do problema de pórtico

É possível reescrever a descrição genérica de um problema de otimização, vista na seção 2.1, para o caso do presente estudo. A formulação geral de um problema de otimização de pórticos planos metálicos baseia-se em encontrar um conjunto de seções transversais a serem utilizadas no projeto, de maneira que se atendam as restrições e de modo que o projeto apresente a configuração mais econômica possível. Em relação à esta economia, estabelece-se em geral como objetivo minimizar o peso da estrutura.

Dado que se tenha organizado os N_m membros (vigas e pilares) do projeto em N_g grupos utilizando o mesmo perfil metálico, tem-se então um problema de programação discreta, onde o objetivo é encontrar um vetor de números inteiros $\mathbf{I}$ (Equação 3.1), que corresponde aos índices de uma lista padrão de perfis, onde cada índice indica um perfil utilizado no projeto para cada um dos N_g grupos

$$\mathbf{I}^T = [I_1, I_2, I_3, \ldots, I_g] \tag{3.1}$$

de forma que seja minimizado o peso W da estrutura:

$$W = \rho \sum_{i=1}^{N_g} A_i \sum_{j=1}^{N_t} L_j + P \tag{3.2}$$

onde ρ é o peso unitário do material, A_i é a área da seção transversal do perfil adotado para o grupo i, N_t é o número total de membros do grupo i e L_j é comprimento do membro j pertencente ao grupo i.

A parcela adicional P, refere-se ao tratamento das restrições do problema. Para os problemas estudados, as restrições serão as de esforços na estrutura, relativos à resistência do perfil com base nos critérios da norma americana de estruturas de aço AISC (2010), e eventuais restrições ao deslocamento máximo da estrutura. O uso da norma americana ao invés da brasileira ABNT (2008) para a verificação das restrições se dá de forma a permitir a comparação dos resultados com outros autores da literatura.

O recurso utilizado para lidar com as restrições foi a do uso de fatores de penalidade. Nestes casos, adiciona-se um peso extra à estrutura correspondente ao quão longe se está de atender a todos os critérios. Em geral, formulam-se as restrições de modo que se obtenha um valor igual à zero no caso das restrições estarem sendo atendidas. Assim, define-se o peso extra devido às restrições não atendidas da seguinte forma:

$$P = \alpha \sum_{i=1}^{n_r} g_i \tag{3.3}$$

onde α é o fator de penalização, e g_i refere-se a i-ésima restrição.

Para o parâmetro α utilizou-se valores bastante elevados (10^9-10^{10}), de modo a excluir da busca qualquer perfil que não atenda as restrições, uma técnica conhecida na literatura como "*Death Penalty*" (Pena de morte). (MEZURA-MONTES; COELLO, 2011)

Foi estudada também a possibilidade de usar uma outra técnica para lidar com as restrições, proposta por Deb (2000), baseada na elegibilidade das soluções, onde as soluções são confrontadas em um torneio binário que segue três simples regras:

- Regra 1: prefere-se soluções elegíveis à não-elegíveis;
- Regra 2: entre duas soluções elegíveis prefere-se a com menor função objetivo;
- Regra 3: entre duas soluções não-elegíveis prefere-se a com menor valor de violação das restrições.

Como dito anteriormente, a técnica utilizada para lidar com restrições foi o uso da penalização ("*Death Penalty*"), pois apesar da técnica acima ser mais refinada, seu uso estava mais associado à otimização de algoritmos genéticos, de modo que não foram obtidos melhores resultados em relação à técnica anterior.

4 Apresentação do algoritmo

4.1 Introdução

O *SGA* (*Search Group Algorithm* ou "Algoritmo do grupo de busca") é um algoritmo metaheurístico de otimização global, capaz de se adequar às particularidades de cada problema por meio da alteração de parâmetros de aleatoriedade e amplitude de busca. O mesmo foi desenvolvido pelo aluno Matheus Silva Gonçalves, da Universidade Federal de Santa Catarina. Sua eficácia para a otimização de treliças foi verificada em um recente artigo publicado. (GONÇALVES et al., 2015)

Nesta seção serão descritos as características do algoritmo, bem como os processos realizados pelo mesmo de forma a minimizar uma dada função objetivo.

Em sua implementação original, o algoritmo avaliava as variáveis de forma contínua, de modo que foi realizada uma adaptação no mesmo. Mudanças foram realizadas nas funções de geração das famílias e da população inicial, a fim de que a rotina gerasse e analisasse somente variáveis discretas inteiras, que condizem a com a formulação do problema de pórtico, reduzindo assim o custo computacional e o número de iterações necessárias pra este tipo de problema. A rotina computacional do algoritmo está disponível para averiguação no Apêndice B.

4.2 População inicial

O início do processo de otimização se dá a partir da geração de uma população inicial, correspondente a uma das possíveis configurações do problema analisado, através de pontos aleatórios do domínio da função objetivo. Estes pontos são então avaliados através da função objetivo, e suas coordenadas e avaliação da função são usadas para o início da formação dos grupos de busca.

4.3 Geração do grupo de busca

A rotina busca a otimização do problema a partir da criação de um grupo de otimização, ou grupo de busca (*search group*). O tamanho do grupo de otimização é definido como uma porcentagem da população total.

O grupo principal é formado pela junção de dois subgrupos: grupo de elite e grupo de torneio. O grupo de elite se refere a um certo número de indivíduos, que é selecionado diretamente para o grupo de otimização em função de sua boa colocação ou seu *rank*, no que diz respeito à menores valores de avaliação da função objetivo.

As vagas remanescentes no grupo de otimização são preenchidas pelo segundo subgrupo, onde os indivíduos são selecionados por um processo de torneio. Neste torneio, toma-se aleatoriamente um certo número fixo de indivíduos e realiza-se a avaliação da função objetivo para cada um. O indivíduo mais bem avaliado de cada conjunto entra para o grupo de otimização.

4.4 Processo iterativo global

Definido o grupo de busca inicial, incia-se o processo iterativo de otimização. Como primeira etapa é realizado um processo de otimização global, no qual busca-se explorar o domínio o máximo possível. Neste processo cada individuo do grupo de otimização pode gerar um determinado número de outros indivíduos. O número de indivíduos gerados por cada membro do grupo é determinado pela sua qualificação. Membros mais bem avaliados podem gerar um número maior de indivíduos.

Em cada iteração da fase global do algoritmo há uma alteração na aleatoriedade. A aleatoriedade da busca é definida pelo parâmetro α do algoritmo, através da seguinte equação:

$$\alpha = (\alpha_0 \cdot \beta + \alpha_{min})(l_{sup} - l_{inf}) \tag{4.1}$$

O valor de α é modificado a cada iteração do algoritmo. Tem-se que α_0 é valor de aleatoriedade inicial, um parâmetro configurável para cada tipo de problema, β é um fator multiplicador que varia de 0 à 1, que promove o decaimento da aleatoriedade, α_{min} é um valor mínimo de aleatoriedade considerado de modo que não seja realizada uma iteração com um valor α igual à zero e l_{sup} e l_{inf} são os limites superior e inferior da variável analisada.

O fator de decaimento β é composto pelo maior valor da ordenada entre duas retas. Este valor decai com base no número de iterações já completadas. O valor de β indica a porcentagem da aleatoriedade inicial que será utilizada na próxima iteração.

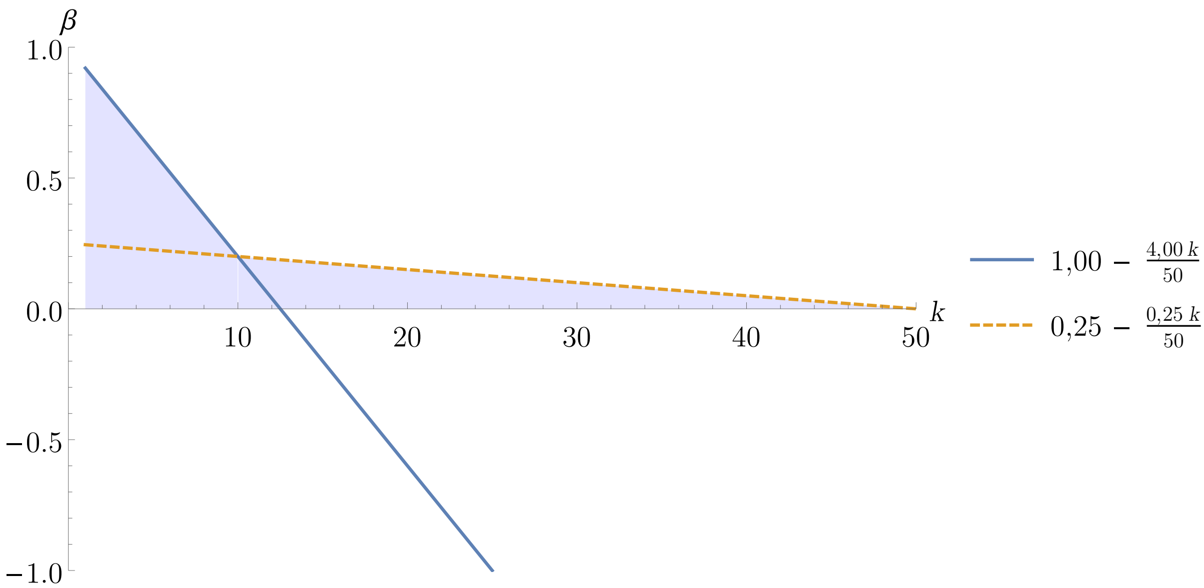
Sua concepção se deu de modo a considerar valores mais altos de aleatoriedade no início do algoritmo, decaindo rapidamente até se completarem 20% do número de iterações da fase global, para posteriormente, no restante das iterações, decair mais suavemente.

Este esquema de decaimento tem como objetivo obter uma alta aleatoriedade inicial, de modo a explorar mais amplamente o domínio. Conforme as regiões mais favoráveis para a localização de mínimos da função vão sendo definidas, ocorre uma redução gradual da aleatoriedade. O fator β segue a seguinte equação:

$$\beta = \max \left\{ \begin{matrix} 1,00 - \dfrac{4,00\ k}{n_{iterg}} \\ \\ 0,25 - \dfrac{0,25\ k}{n_{iterg}} \end{matrix} \right\} \tag{4.2}$$

onde k é iteração atual do algoritmo e n_{interg} é o numero total de iterações a serem realizadas na fase global.

Figura 2: Decaimento de β para 50 iterações globais



Fonte: Autor

Na Figura 2 fica exemplificado o decaimento de β a cada iteração k, para uma etapa global com 50 iterações. A área sombreada indica o trecho entre o eixo das abcissas e o valor máximo entre as duas retas, onde é possível verificar o decaimento acentuado da aleatoriedade até a décima iteração (20%), seguido de um decaimento mais suave ao longo das iterações restantes.

Definido o parâmetro α da iteração, cada indivíduo gera um certo número de filhos com base em seu rank, compondo uma família. A definição do número de indivíduos que cada membro pode gerar pode obedecer qualquer função arbitrada pelo usuário. Por padrão, o modelo utilizado baseia-se na ideia da divisão do grupo em duas metades. A primeira metade, composta pelos membros mais bem avaliados, gera dois terços do número total de novos indivíduos, enquanto que a segunda metade gera o terço restante. Ainda, estabelece-se como regra para a definição da função de geração de indivíduos, que o número total de indivíduos gerados seja igual à diferença entra a população total e o número de componentes dos grupos de otimização. Essa regra tem a função de manter constante o número de avaliações da função objetivo.

Uma análise matemática simples, mostra que para uma população inicial de n_{pop}, com uma porcentagem pn_g do total, formando um grupo de otimização de $n_g = n_{pop} \cdot pn_g$ indivíduos, tem-se que a soma do número x de indivíduos gerados por cada membro do primeiro subgrupo somada ao número y de indivíduos gerados por cada membro do segundo subgrupo é de:

$$
\begin{aligned}
\frac{n_g}{2}x + \frac{n_g}{2}y &= (n_{pop} - n_g) \\
\frac{n_g}{2}(x + y) &= (n_{pop} - n_g) \\
x + y &= \frac{2n_{pop} - 2n_g}{n_g} \\
x + y &= \frac{2n_{pop}}{n_g} - 2 \\
x + y &= \frac{2}{pn_g} - 2
\end{aligned} \tag{4.3}
$$

Utilizando um exemplo, dada uma população inicial de 80 indivíduos, com 20% deles compondo o grupo de otimização, obtém-se a relação $x + y = 8$. Logo, utilizando-se aproximadamente a relação 2/3 para 1/3 já citada acima, metade do grupo de otimização (8 indivíduos) gerarão 5 indivíduos cada, enquanto que a outra metade gerará 3 indivíduos cada, compondo os 64 indivíduos gerados.

Cada membro do grupo de busca gera descendentes compondo uma família. Estes descendentes são gerados com base no valor da aleatoriedade α da iteração corrente. Atribui-se para cada coordenada x do indivíduo uma valor aleatório dentro do intervalo $(x - \frac{\alpha}{2}, x + \frac{\alpha}{2})$, ou seja, dentro de uma amplitude α de variação.

Cada família é então avaliada separadamente, sendo escolhido para o grupo de busca da próxima iteração, o membro mais bem avaliado. Ordena-se o grupo de busca de modo a manter sempre na primeira posição o melhor indivíduo encontrado até então.

4.5 Processo iterativo local

Na etapa local do processo iterativo, entende-se que o algoritmo já explorou boa parte do domínio, estabelecendo-se em regiões favoráveis para a localização de mínimos. Nesta etapa busca-se refinar os resultados encontrados na etapa global, buscando o valor ótimo da função objetivo.

O processo de otimização local, diferencia-se do global no que tange a geração dos novos grupos. Ao passo que na etapa global cada família era analisada individualmente, neste processo, após a geração das famílias, os indivíduos são avaliados como um todo, passando para o novo grupo, os membros com melhor avaliação, independente da família que os gerou.

A aleatoriedade também é diferenciada, utilizando-se somente uma reta de decaimento. O multiplicador β fica definido como:

$$\beta = \frac{n_{iterl} - k}{n_{iterl}} \tag{4.4}$$

onde n_{iterl} é o número de iterações da etapa local e k é iteração corrente.

O valor de α fica então definido como:

$$\alpha = (\beta\ \alpha_{min} + r_{min}\ \alpha_{min})(l_{sup} - l_{inf}) \tag{4.5}$$

onde r_{min} corresponde ao residual de aleatoriedade, definido como uma porcentagem de α_{min}.

Verifica-se que α decresce a partir do valor de α_{min} produzido no fim da etapa global, sendo reduzido linearmente com o passar das iterações, acrescido de um pequeno valor residual de modo a não se obter aleatoriedade nula.

4.6 Mutação dos grupos de busca

No início de cada iteração, antes de se iniciar a criação das famílias, ocorre um processo de mutação nos membros do grupo de busca. Esta mutação se dá através da substituição de membros do grupo por novos membros gerados a partir da média e desvio padrão do grupo atual. A ideia principal é explorar novas regiões do domínio, fora da amplitude α definida para a geração de descendentes.

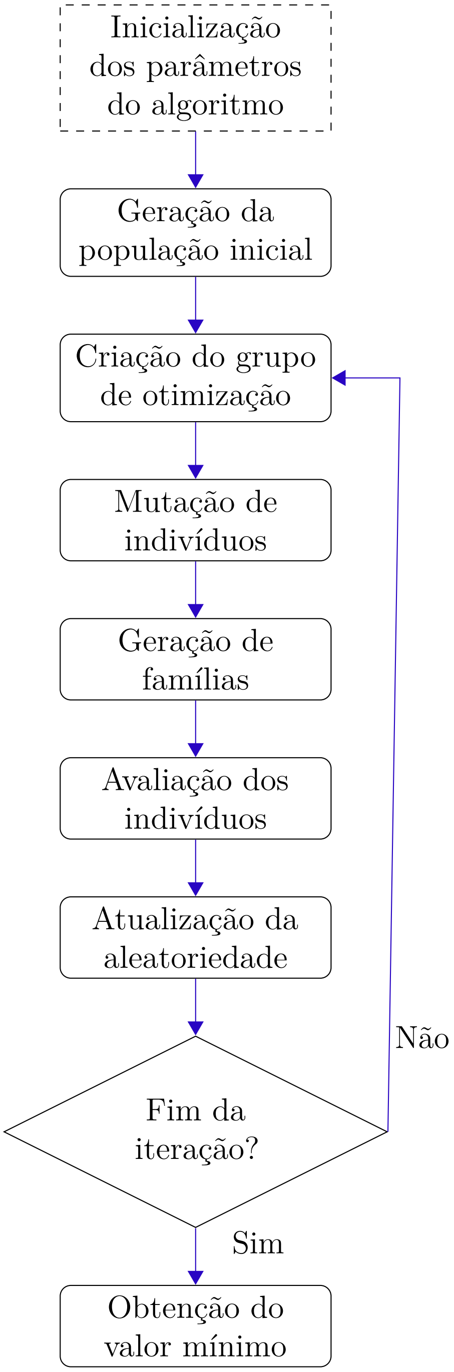
Para a substituição de indivíduos é utilizado um processo de torneio inverso. Semelhante ao processo de torneio comentado na seção 4.3, neste processo escolhe-se aleatoriamente subgrupos de tamanho fixo, sendo estes individualmente analisados. No caso do torneio inverso, dentro de cada subgrupo, "vence" o torneio o indivíduo com a pior qualificação, sendo este o escolhido para ser substituído pelo membro proveniente da mutação. A mutação é definida para cada variável x pela seguinte equação:

$$\text{mutação} = \bar{x} + t\ \epsilon\ \sigma \tag{4.6}$$

onde $\bar{x}$ e σ são respectivamente a média e o desvio padrão da coordenada no grupo de otimização, ϵ é uma variável aleatória com intervalo de -0,5 até 0,5 e t varia de 1 até o número de substituições que serão realizadas, indicando a amplitude de geração. O número de mutações que serão adicionadas no grupo de busca é um dos parâmetros configuráveis do algoritmo.

4.7 Resumo do processo

Figura 3: Fluxograma do algoritmo *SGA*



Fonte: Autor

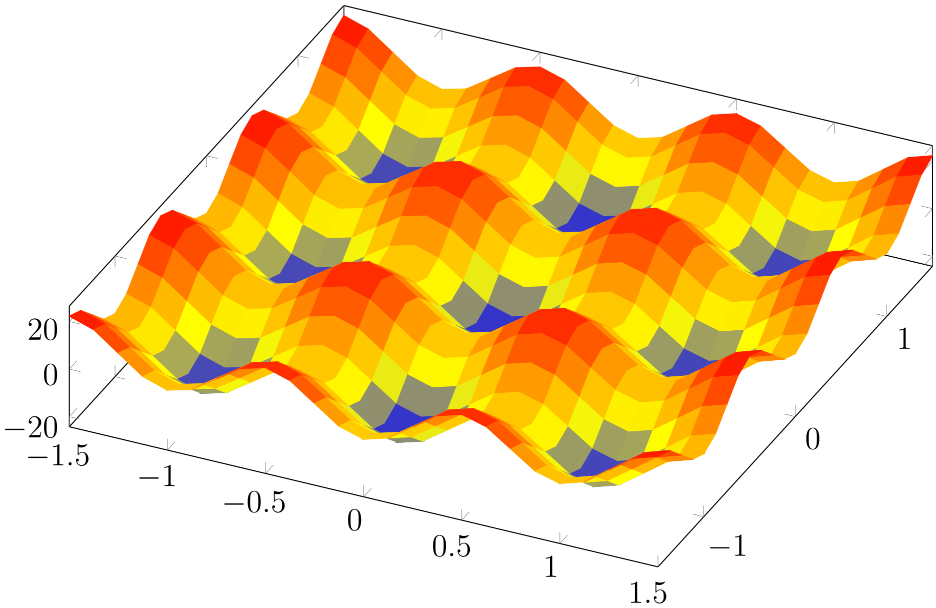
4.8 Problema-exemplo

Para melhor visualizar a forma de atuação do algoritmo é realizada a otimização de um problema-exemplo. O função escolhida para ser otimizada é a seguinte:

$$f(x,y) = x^2 - 10\cos(2\pi x) - 10\cos(2\pi y) + y^2 \tag{4.7}$$

Esta função conta com vários mínimos locais e um mínimo global localizado na origem (0,0), com um valor da função de -20. Na Figura 4 verifica-se a representação 3D da função estudada.

Figura 4: Representação da função

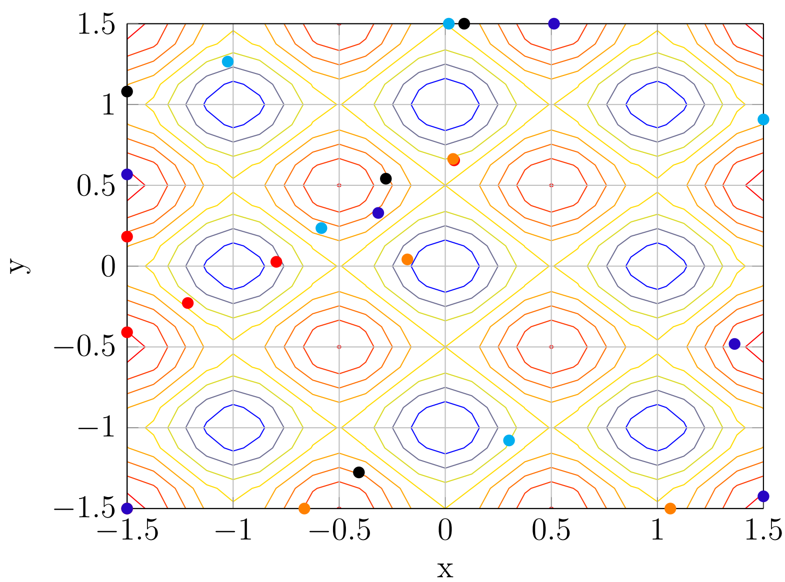


Fonte: Autor

Para esta análise utilizou-se como parâmetros do algoritmo: $n_{pop} = 25$, $n_g = 5$, $it = 30$, $it^{max}_{global} = 21$. Para a aleatoriedade utilizou-se $\alpha_0 = 1.5$ e $\alpha_{min} = 0,005$. Para a mutação do grupo foi usado $n_{perturb} = 1$. Nas Figuras 5, 6, 7, 8 e 9 é possível observar a evolução do algoritmo ao longo das iterações.

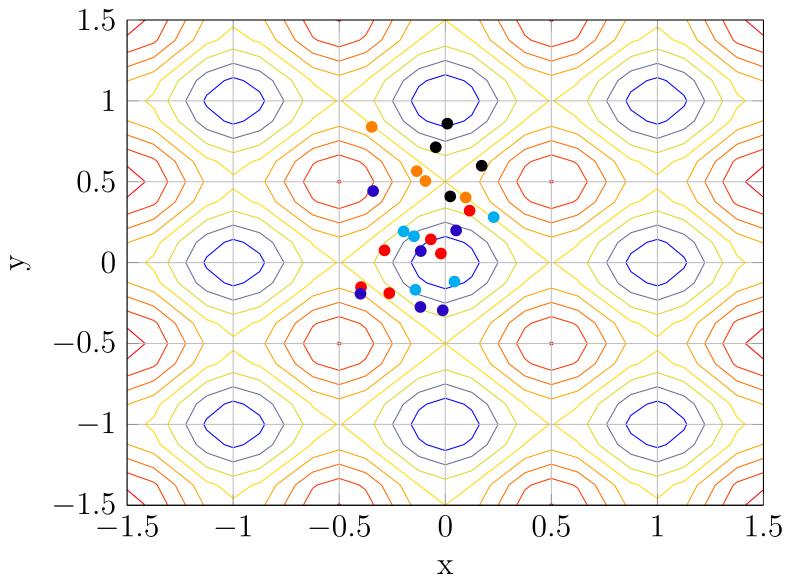
Na iteração inicial (Figura 5) tem-se indivíduos espalhados aleatoriamente pelo domínio. Cada cor diferente representa uma família. Estando na etapa global, a cada iteração o melhor indivíduo de cada família é mantido, gerando-se na iteração seguinte uma nova família a partir do mesmo. Já na iteração 5 (Figura 6)) é possível observar o início de um agrupamento da população na região central, que contém o valor mínimo global -20, no ponto (0,0). Neste ponto já se tem um valor bastante próximo do ótimo, -19,3 representado pelo ponto vermelho que possui coordenadas (-0,021; 0,056). Na iteração seguinte (Figura 7) nota-se um maior agrupamento dos indivíduos, causados pela redução da aleatoriedade. Já tem-se indivíduos de diferentes famílias em localizações bastante próximas da origem. Na iteração 17 (Figura 8) tem-se um agrupamento ainda maior dos indivíduos, com o melhor indivíduo da população, representada pelo ponto preto, com coordenadas de (0,0026; 0.0071) tendo uma avaliação da função objetivo de -19,9886. Dando prosseguimento nas iterações tem-se o início da etapa local na iteração 22, onde o resultado é então refinado. Na última iteração (Figura 9) percebe-se que todos os indivíduos migraram para a origem, uma vez que na fase local não há mais distinção de famílias, sendo selecionados os melhores indivíduos do grupo de busca. Ao fim das iterações obtém-se o ótimo global de -20 nas coordenadas (0,0).

Figura 5: Problema-exemplo - Iteração 1



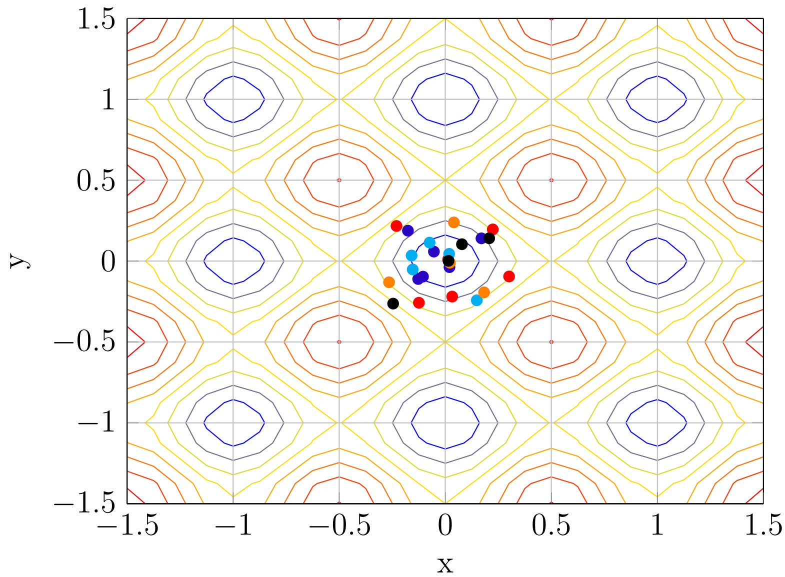
Fonte: Autor

Figura 6: Problema-exemplo - Iteração 5



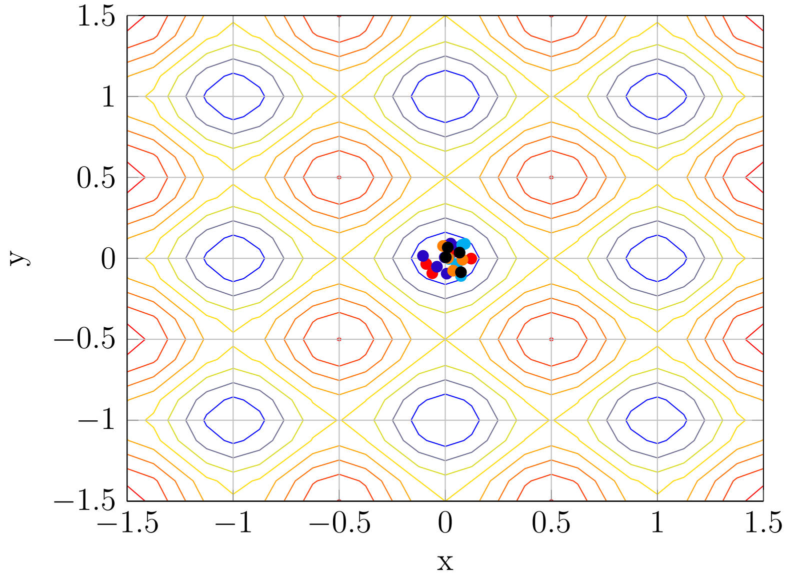
Fonte: Autor

Figura 7: Problema-exemplo - Iteração 10



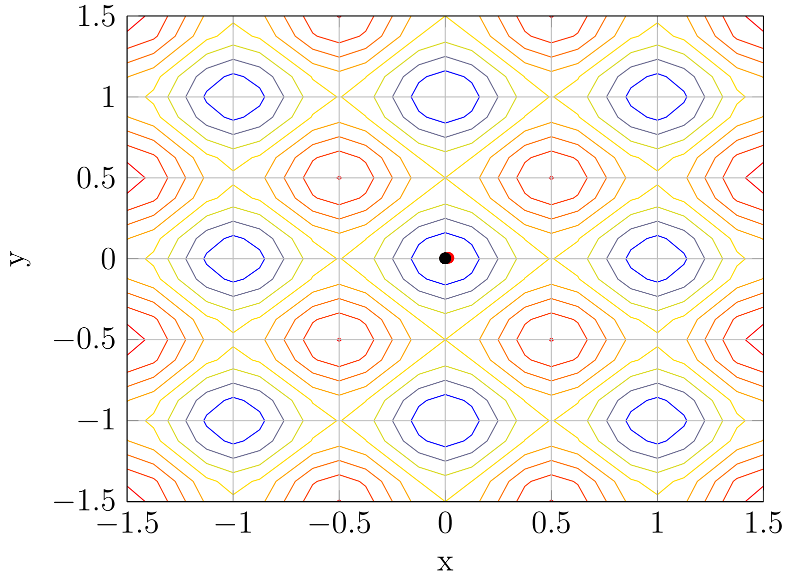
Fonte: Autor

Figura 8: Problema-exemplo - Iteração 17



Fonte: Autor

Figura 9: Problema-exemplo - Iteração 25



Fonte: Autor

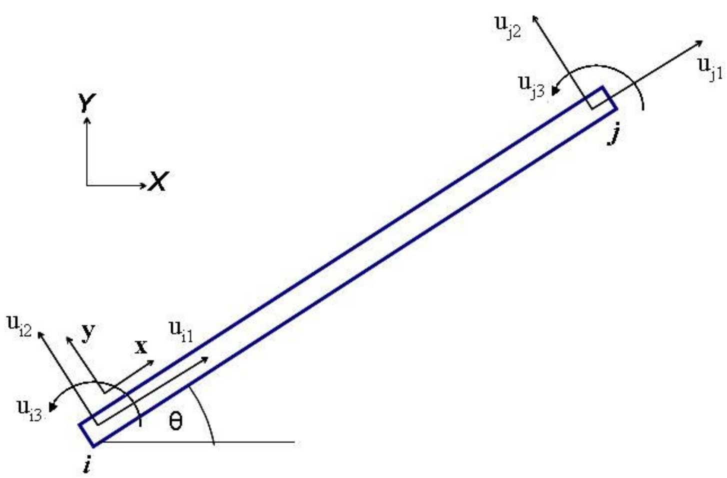
5 Análise Estrutural

A análise estrutural dos problemas estudados é feita a partir de uma rotina computacional, implementada na plataforma *Matlab* ®, que utiliza o método dos deslocamentos com uma formulação matricial, também conhecido como método da rigidez direta. Neste método as incógnitas principais do problema são deslocamentos e rotações. Todas as outras incógnitas são expressas em termos das incógnitas principais escolhidas e substituídas em equações de equilíbrio que são então resolvidas.

Este método é a base da maioria dos programas de computador utilizados para analisar todos os tipos de estruturas determinadas e indeterminadas, planas e espaciais, incluindo treliças, pórticos e cascas tridimensionais.(LEET et al., 2009)

Para os problemas de pórtico plano, analisados geralmente no plano xy, cada nó do elemento básico conta com 3 graus de liberdade: deslocamento em x, deslocamento em y e rotação em torno do eixo z, conforme a Figura 10.

Figura 10: Graus de liberdade de um elemento de pórtico plano



Fonte: (LOGAN, 2011)

O princípio básico do método dos deslocamentos é resolver a seguinte equação matricial para os deslocamentos:

$$\{F\} = [K]\{d\} \tag{5.1}$$

onde $\{F\}$ é vetor de forças nodais equivalente, $\{d\}$ são os deslocamentos dos graus de liberdade da estrutura é $[K]$ é a matriz de rigidez que relaciona estas duas grandezas.

Cada um dos coeficientes desta matriz é obtido através da reação decorrente da imposição de um deslocamento unitário à um dos graus de liberdade, mantendo-se todos os outros restringidos.

De posse dos deslocamentos, obtém-se os esforços internos nas barras da estrutura.

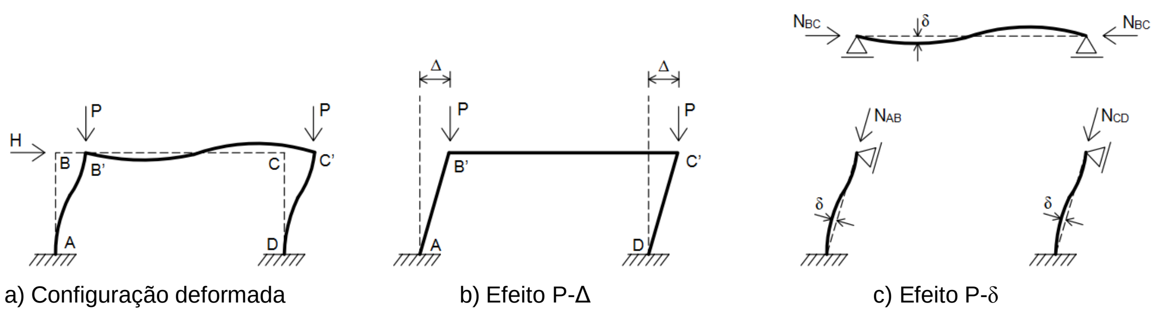
5.1 Efeitos de segunda ordem

Tanto a norma de estruturas de aço brasileira ABNT (2008) quanto a americana AISC (2010), enfatizam a necessidade da consideração dos esforços de segunda ordem nas estruturas. Basicamente estes esforços surgem a partir das cargas atuando na estrutura em sua condição deformada. Estes efeitos são particularmente importantes nos pórticos, principalmente nas estruturas analisados neste estudo, que não possuem contraventamentos impedindo o deslocamento lateral.

A Figura 11 ilustra dois dos tipos de efeitos de segunda ordem que ocorrem nas estruturas:

- Efeito P-δ: efeito localizado, presente em elementos carregados axialmente que possuam um deslocamento em relação ao eixo que liga seus nós.
- Efeito P-Δ: efeito atuante na estrutura como um todo, causado pelas cargas verticais atuando no pórtico com deslocamento lateral.

Figura 11: Efeitos de segunda ordem em pórticos



Fonte: (SILVESTRE; CAMOTIM, 2007)

A utilização do processo P-Δ para a consideração dos esforços adicionais na estrutura acarreta em um cálculo iterativo, ou seja, envolve um custo computacional mais elevado. Deste modo, optou-se por uma consideração, sendo esta permitida pelas normas americana e brasileira, mais simplificada porém eficiente do ponto de vista computacional.

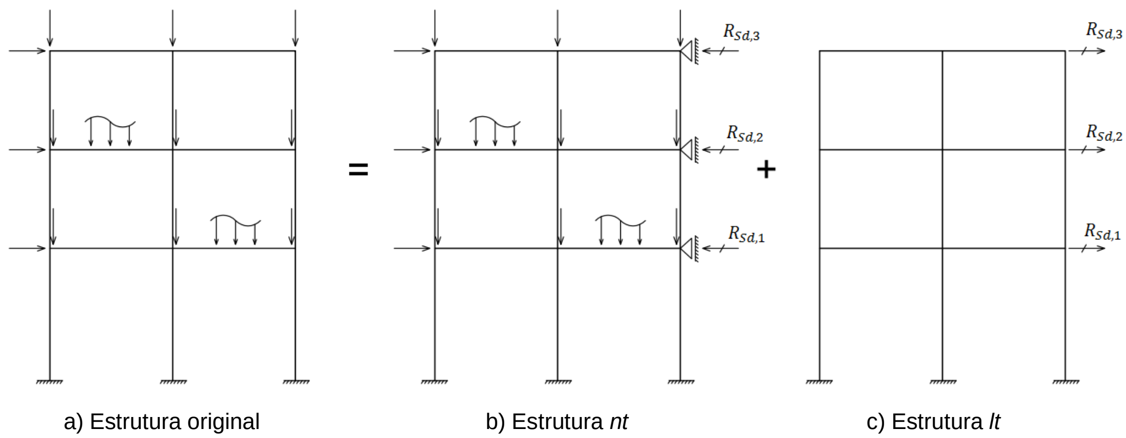
O método utilizado para a consideração destes efeitos é o Método da amplificação de esforços solicitantes (B_1-B_2). Por este método, os esforços solicitantes de projeto resultantes da análise de segunda ordem podem ser obtidos de forma aproximada, aplicando-se coeficientes B_1 e B_2 , respectivamente, aos resultados de duas análises lineares a serem superpostas: análise da estrutura impedida de deslocar-se lateralmente (estrutura nt) e análise da estrutura deslocável (estrutura lt) sujeita apenas às cargas horizontais iguais às reações obtidas na estrutura nt, conforme as equações 5.2 e 5.3 (PFEIL; PFEIL, 2014)

$$M_u = B_1\ M_{nt} + B_2\ M_{lt} \tag{5.2}$$

$$P_u = P_{nt} + B_2\ P_{lt} \tag{5.3}$$

onde M_{nt} e P_{nt} são, respectivamente, o momento e esforço axial obtidos da análise do pórtico sem deslocamento lateral, e M_{lt} e P_{lt} são o momento e esforço axial obtidos da análise da estrutura deslocável.

Figura 12: Divisão da estrutura para o cálculo de B_1 e B_2



Fonte: (ABNT, 2008)

Desta forma, na rotina computacional, é necessário realizar a análise da estrutura duas vezes apenas. Utilizando uma simplificação permitida pela norma, ao invés de se restringir o deslocamento lateral para o cálculo da estrutura nt, são aplicadas somente as cargas verticais da estrutura. Tal consideração não causa deslocamentos horizontais significativos, tornando possível trabalhar-se com as mesmas condições de vinculação para os dois modelos.

O multiplicador B_1 leva em conta o efeito P-δ e é dado pela equação:

$$B_1 = C_m\ \frac{1}{1 - P/P_{cr}} \tag{5.4}$$

onde C_m é dado por:

$$C_m = 0,6 - 0,4\,\frac{M_1}{M_2} \tag{5.5}$$

sendo M_1 e M_2 os momentos nas extremidades do elemento, com $|M_2| > |M_1|$.

Já o multiplicador B_2, que é calculado para cada pavimento da estrutura e que leva em conta o efeito P-Δ, é dado pela seguinte equação:

$$B_2 = \frac{1}{1 - \dfrac{1}{0,85}\dfrac{\Delta_h}{h}\dfrac{\sum N_d}{\sum H_d}} \tag{5.6}$$

onde: h é a altura do pavimento analisado, Δ_h é o deslocamento inter-pavimentos e $\sum N_d$ e $\sum H_d$ são respectivamente, o somatório de cargas verticais e horizontais no pavimento.

5.2 Obtenção de esforços máximos

A obtenção da maioria dos esforços máximos axiais e de momento se dá através do método dos deslocamentos, que fornece corretamente os esforços nas extremidades das barras. Há um caso específico porém, no caso das vigas, que apresentam nos problemas analisados carregamentos distribuídos. Nestes casos, o diagrama de momentos é quadrático, sendo necessário verificar se o momento máximo ocorre nas extremidades ou no vão da mesma. O cálculo deste momento no vão, que é posteriormente comparado aos momentos na extremidade para se obter o máximo, é realizado da seguinte maneira:

$$M_{\text{vão}} = M_1 - \frac{{V_1}^2}{2q} \tag{5.7}$$

onde M_1 é o momento em uma extremidade, V_1 é o esforço cortante nesta mesma extremidade e q é a carga distribuída aplicada.

6 Dimensionamento

Na avaliação dos esforços em pórticos, verifica-se que tanto vigas como pilares apresentam-se solicitados por uma combinação de esforços axiais e momento fletores. Estes elementos estruturais são dimensionados para a flexo-tração ou flexo-compressão conforme a especificação do “American Insititute of Steel Cosntrutction” AISC (2010) utilizando-se a seguinte expressão de interação:

$$\begin{cases} \dfrac{P_u}{\phi P_n} + \dfrac{8}{9}\left(\dfrac{M_{ux}}{\phi_b M_{nx}} + \dfrac{M_{uy}}{\phi_b M_{ny}}\right) \quad \text{para} \quad \dfrac{P_u}{\phi P_n} \geq 0,2 \\ \\ \dfrac{P_u}{\phi P_n} + \dfrac{8}{9}\left(\dfrac{M_{ux}}{\phi_b M_{nx}} + \dfrac{M_{uy}}{\phi_b M_{ny}}\right) \quad \text{para} \quad \dfrac{P_u}{\phi P_n} < 0,2 \end{cases} \tag{6.1}$$

Nesta expressão, P_u é o esforço solicitante axial de tração ou compressão, M_{ux} e M_{uy} são os momentos fletores solicitantes nos eixos x e y. Há também o esforço resistente axial P_n e os momentos resistentes M_{nx} e M_{ny} nos dois eixos principais da seção. Os multiplicadores ϕ e ϕ_b são os fatores de resistência. O multiplicador ϕ representa o fator de resistência axial e assume o valor de 0,85 para peças comprimidas e 0,90 para peças tracionadas, enquanto que o multiplicador ϕ_b, representando o fator de resistência à flexão assume o valor de 0,90.

6.1 Resistência axial

O esforço axial resistente P_n para hastes metálicas, sem efeito de flambagem local, sujeitas à compressão axial é dado pela seguinte equação:

$$P_n = F_{cr} \cdot A_g \tag{6.2}$$

onde F_{cr} representa a tensão crítica de flambagem e A_g a área da seção transversal do perfil.

O valor de F_{cr} é dado pela seguinte relação entre a esbeltez do elemento (KL/r) e a carga crítica de Euler (F_e)

$$F_{cr} = \begin{cases} \left[0,658^{\frac{F_y}{F_e}}\right] F_y & \text{se } \dfrac{KL}{r} \leq 4,71\sqrt{\dfrac{E}{F_y}} \\ \\ 0,877 \cdot F_e & \text{se } \dfrac{KL}{r} > 4,71\sqrt{\dfrac{E}{F_y}} \end{cases} \tag{6.3}$$

onde F_y é a tensão de escoamento do aço e a carga crítica de Euler é definida como:

$$F_e = \frac{\pi^2 E}{\left(\dfrac{KL}{r}\right)^2} \tag{6.4}$$

A flambagem aqui analisada apresenta caráter global no elemento, no qual a instabilidade ocorre na peça como um todo, havendo deslocamentos em relação as extremidades. Tanto a norma brasileira ABNT (2008) quanto a americana AISC (2010) adotam uma curva única para a caracterização da tensão crítica, de onde são definidas as relações acima.

Esta curva é dividida em dois trechos. Para hastes esbeltas, o modo de falha é baseado na resistência à flambagem elástica. O multiplicador 0,877 da carga crítica de Euler leva em conta a imperfeição geométrica e é usado para estabelecer a resistência nominal da haste. Para hastes mais curtas, a curva é baseada tanto em resultados experimentais quanto analíticos e leva em conta efeitos de momentos causados por carregamento excêntrico ou desaprumo da peça e também as tensões residuais provenientes do processo de fabricação do perfil.(GALAMBOS; SUROVEK, 2008)

Há ainda que se considerar o efeito de flambagem de placa, uma instabilidade que ocorre localmente nas placas componentes do perfil comprimido, causando deslocamentos laterais na forma de ondulações. No caso dos perfis analisados, do tipo “I”, verifica-se a flambagem local na alma e na mesa da seção. Sendo os mesmos constituídos de aço do tipo A36, verficou-se que para todos os perfis I da série W, a seguinte relação é atendida

$$\frac{b}{2\ t_f} \leq 0,56\sqrt{\frac{E}{F_y}} \tag{6.5}$$

onde b é a largura da mesa, e t_f sua espessura, como visto na Figura 13.

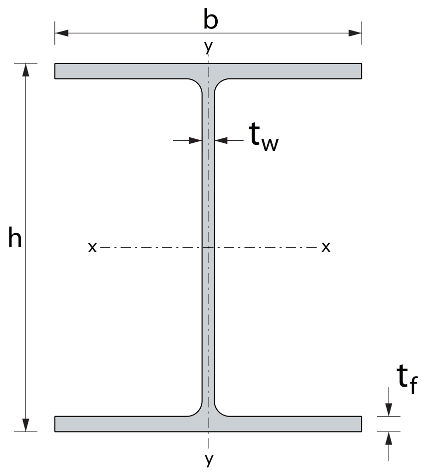
Desta maneira tem-se que todas as mesas são consideradas compactadas não ocorrendo instabilidade local nas mesmas.

Para o caso da verificação da alma esbelta tem-se a seguinte relação:

$$\frac{h}{t_w} < 1,49\sqrt{\frac{E}{F_y}} \tag{6.6}$$

onde h é a altura da alma, e t_w sua espessura, como visto na Figura 13.

Figura 13: Medidas dos elementos do perfil I



Fonte: (GARDNER, 2011)

Caso a relação não seja atendida, tem-se então a situação de alma esbelta, sendo necessário o cálculo do fator de redução Q_a, expresso por:

$$Q_a = \frac{A_g}{A_e} \tag{6.7}$$

sendo A_e a área equivalente do perfil.

Esta área equivalente é calculada utilizando-se a altura reduzida b_e dada por:

$$b_e = 1,92t_w\sqrt{\frac{E}{f}}\left[1 - \frac{0,34}{h/t_w}\sqrt{\frac{E}{f}}\right] \leq h \tag{6.8}$$

Feito o cálculo de Q_a, tem se a seguinte expressão de F_{cr} que leva em conta o fator de redução:

$$F_{cr} = \begin{cases} \left[Q_a \; 0,658^{\frac{Q_a F_y}{F_e}} \right] F_y & \text{se } F_e \geq 0,44 \; Q_a F_y \\ \\ 0,877 \; F_e & \text{se } F_e < 0,44 \; Q_a F_y \end{cases} \tag{6.9}$$

A partir do F_{cr} para o caso de alma esbelta, a resistência à compressão do perfil é calculada pela Equação 6.2.

Para o caso de hastes sujeitas à esforços axiais de tração, utiliza-se novamente a Equação 6.2, substituindo-se F_{cr} pela tensão de escoamento do aço (F_y).

6.2 Resistência à flexão

Da mesma forma que ocorre para o dimensionamento à esforços axiais, o cálculo do esforço resistente de flexão depende da classificação da seção quanto a sua esbeltez.

Segundo as normas americana (AISC, 2010) e brasileira (ABNT, 2008), as seções de elementos sujeitos à flexão podem ser dividias em três classes conforme a influência da flambagem local sobre os respectivos momentos fletores resistentes (M_{res}):

- Seção compacta - atinge o momento de plastificação total ($M_{res} = M_p$), exibindo suficiente capacidade de rotação inelástica para configurar uma rótula plástica;
- Seção semi-compacta - a flambagem local ocorre após ter desenvolvido plastificação parcial ($M_{res} > M_y$), não apresentando rotação significativa;
- Seção esbelta - a ocorrência da flambagem local impede que seja atingido o momento de início de plastificação ($M_{res} < M_y$)

Para perfis do tipo "I", a classificação quanto ao tipo de seção se dá através da comparação da esbeltez do elemento componente (alma ou mesa), em relação aos limites de cada região, λ_p (compacto/semi-compacto) e λ_r (semi-compacto/esbelto). Estes limites para elementos solicitados à flexão em relação ao seu eixo principal de inércia são:

Tabela 1: Parâmetros de esbeltez na flexão

Esbeltez da placa componente	Limites λ entre regiões	
	λ_p	λ_r
$\dfrac{b}{2\ t_f}$	$0,38\sqrt{\dfrac{E}{F_y}}$	$1,0\sqrt{\dfrac{E}{F_y}}$
$\dfrac{h}{t_w}$	$3,76\sqrt{\dfrac{E}{F_y}}$	$5,70\sqrt{\dfrac{E}{F_y}}$

Fonte: (AISC, 2010)

Em relação à implementação computacional dos exemplos analisados, verificou-se que todas as almas dos perfis série da W, utilizando aço do tipo A36 são compactas. Da mesma forma, as mesas são compactas com exceção do perfil W6x15, que é semi-compacto, sendo tratado na rotina de análise como um caso especial.

Na ocorrência de ambas alma e mesa serem compactas, são dois os estados limites na flexão que devem ser verificados: escoamento da seção bruta e flambagem lateral com torção, utilizando-se o menor valor obtido como resistência nominal à flexão M_n.

6.2.1 Plastificação da seção transversal

Para o estado limite de plastificação da seção transversal, o momento nominal (M_n) é igual ao próprio momento de plastificação (M_p).

$$M_n = M_p = ZF_y \tag{6.10}$$

onde Z é o módulo plástico da seção e F_y é a tensão de escoamento do aço.

6.2.2 Flambagem lateral com torção

O momento nominal segundo este estado limite depende da classificação do elemento quanto a seu comprimento. A classificação dos elementos varia entre curtos, intermediários e longos, dando-se através da comparação de L_b, sendo este, o comprimento de trecho entre dois pontos de contenção lateral, e os valores limite L_p e L_r dados por:

$$L_p = 1,76\ r_y\sqrt{\frac{E}{F_y}} \tag{6.11}$$

$$L_r = 1,95\ r_{ts}\ \frac{E}{0,7F_y}\sqrt{\frac{J}{S_x h_0} + \sqrt{\left(\frac{J}{S_x h_0}\right)^2 + 6,76\left(\frac{0,7F_y}{E}\right)^2}} \tag{6.12}$$

onde

$$r_{ts} = \sqrt{\frac{\sqrt{I_y C_w}}{S_x}} \tag{6.13}$$

Sendo $L_b < L_p$, o elemento é dito curto, e o momento resistente nominal é calculado pelo escoamento da seção bruta, conforme a Equação 6.10.

Sendo $L_b > L_r$, o elemento é dito longo, e o momento resistente nominal é calculado a partir da seguinte expressão:

$$M_n = C_b\ \frac{\pi^2 E I_y}{L_b^2}\sqrt{\frac{C_w}{I_y}\left(1 + 0,039\ \frac{J L_b^2}{C_w}\right)} \tag{6.14}$$

onde C_b é o coeficiente que leva em conta o efeito favorável de o momento não ser uniforme a longo do trecho L_b, dado por:

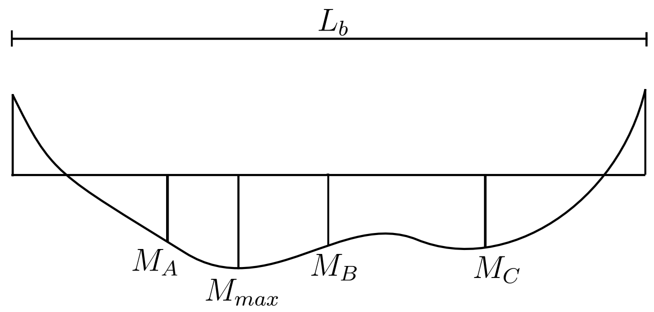
$$C_b = \frac{12,5M_{max}}{2,5M_{max} + 3M_A + 4M_B + 3M_C} \leq 3,0 \tag{6.15}$$

O cálculo de C_b nada mais é do que uma média ponderada, relacionando o momento máximo M_{max} entre pontos de contraventamento e os valores de momento à distâncias de $L_b/4$, $L_b/2$ e $3L_b/4$ de um dos dois pontos de contenção lateral, conforme ilustra a Figura 14.

Há ainda o caso de elementos intermediários, quando o ocorre que $L_p < L_b < L_r$. Neste caso é feita uma interpolação linear entre o momento de plastificação M_p e o momento M_r:

Figura 14: Momentos para o cálculo de C_b

[DMF]



Fonte: Autor

$$M_n = M_p - (M_p - M_r)\left(\frac{L_b - L_p}{L_r - L_p}\right) \tag{6.16}$$

onde $M_r = (F_y - 0,3F_y)S_x = 0,7F_yS_x$, sendo descontado 30% do valor da tensão de escoamento do aço referente à parcela de tensão residual, que provém do processo de fabricação.

$$M_n = F_{cr}S_x \tag{6.17}$$

No caso especial do perfil W6x15, que possui alma compacta e mesa semi-compacta, além das considerações dos dois estados limites já mencionados, cabe ainda a verificação do estado limite de flambagem da mesa, no qual verifica-se o valor de M_n a partir da equação abaixo:

$$M_n = M_p - (M_p - M_r)\left(\frac{\lambda - \lambda_p}{\lambda_r - \lambda_p}\right) \tag{6.18}$$

onde $\lambda = \dfrac{b}{t}$ (esbeltez da mesa) e λ_p e λ_r são os limites indicados na Tabela 1.

6.3 Contraventamento

Para todos os problemas de otimização de pórticos estudados neste trabalho, existe a consideração de que o pórtico está livre para se deslocar lateralmente, ou seja, tratam-se de pórticos não contraventados.

Como já visto acima, na etapa do dimensionamento de perfis, é necessário conhecer o comprimento efetivo da peça de modo a calcular sua resistência à compressão, sendo este, o comprimento destravado da peça, que fica sujeito ao fenômeno da flambagem. O comprimento efetivo é dado com relação à uma constante K, que varia conforme a vinculação adotada para as extremidades da barra.

$$L_e = K\ L \tag{6.19}$$

O comprimento efetivo de cada barra varia para cada direção analisada. No caso da direção x, que se refere ao deslocamento lateral do pórtico plano, a seguinte equação proposta por Dumonteil (1992) é utilizada para o cálculo de K_x, em todos os problemas estudados:

$$K = \sqrt{\frac{1,6\ G_A\ G_B + 4,0(G_A + G_B) + 7,5}{G_A + G_B + 7,5}} \tag{6.20}$$

onde G_A e G_B denotam os valores de G nas extremidades da barra analisada.

O valor de G por sua vez, calculado para todos os nós da estrutura, é uma relação entre as rigidezes e comprimentos das vigas e pilares conectados a cada nó. Ele leva em consideração o fato da rigidez de vigas e pilares conectados ao elemento considerado contribuírem para a estabilidade do mesmo.

$$G = \frac{\sum(I_{pilar}/L_{pilar})}{\sum(I_{viga}/L_{viga})} \tag{6.21}$$

No caso de nós restringidos, por exemplo em apoios, utiliza-se o valor de $G = 10$.

6.4 Esforço cortante

O esforço cortante não é verificado nos problemas estudados, uma vez que tal verificação também não é feita pelos autores na literatura, pelo fato de o esforço cortante geralmente não governar o dimensionamento, com exceção de casos específicos como vigas curtas ou com furos.

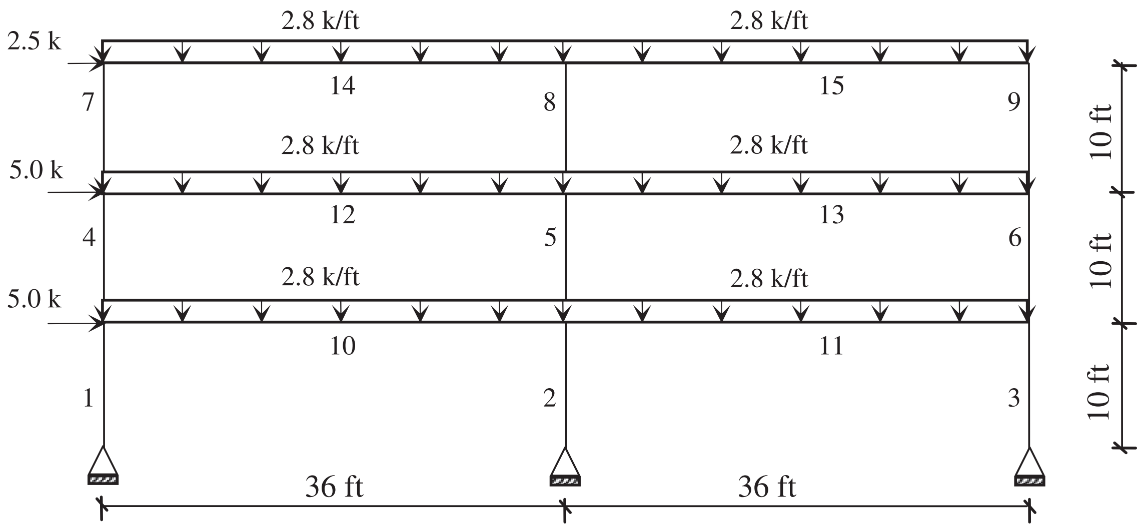
7 Problemas estudados

7.1 Problema 1

Como primeira análise de pórtico utilizou-se um exemplo simples, com o intuito de validação dos resultados. Trata-se de um pórtico plano que contem 15 elementos, dispostos em 2 vãos e 3 pavimentos, utilizado como *benchmark* e já analisado por diversos autores da literatura como: Camp et al. (2005), Dogan e Saka (2012), Pezeshk e Chen (2000), Kaveh e Talatahari (2010a), Degertekin (2008), Togan (2012), etc.

Os carregamentos descritos na Figura 15 já são os carregamentos fatorados, de modo que podem ser aplicadas diretamente as provisões normativas, sem a necessidade da ponderação das ações.

Figura 15: Pórtico - Exemplo 1



1 k = 4,45 kN
1 ft = 0,305 m

Fonte: (TOGAN, 2012)

Para este problema não são estabelecidas restrições quanto à deslocamentos. Tem-se a utilização de perfis de aço, cujo módulo de elasticidade E é igual a 29000 ksi (200 GPa), com uma tensão de escoamento F_y igual a 36 ksi (248,2 MPa) e uma massa específica γ de 0,284 lb/in^3 (7861 kg/m^3). O objetivo do problema é minimizar o peso da estrutura, tendo como restrição as prescrições de dimensionamento vistas na Capítulo 6, conforme a norma AISC (2010).

Os membros são agrupados em dois grupos, que consistem em seis vigas e nove pilares. Trata-se portanto de um problema de otimização discreta com duas variáveis. No projeto, as vigas são selecionadas de uma tabela de 267 perfis do tipo "I", série W (Tabela 2), enquanto que os pilares são selecionados da série W10, ou seja, pilares da série W com 10 polegadas de altura. Foi utilizado o banco de dados com as propriedades de cada perfil disponível no próprio site oficial do AISC. (STEEL..., 2015)

Em relação a este banco de dados, utilizou-se o "AISC Shapes Database V14.1H - Historic", utilizando a edição LRFD 3 dos dados, pelo fato destes apresentarem 267 perfis W, em concordância com o que é utilizado na literatura.

Tabela 2: Tabela de perfis - Série W

Índice i	Perfil W (AISC)	A$(i)(in^2)$	I$(i)(in^4)$
1	W6x8,5	2,51	14,8
2	W6x9	2,68	16,4
3	W8x10	2,96	30,8
$\vdots$	$\vdots$	$\vdots$	$\vdots$
265	W14x730	215	14300
266	W36x798	235	62600
267	W14x808	237	16000

Para o cálculo do fator de comprimento efetivo, K_x, foi utilizado a equação aproximada proposta por Dumonteil (1992), detalhada na seção 6.3. Para cada coluna, o fator de comprimento efetivo fora do plano, K_y foi considerado 1,0. Já para as vigas, utilizou-se o valor de comprimento efetivo de um sexto do vão livre.

7.2 Problema 2

O segundo problema estudado é o de um pórtico plano com 1 vão e 10 andares. Este é um exemplo mais complexo, uma vez que possui efeitos de segunda ordem mais notáveis, além de existirem mais variáveis de projeto a serem otimizadas, visto que a seção das vigas e pilares muda em relação aos andares.

O pórtico é constituído de 30 elementos, sendo 10 vigas e 20 pilares, organizados em nove grupos diferentes. Dentro de um dado grupo, os elementos possuem uma mesma seção, sendo estas consideradas as variáveis discretas de projeto.

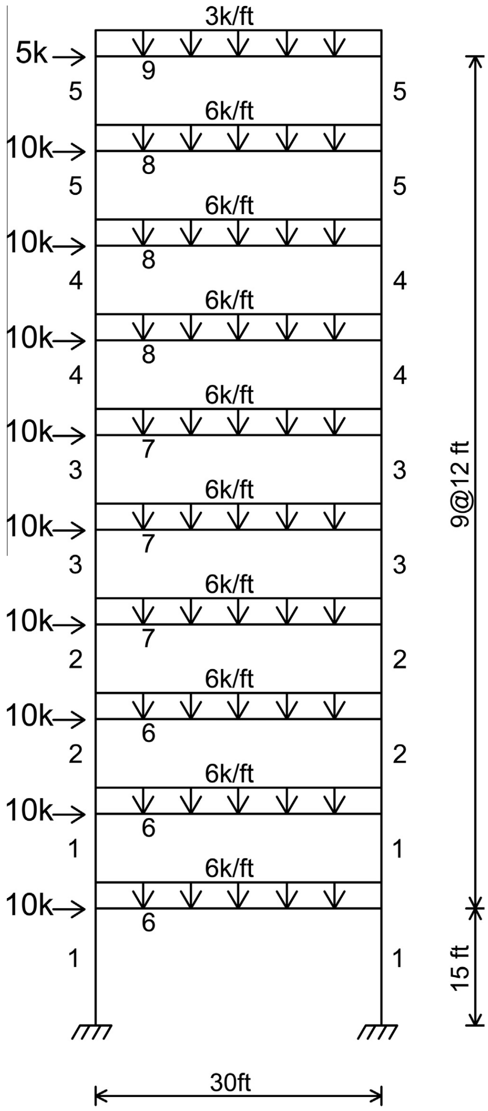
Começando no primeiro pavimento e excluindo a viga da cobertura, tem-se que os elementos de viga precisam manter a mesma seção por três pavimentos consecutivos. Imposição semelhante ocorre para os pilares, a partir do nível da fundação, onde é requerido que permaneçam com a mesma seção a cada dois andares. Na Figura 16 fica indicado quais membros pertencem a cada grupo de perfis com a mesma seção. Para cada um dos 4 grupos de vigas podem ser utilizados todos os 267 perfis I da série W. Para os 5 grupos de pilares, no entanto, fica restrito o uso de perfis W12 e W14, compreendendo 66 perfis.

O material é o mesmo do primeiro problema analisado: aço com tensão de escoamento $F_y = 36$ ksi e módulo de elasticidade $E = 29000$ ksi.

Este problema está sujeito às restrições normativas da norma de projeto de estruturas de aço americana (AISC 2010), bem como uma restrição adicional de deslocamento. O deslocamento entre pavimentos consecutivos não deve ser superior à $h/300$, onde h é a altura entre os pavimentos considerados.

O fator de comprimento efetivo K_x é calculado conforme a equação de Dumonteil (1992), descrita na seção 6.3.

Figura 16: Pórtico - Exemplo 2



No SI: $\begin{array}{rcl} 1\ \text{k} & = & 4{,}45\ \text{kN} \\ 1\ \text{ft} & = & 0{,}305\ \text{m} \end{array}$

Fonte: (MAHERI; NARIMANI, 2014)

7.3 Problema 3

Por fim, tem-se como último problema estudado, um pórtico plano de 3 vãos e 24 andares. A estrutura é composta de 168 elementos, sendo estes 96 pilares e 72 vigas.

Tem-se como cargas aplicadas no pórtico: W = 5761,85 lb, W_1 = 300 lb/ft, W_2 = 436 lb/ft, W_3 = 474 lb/ft e W_4 = 408 lb/ft (1 lb = 4,45 N e 1 ft = 0,305 m). O pórtico foi divido em 20 grupos de projeto, sendo 16 grupos de pilares e 4 grupos de vigas. No caso das vigas, estas podem assumir qualquer um dos 267 perfis da série W, já os pilares ficam limitados aos 37 perfis W14. Tais limitações, para as 20 variáveis de projeto, resulta em um espaço de busca contendo $6,2 \times 10^{34}$ combinações das variáveis.

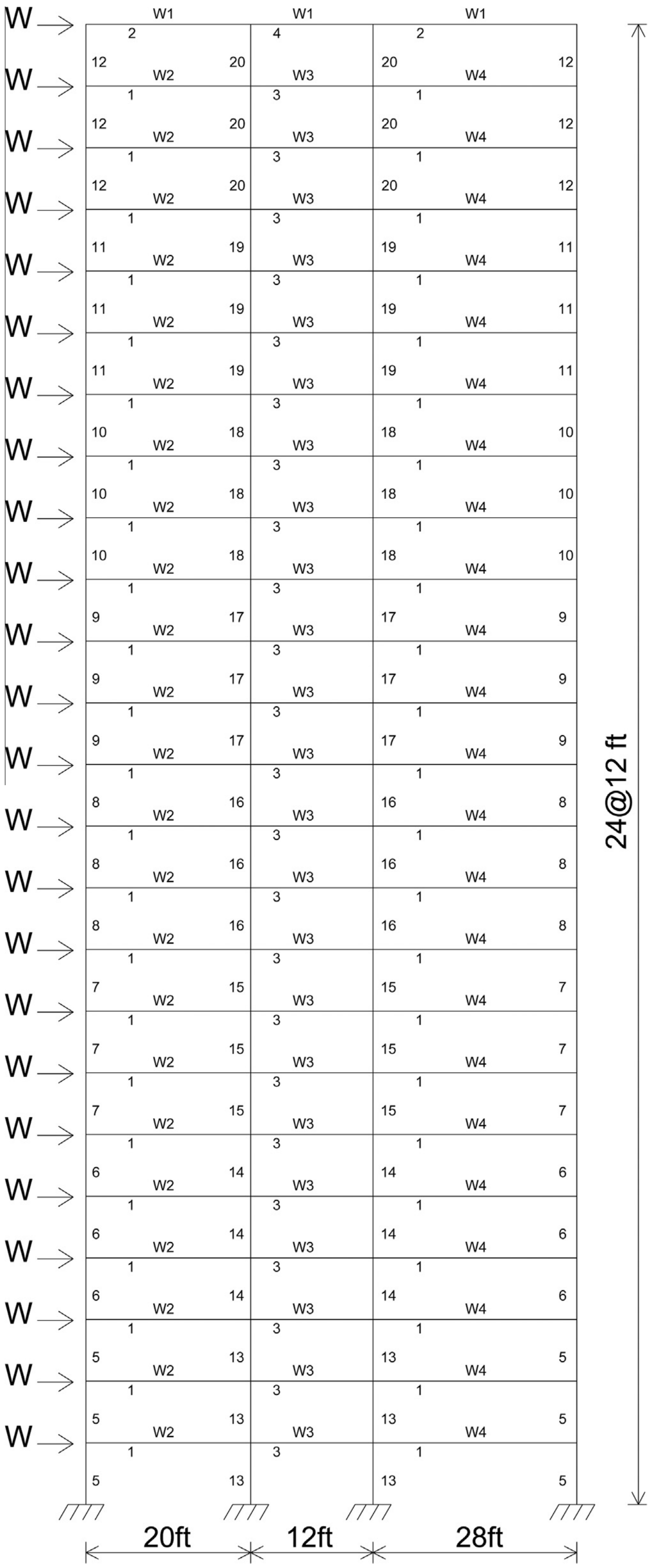
A informação dos grupos associados a cada elemento pode ser visualizada na Figura 17.

As propriedades do aço utilizado variam um pouco em relação aos outros problemas. Utiliza-se um aço com módulo de elasticidade E = 29.732 ksi (205 GPa) e tensão de escoamento F_y = 33,4 ksi (230,3 MPa).

As mesmas restrições aplicadas ao Problema 2, são também aplicadas à este problema, ou seja, restrições normativas a cada elemento e deslocamento inter-pavimentos máximo igual à $h/300$.

O comprimento efetivo K_x dos elementos é calculado através da equação proposta por Dumonteil (1992). Na direção fora do plano, o comprimento efetivo é de K_y = 1,0.

Figura 17: Pórtico - Exemplo 3



Fonte: (MAHERI; NARIMANI, 2014)

8 Análise Numérica

8.1 Problema 1

8.1.1 Resultados

Para este problema foram utilizados como parâmetros do algoritmo: uma população total $n_{pop} = 20$, um grupo de busca com $n_g = 4$ indivíduos, realizando um número de iterações $it = 20$, sendo destas $it_{global}^{max} = 14$, referente a etapa global e com $n_{perturb} = 1$. Para a aleatoriedade utilizou-se: $\alpha_0 = 0,45$ e $\alpha_{min} = 0,01$. O número de avaliações da função objetivo é calculado a partir da expressão $(n_{pop} - n_g)\ it$.

Na Tabela 3 são apresentados os resultados obtidos por cada autor.

Tabela 3: Resultados da otimização do Problema 1

Grupo de elementos	Perfis W				
	Pezeshk e Chen (2000)	Camp et al. (2005)	Degertekin (2008)	Togan (2012)	***SGA*** **(este estudo)**
1 (Pilares)	W10x60	W10x60	W10x54	W10x49	W10x60
2 (Vigas)	W24x62	W24x62	W21x62	W24x62	W24x62
Nº de avaliações	1800	3000	1853	800	320
Peso (lb)	18.792	18.792	18.292	17.789	18.792

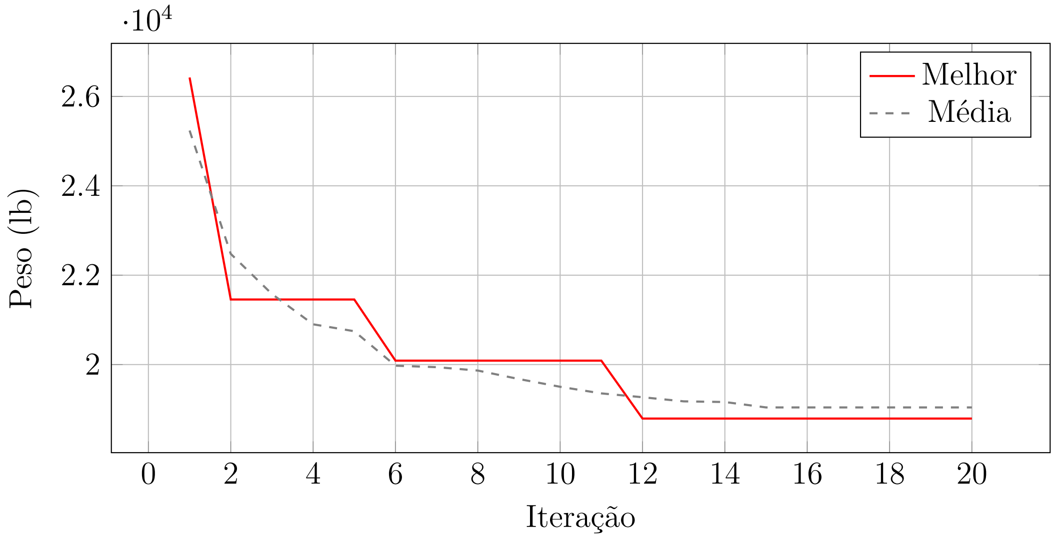
Na Tabela 4 são apresentados os resultados referentes às execuções independentes do algoritmo.

Tabela 4: Resumo da série de testes - Problema 1

Número de execuções independentes do algoritmo	150
Mínimo observado	18.792 lb
Média	19.041 lb
Desvio padrão	453,88 lb

Na Figura 18, fica representada a convergência do método de busca, ou seja, o melhor valor encontrado para cada iteração. Os itens "Melhor" e "Média" visualizados na legenda desta e de outras curvas apresentadas refere-se, respectivamente, aos resultados da melhor execução da série independente e da média entre valores desta série.

Figura 18: Curvas de convergência - Problema 1



Fonte: Autor

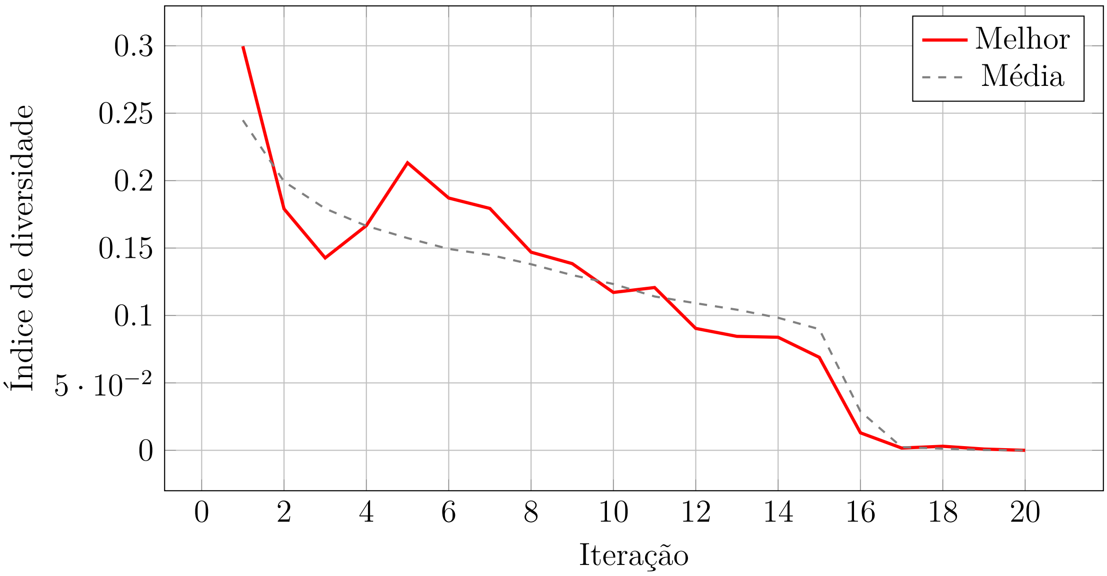
Além das curvas do convergência, foi elaborado mais um recurso para visualização do comportamento do algoritmo. Serão exibidas também para todos os problemas analisados uma curva "Índice de diversidade x iteração", como visto na Figura 19, novamente com os valores da melhor execução e média da série independente de execuções realizada. A partir deste gráfico, pode-se observar a variação da diversidade da população ao longo da execução do algoritmo. O índice, concebido por Kaveh e Zolghadr (2014), tem a seguinte formulação:

$$\frac{1}{n_{pop}} \sum_{i=1}^{n_{pop}} \sqrt{\sum_{j=1}^{N_g} \left(\frac{O(i) - X_j(i)}{X_{i,max} - X_{i,min}} \right)^2} \tag{8.1}$$

onde $O(i)$ é o valor da i-ésima variável de projeto do melhor indivíduo encontrado até o momento na população e $X_{i,min}$ e $X_{i,max}$ são, respectivamente, os limites superior e inferior de uma dada variável.

O índice pode ser descrito como a média na população da raiz quadrada da soma do quardado das distâncias relativas das coordenadas de um dado indivíduo em relação ao melhor indivíduo encontrado até o momento. Desta maneira, quanto menor o índice de diversidade, mais próximos estão os indivíduos.

Figura 19: Diversidade da população - Problema 1



Fonte: Autor

8.1.2 Discussão

O objetivo maior deste primeiro problema era a validação da rotina, e a possibilidade de experimentação dos parâmetros do mesmo. Por se tratar de um problema pequeno, com apenas dois grupos de projeto (variáveis), conseguiu-se realizar diversos testes no que diz respeito às configurações do algoritmo. Deste modo, obteve-se experiência e maior sensibilidade na influência dos parâmetros nos resultados. A experiência adquirida possibilitou uma melhor escolha nas configurações da rotina para os problemas similares, porém mais complexos, vistos em seguida.

Em relação aos resultados, o *SGA* obteve o mesmo valor ótimo já encontrado por dois outros autores (Tabela 4), em um número reduzido (320) de avaliações da função objetivo.

Um fato importante a ser analisado é que uma busca exaustiva, ou seja, o teste de todas as combinações de perfis possíveis, já foi realizada por Pezeshk e Chen (2000), validando o ótimo global de 18.972 lb.

Tal busca torna-se possível uma vez existem 267 opções de seção para o grupo de vigas e 18 opções para o grupo de pilares, totalizando um espaço de busca de apenas 4806 configurações para a estrutura, bem dentro das capacidades de qualquer computador. A mesma foi repetida pelo autor deste trabalho, obtendo-se o mesmo resultado. Esta situação foi reportada também por Murren e Khandelwal (2014), que classificaram a estrutura apresentada por Degertekin (2008) como inviável. Da mesma forma, o resultado de Togan (2012), que se apresenta com um valor menor do que o obtido na busca exaustiva, foi descartado da comparação. Tal situação pode estar ligada a diferentes considerações frente a norma, resultando em diferentes avaliações para as restrições do problema.

Com relação a diversidade da população, através da Figura 19 nota-se uma queda na diversidade a partir da décima quinta iteração, o que corresponde com o início da etapa local de busca. A diversidade se aproxima de 0 pelo fato do grupo de busca ser pequeno, com apenas 4 indivíduos e os mesmos apresentarem valores muito próximos, chagando a se igualar em alguns casos.

8.2 Problema 2

8.2.1 Resultados

Para o segundo problema analisado foram utilizados como parâmetros do algoritmo: uma população total $n_{pop} = 50$, grupos de busca com $n_g = 10$, realizando $it = 150$ iterações, sendo destas $it^{max}_{global} = 107$, referente a etapa global além de $n_{perturb} = 4$, referente às perturbações. Para a aleatoriedade utilizou-se: $\alpha_0 = 0,18$ e $\alpha_{min} = 0,06$.

Na Tabela 5 são apresentados os resultados referentes às análises do Problema 2 por diversos autores.

Tabela 5: Resultados para cada grupo - Problema 2

Grupos de elementos	Perfis W				
	Pezeshk e Chen (2000)	Camp et al. (2005)	Degertekin (2008)	Kaveh e Talatahari (2010a)	***SGA* (este estudo)**
1 (Pilares Pav. 1-2)	W14x233	W14x233	W14x211	W14x233	W14x233
2 (Pilares Pav. 3-4)	W14x176	W14x176	W14x176	W14x176	W14x176
3 (Pilares Pav. 5-6)	W14x159	W14x145	W14x145	W14x145	W14x145
4 (Pilares Pav. 7-8)	W14x99	W14x99	W14x90	W14x90	W14x99
5 (Pilares Pav. 9-10)	W12x79	W12x65	W14x61	W12x65	W14x74
6 (Vigas Pav. 1-3)	W33x118	W30x108	W33x118	W33x118	W30x108
7 (Vigas Pav. 4-6)	W30x90	W30x90	W30x99	W30x90	W30x90
8 (Vigas Pav. 7-9)	W27x84	W27x84	W24x76	W24x76	W24x84
9 (Viga Pav. 10)	W24x55	W21x44	W18x46	W14x30	W24x62
Nº de avaliações	3.000	8.300	3.690	2.440	6.000
Peso (lb)	65.136	62.610	61.864	61.820	63.534

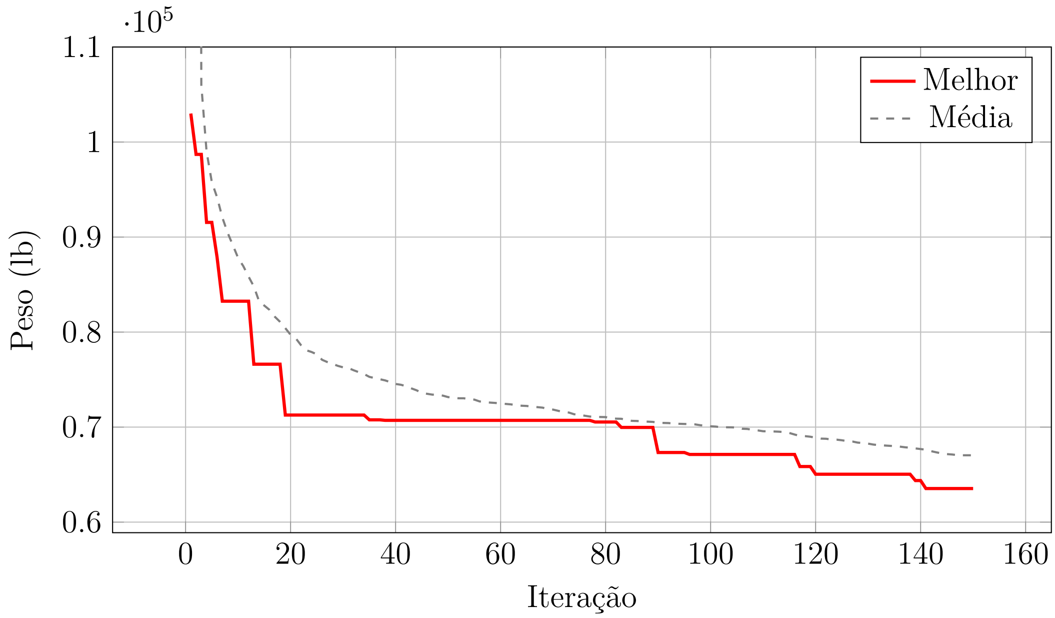
A seguir, na Tabela 6 é apresentado o resumo dos resultados obtidos para a série de execuções.

Tabela 6: Resumo da série de testes - Problema 2

Número de execuções independentes do algoritmo	50
Mínimo observado	63.534 lb
Média	67.035 lb
Desvio padrão	1.940,8 lb

A Figura 20 ilustra a convergência da rotina para este problema.

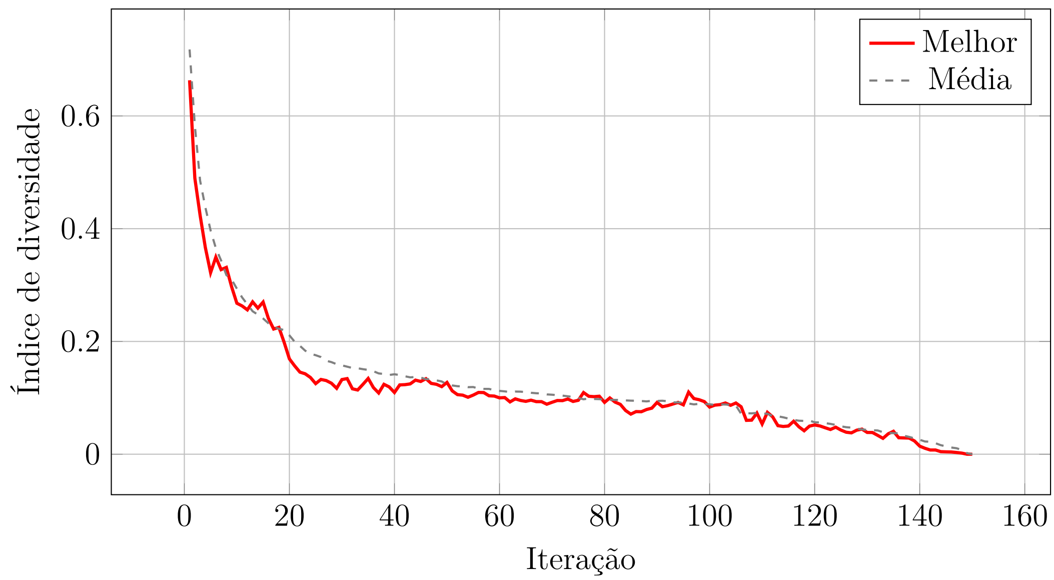
Figura 20: Curvas de convergência - Problema 2



Fonte: Autor

Na Figura 21 pode ser visualizada a diversidade da população ao longo da execução do método.

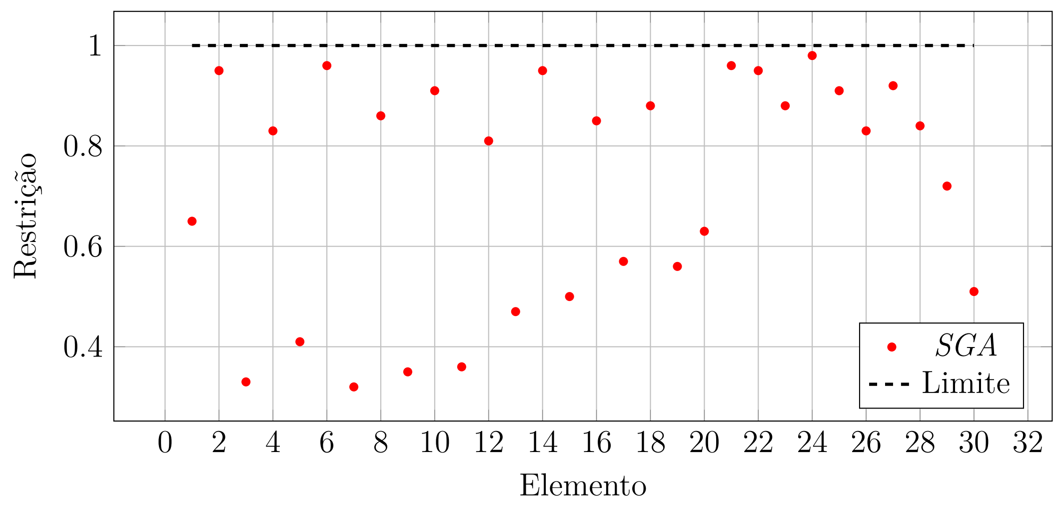
Figura 21: Diversidade da população - Problema 2



Fonte: Autor

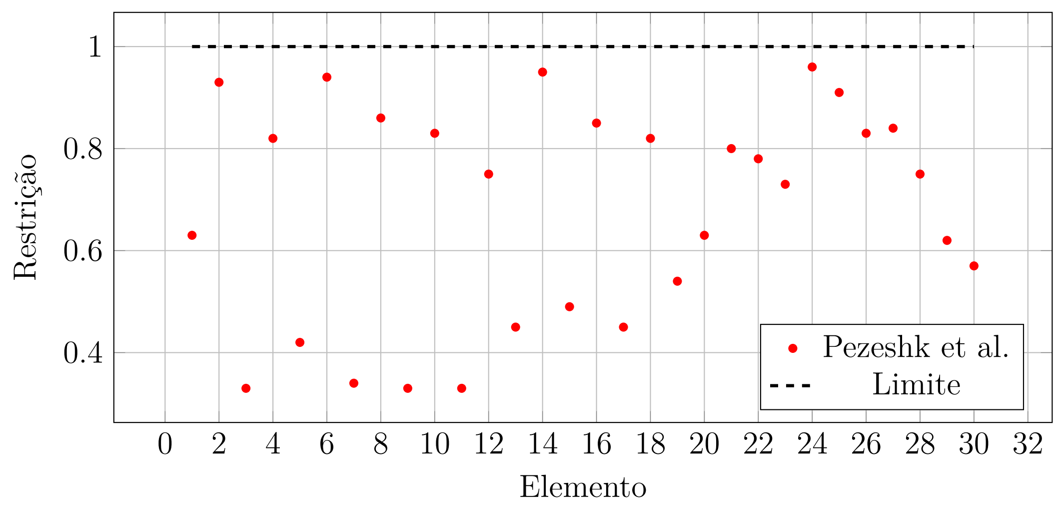
Para os exemplos 2 e 3 foram elaborados alguns gráficos a fim de avaliar o resultado das restrições impostas utilizando-se a função objetivo criada pelo autor, para os melhores resultados obtidos pelos autores da literatura. As restrições apresentadas referem-se àquelas normativas (Equação 6.1), sendo indicado o valor para cada um dos elementos (vigas e pilares). Nas Figuras 22, 23, 24, 25 e 26 podem ser visualizados os gráficos referentes ao Problema 2.

Figura 22: Restrição por elemento - Problema 2 - *SGA*



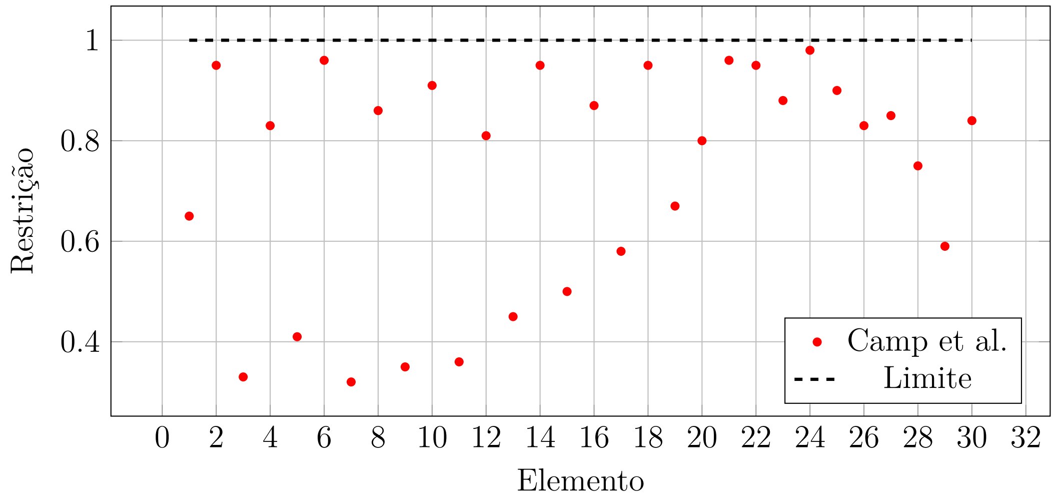
Fonte: Autor

Figura 23: Restrição por elemento - Problema 2 - Pezeshk



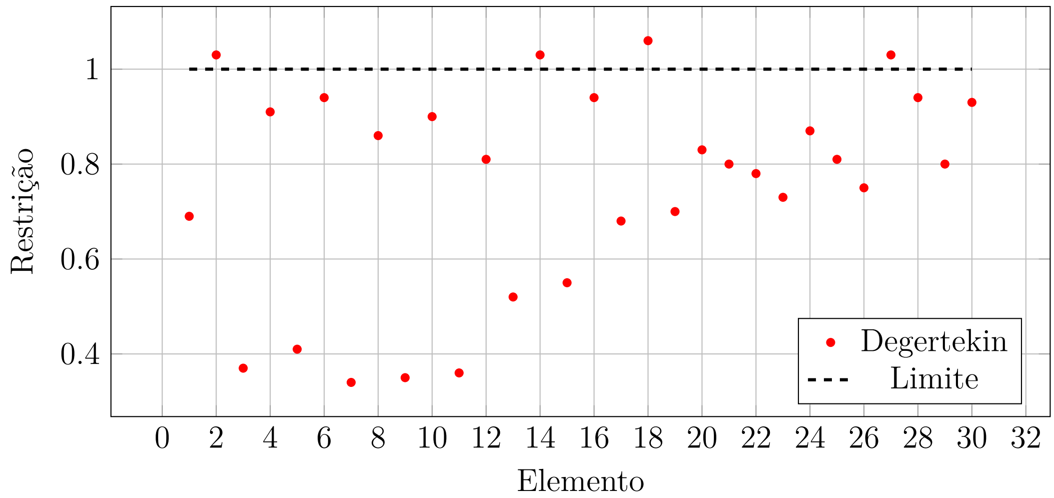
Fonte: Autor

Figura 24: Restrição por elemento - Problema 2 - Camp. et al



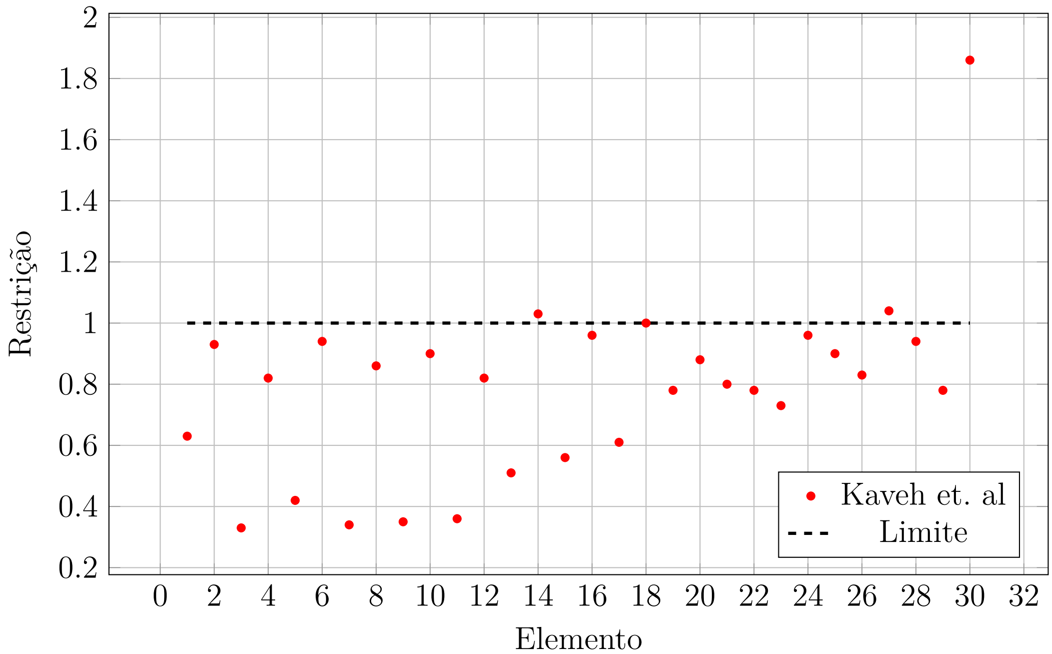
Fonte: Autor

Figura 25: Restrição por elemento - Problema 2 - Degertekin



Fonte: Autor

Figura 26: Restrição por elemento - Problema 2 - Kaveh et. al



Fonte: Autor

8.2.2 Discussão

Em relação ao problema anterior, que continha um espaço de busca de 4806 configurações estruturais, para este problema tem-se um espaço com $6,36 \times 10^{18}$ configurações de perfis válidas. Deste modo houve a necessidade de se aumentar o número máximo de avaliações da função antes da parada da rotina de modo que a mesma conseguisse explorar de forma satisfatória o domínio.

Pela curva de convergência da Figura 20 é possível notar uma queda acentuada nas primeiras iterações até a chegada em um patamar, à aproximadamente 70.000 lbs, onde a redução se torna mais gradual. Avaliando conjuntamente a Figura 21, percebe-se que a diversidade da população reduz rapidamente no início das iterações, decai mais lentamente no trecho do patamar, vindo a decair um pouco mais rapidamente com o início da etapa local, na iteração 106. Tal comportamento da diversidade se mostrou adequado para a otimização deste tipo de problema conforme verifica-se pela convergência dos resultados.

Observa-se nas Figuras 25 e 26 que para os perfis obtidos por Degertekin (2008) e Kaveh e Talatahari (2010a), ocorrem violações nas restrições de certos elementos, caracterizados pela presença de valores maiores que o limite. Desta forma, a comparação direta entre estes algoritmos fica prejudicada pois utilizando-se da mesma entrada, são obtidos diferentes valores da função objetivo. Tais resultados são descartados da análise pela rotina usada por serem considerados impraticáveis, uma vez que violam as restrições.

Desconsiderando os resultados inválidos para a rotina criada, pode-se dizer que foi obtido um bom resultado (63.534 lb), próximo ao melhor valor encontrado na literatura. Analisando a Figura 22 e a Tabela 5, verifica-se que foi obtido o segundo melhor resultado para a estrutura, ficando somente atrás da solução obtida por Camp et al. (2005), que por sua vez utilizou um número maior de avaliações da função objetivo. Em 100 execuções independentes foram reportadas, no entanto, melhores média e desvio padrão, 63.308 e 684, respectivamente.

De toda maneira, nas 50 execuções independentes do *SGA* realizadas, foi possível chegar consistentemente a valores abaixo da faixa de 70.000 lbs, como evidencia a média e o desvio padrão obtidos ($67.035 \pm 1940,8$ lb), conforme a Tabela 6.

8.3 Problema 3

8.3.1 Resultados

Para o último problema analisado utilizou-se como parâmetros do algoritmo: uma população total $n_{pop} = 50$, com um grupo de busca de $n_g = 10$ indivíduos, realizando um número de iterações $it = 200$. Para a etapa global utilizou-se $it_{global}^{max} = 140$. Referente às perturbações, utilizou-se $n_{perturb} = 5$. Para a aleatoriedade utilizou-se: $\alpha_0 = 0,14$ e $\alpha_{min} = 0,08$.

Na Tabela 7 estão dispostos os resultados de cada grupo de projeto obtidos pelos autores.

Tabela 7: Resultados para cada grupo - Problema 3

Nº do grupo	Perfis W						
	Camp et al. (2005)	Degertekin (2008)	Kaveh e Talatahari (2010a)	Kaveh e Talatahari (2010b)	Togan (2012)	Maheri e Narimani (2014)	***SGA* (este estudo)**
1	W30x90	W30x90	W30x99	W30x90	W30x90	W10x19	W24x68
2	W8x18	W10x22	W16x26	W21x50	W8x18	W12x190	W18x40
3	W24x55	W18x40	W18x35	W24x55	W24x62	W6x8.5	W24x55
4	W8x21	W12x16	W14x22	W8x28	W6x9	W24x370	W33x118
5	W14x145	W14x176	W14x145	W14x109	W14x132	W14x132	W14x159
6	W14x132	W14x176	W14x132	W14x159	W14x120	W14x30	W14x176
7	W14x132	W14x132	W14x120	W14x120	W14x99	W14x99	W14x99
8	W14x132	W14x109	W14x109	W14x90	W14x82	W14x53	W14x99
9	W14x68	W14x82	W14x48	W14x74	W14x74	W14x74	W14x74
10	W14x53	W14x74	W14x48	W14x68	W14x53	W14x26	W14x82
11	W14x43	W14x34	W14x34	W14x30	W14x34	W14x68	W14x30
12	W14x43	W14x22	W14x30	W14x38	W14x22	W14x193	W14x30
13	W14x145	W14x145	W14x159	W14x159	W14x109	W14x145	W14x109
14	W14x145	W14x132	W14x120	W14x132	W14x99	W14x26	W14x74
15	W14x120	W14x109	W14x109	W14x99	W14x99	W14x26	W14x99
16	W14x90	W14x82	W14x99	W14x82	W14x90	W14x43	W14x74
17	W14x90	W14x61	W14x82	W14x68	W14x68	W14x26	W14x68
18	W14x61	W14x48	W14x53	W14x48	W14x53	W14x120	W14x82
19	W14x30	W14x30	W14x38	W14x34	W14x34	W14x426	W14x99
20	W14x26	W14x22	W14x26	W14x22	W14x22	W14x68	W14x82
Peso (lb)	220.465	214.860	217.464	212.736	203.008	194.400	196.980
Nº de avaliações	15.500	14.651	3.500	7.500	12.000	1.259	8.000

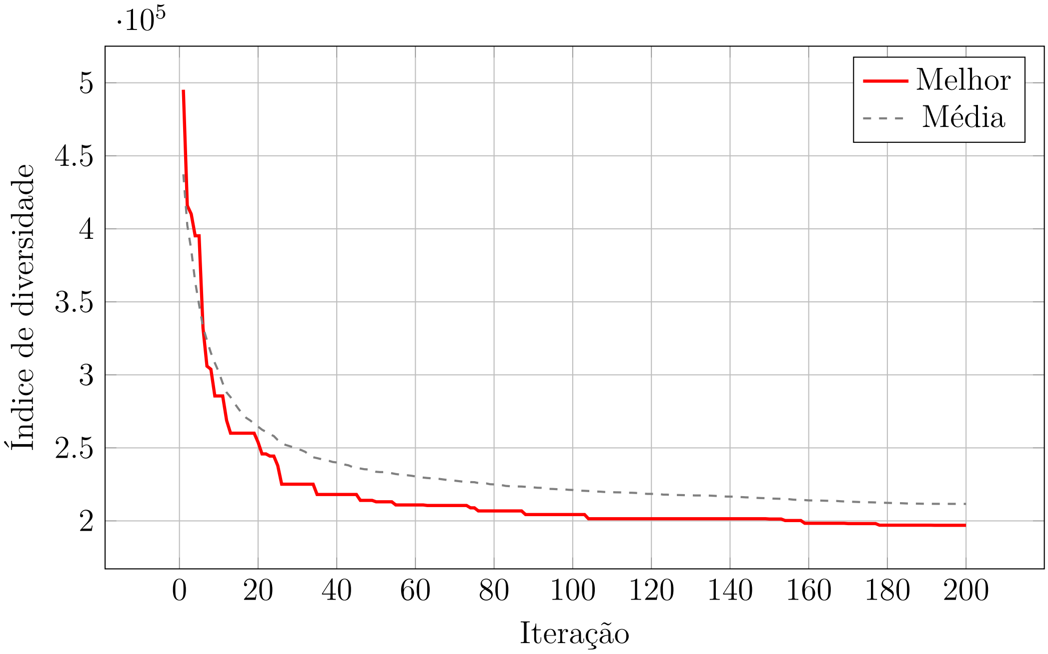
A partir de uma série de execuções independentes, obteve-se os resultados apresentados na Tabela 8.

Tabela 8: Resumo da série de testes - Problema 3

Número de execuções independentes do algoritmo	50
Mínimo observado	196.980 lb
Média	211.630 lb
Desvio padrão	6.649,9 lb

Na Figura 27 é possível visualizar a curva de convergência correspondente ao Problema 3.

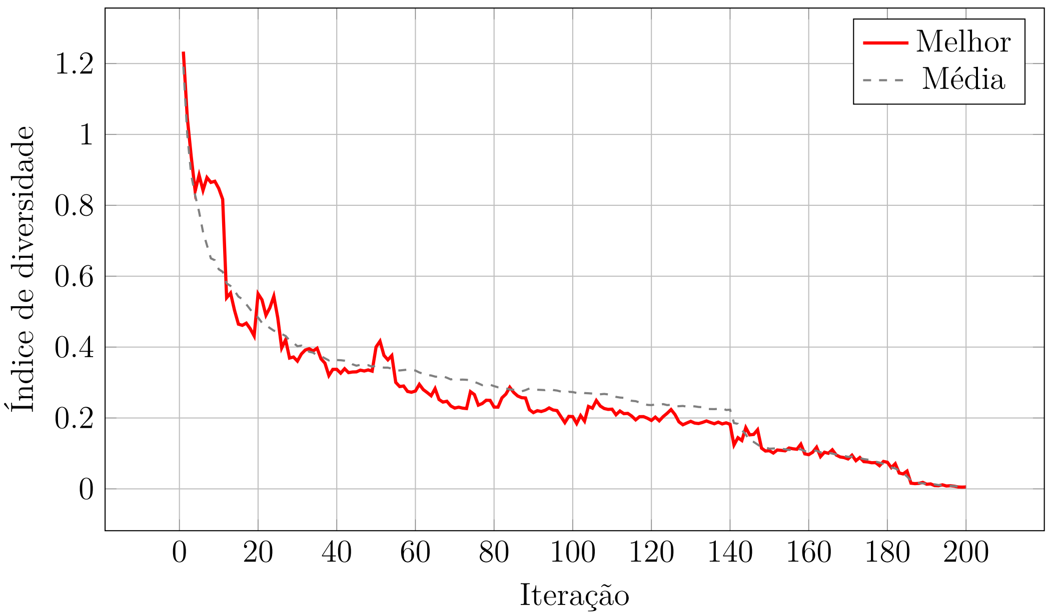
Figura 27: Curva de convergência - Problema 3



Fonte: Autor

A diversidade da população ao longo da execução do algoritmo pode ser observada na Figura 28.

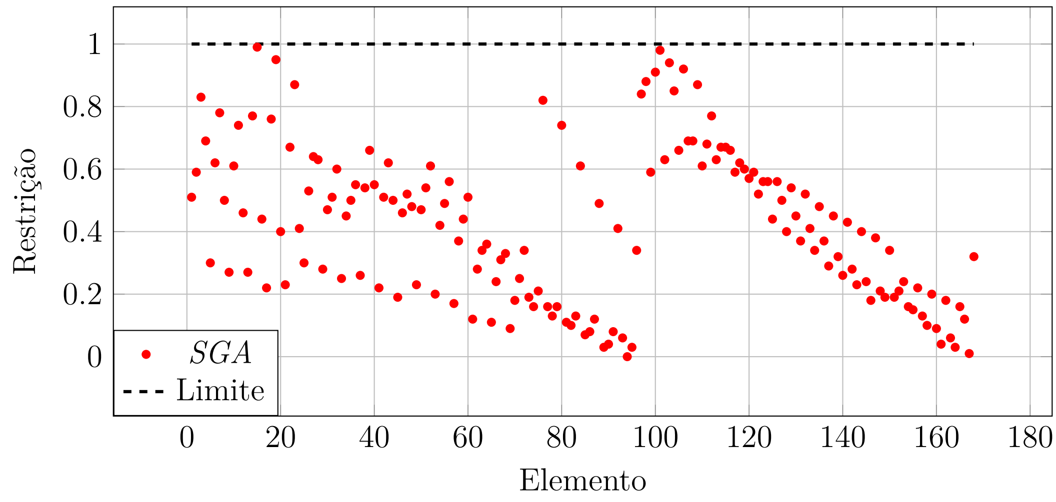
Figura 28: Diversidade da população - Problema 3



Fonte: Autor

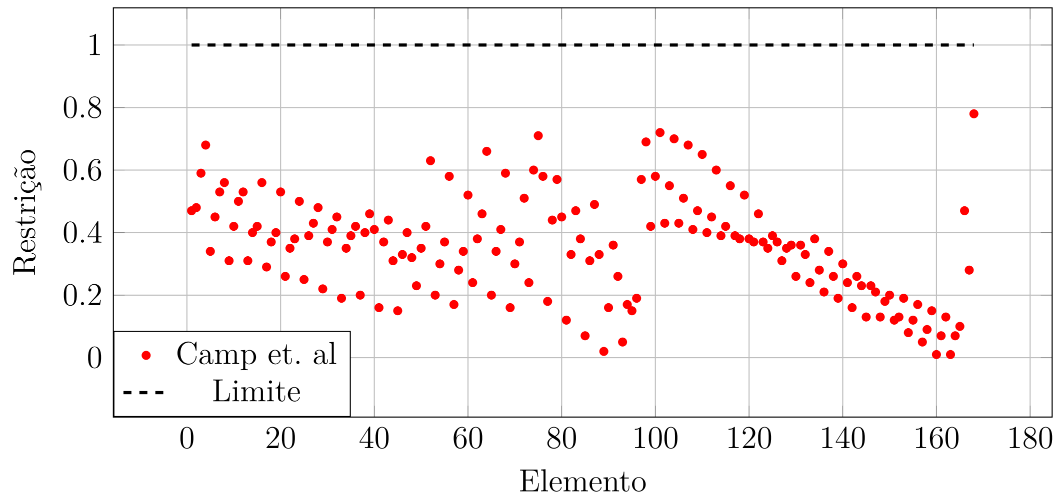
Nas Figuras 29, 30, 31, 32, 33, 34 e 35 estão representados os gráficos das restrições associadas a cada elemento, para o melhor valor encontrado por cada autor.

Figura 29: Restrição por elemento - Problema 3 - *SGA*



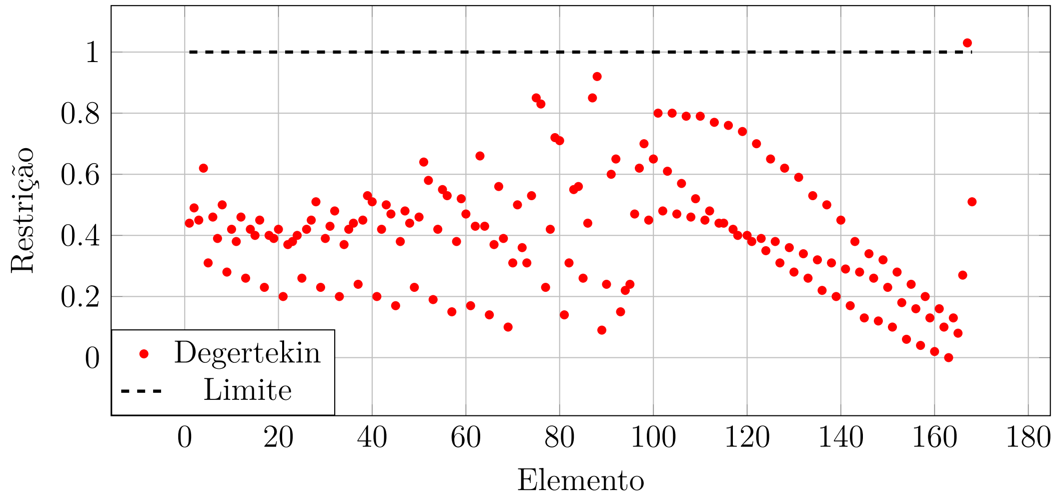
Fonte: Autor

Figura 30: Restrição por elemento - Problema 3 - Camp et. al



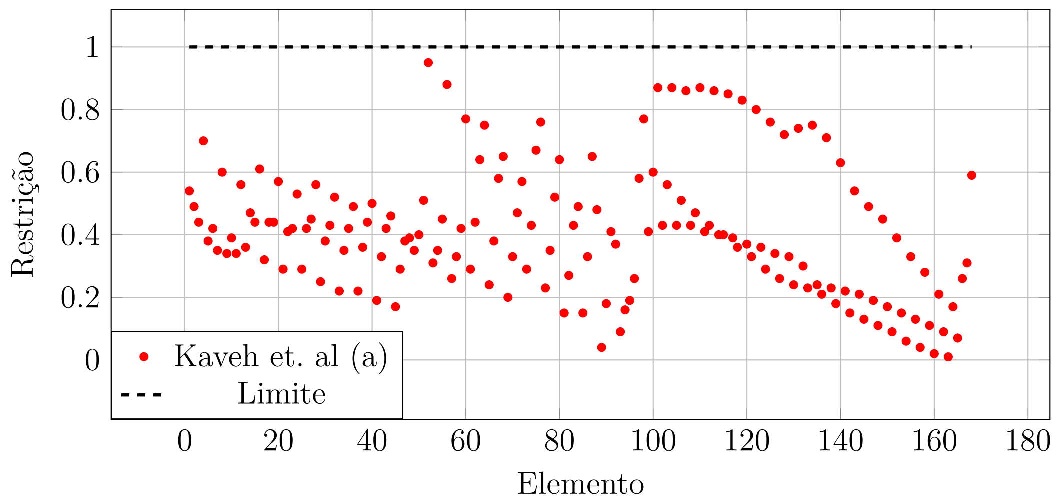
Fonte: Autor

Figura 31: Restrição por elemento - Problema 3 - Degertekin



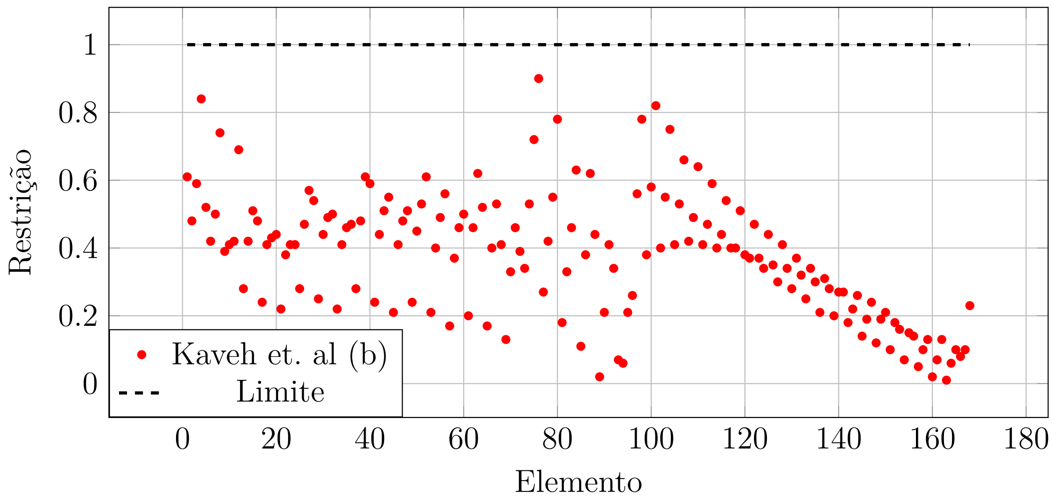
Fonte: Autor

Figura 32: Restrição por elemento - Problema 3 - Kaveh et. al (a)



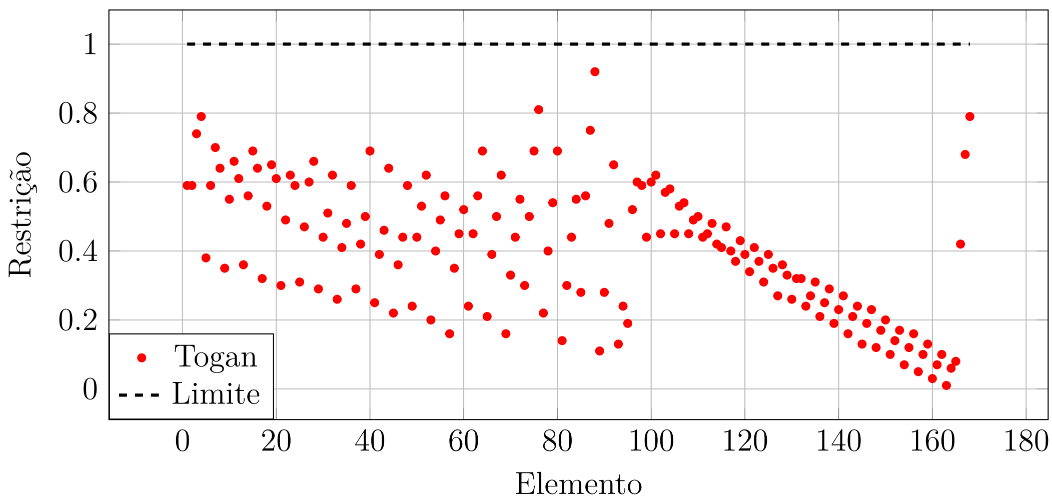
Fonte: Autor

Figura 33: Restrição por elemento - Problema 3 - Kaveh et. al (b)



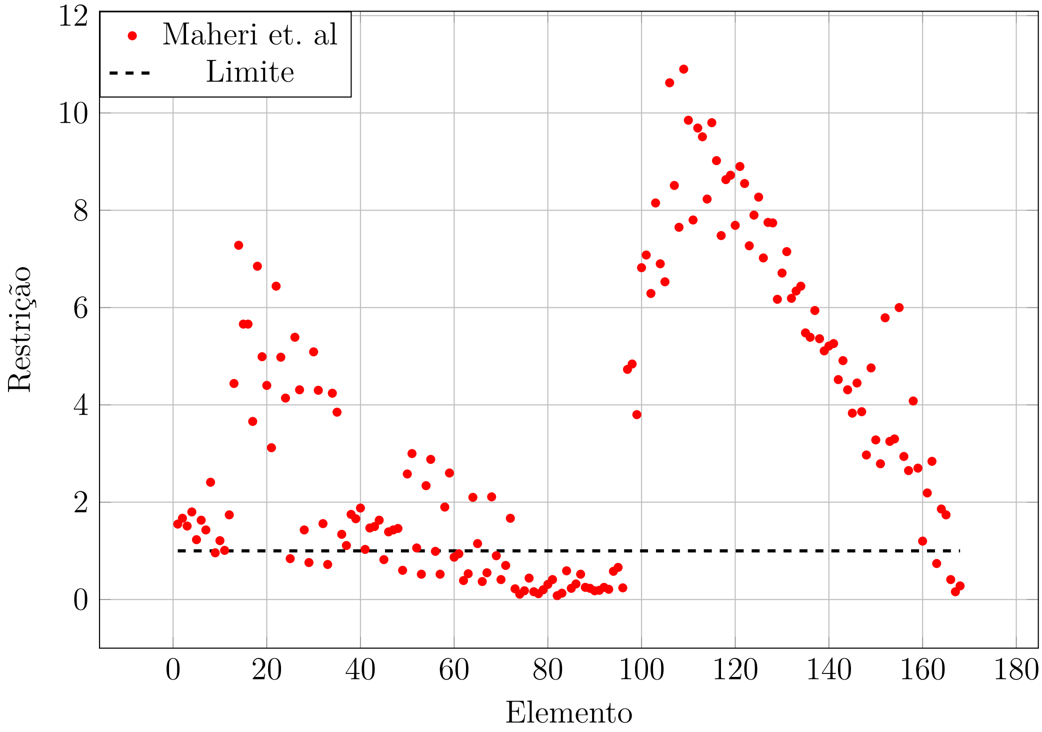
Fonte: Autor

Figura 34: Restrição por elemento - Problema 3 - Togan



Fonte: Autor

Figura 35: Restrição por elemento - Problema 3 - Maheri et. al



Fonte: Autor

8.3.2 Discussão

Para o terceiro e último problema analisado, obteve-se resultados convincentes quanto ao desempenho do algoritmo, principalmente quando se leva em consideração o fato deste ser o problema mais complexo, com maior número de variáveis envolvidas. Obteve-se o segundo menor peso da estrutura (196.980 lb), dentre os resultados analisados, para um número de avaliações de 8.000.

Observando-se a Tabela 7, verifica-se que somente o algoritmo proposto por Maheri e Narimani (2014) obteve um peso menor para um número também menor de avaliações. Entretanto, tal comparação não é justa uma vez que para a rotina desenvolvida, os resultados fornecidos pelo autor são altamente inconsistentes em relação aos limites impostos pela norma, conforme pode ser visualizado na Figura 35. Desta forma pode-se dizer que o resultado apresentado pelo *SGA* foi o melhor dentre aqueles cuja comparação direta foi possível, ou seja, dentre aqueles cujas restrições não ultrapassaram os limites impostos (Figuras 29, 30, 32 e 34).

O bom desempenho do algortimo neste problema pode ser observado também a partir da média obtida em 50 execuções do mesmo. Obteve-se um valor médio de 211.630 lb, valor este abaixo do mínimo obtido para quatro dentre os seis autores cujos resultados foram comparados.

Analisando-se a Figura 27, verifica-se que a curva de convergência apresenta o formato típico de uma curva de otimização, onde existe uma queda maior do objetivo nas iterações iniciais, porém havendo ainda um decréscimo do mesmo, mais suave, ao longo de todo o processo. Nota-se novamente, a partir da curva de diversidade (Figura 28), o decréscimo da mesma junto a interação 140, marcando o início da etapa local do algoritmo, no qual as famílias geradas não são mais avaliadas separadamente.

Conclusão

A partir da análise dos resultados é possível verificar que a adaptação do algoritmo *SGA* para o estudo de estruturas de pórtico foi realizada com sucesso. Obteve-se resultados competitivos comparando-se, quando possível, com as soluções de outros autores que utilizam algoritmos já estabelecidos na literatura, ou variações destes.

As modificações realizadas no algoritmo de modo a adaptá-lo para lidar com variáveis discretas, fez com o número de iterações necessárias para a convergência fosse reduzido, mantendo-se no mesmo nível de competitividade apresentado na otimização de problemas contínuos.

Obteve-se para o primeiro problema estudado o valor ótimo global, em um número pequeno de avaliações. Para o segundo problema obteve-se o segundo melhor resultado, desconsiderando os resultados inconsistentes frente a rotina utilizada, realizando-se um número menor de avaliações, porém com uma média e desvio padrão maior. Por fim, para o último e mais complexo problema, o algoritmo obteve o menor peso dentre todos os autores cuja comparação foi possível, fazendo uso de um número razoável de avaliações, além de apresentar na média valores menores que os mínimos reportados por alguns autores.

Tem-se que em geral, a otimização de pórticos planos foi bem atendida pelo uso do algoritmo *SGA*. Para futuros trabalhos, poderia-se avaliar o desempenho do algoritmo em estruturas mais complexas, como pórticos espaciais. Outra sugestão seria a adaptação do algoritmo à problemas multi-objetivo. Por fim, poderia ser feita uma avaliação de novas estratégias para lidar com as restrições, utilizando-se alguma alternativa que não excluísse resultados quen não atendessem as restrições, mas sim usasse-os para convergir mais rapidamente para o valor ótimo.

Referências

AMERICAN INSTITUTE OF STEEL CONSTRUCTION. *ANSI/AISC 360-10*: Specification for structural steel buildings. Chicago, 2010. Citado 8 vezes nas páginas 17, 31, 34, 35, 37, 38, 42 e 44.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 8800*: Projeto de estruturas de aço e de estruturas mistas de aço e concreto de edifícios - procedimento. Rio de Janeiro, 2008. Citado 5 vezes nas páginas 17, 31, 32, 35 e 37.

BARBARESCO, G. M. *Otimização de problemas de engenharia utilizando o algoritmo da Competição Imperialista (ICA)*. 64 f. Monografia (Trabalho de conclusão de curso) — Departamento de Engenharia Civil, Universidade Federal de Santa Catarina, Florianópolis, 2014. Citado na página 13.

CAMP, C. V.; BICHON, B. J.; STOVALL, S. P. Design of Steel Frames Using Ant Colony Optimization. *Journal of Structural Engineering*, v. 131, n. 3, p. 369–379, 2005. ISSN 0733-9445. Citado 5 vezes nas páginas 42, 48, 52, 57 e 58.

CARLON, A. G. *Otimização em treliças de estruturas metálicas aplicando o algoritmo ICA*. 74 f. Monografia (Trabalho de conclusão de curso) — Departamento de Engenharia Civil, Universidade Federal de Santa Catarina, Florianópolis, 2013. Citado na página 13.

DEB, K. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, v. 186, n. 2–4, p. 311 – 338, 2000. ISSN 0045-7825. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0045782599003898>. Citado na página 18.

DEGERTEKIN, S. Optimum design of steel frames using harmony search algorithm. *Structural and Multidisciplinary Optimization*, Springer-Verlag, v. 36, n. 4, p. 393–401, 2008. ISSN 1615-147X. Citado 6 vezes nas páginas 42, 48, 51, 52, 57 e 58.

DOGAN, E.; SAKA, M. Optimum design of unbraced steel frames to LRFD–AISC using particle swarm optimization. *Advances in Engineering Software*, v. 46, n. 1, p. 27–34, 2012. ISSN 09659978. Citado na página 42.

DUMONTEIL, P. Simple equations for effective length factors. *Engineering Journal*, v. 29, n. 3, p. 111–115, 1992. Citado 4 vezes nas páginas 41, 43, 44 e 46.

GALAMBOS, T.; SUROVEK, A. *Structural Stability of Steel: Concepts and Applications for Structural Engineers*. [S.l.]: Wiley, 2008. ISBN 9780470037782. Citado na página 35.

GARDNER, L. *Stability of Steel Beams and Columns: In Accordance with Eurocodes and the UK National Annexes*. Berkshire, UK: Beliveau Editeur, 2011. ISBN 9781859421994. Citado na página 36.

GONÇALVES, M. S.; LOPEZ, R. H.; MIGUEL, L. F. F. Search group algorithm: A new metaheuristic method for the optimization of truss structures. *Computers and Structures*, v. 153, p. 165–184, 2015. Citado na página 19.

KAVEH, A.; TALATAHARI, S. An improved ant colony optimization for the design of planar steel frames. *Engineering Structures*, Elsevier Ltd, v. 32, n. 3, p. 864–873, 2010. ISSN 01410296. Disponível em: <http://dx.doi.org/10.1016/j.engstruct.2009.12.012>. Citado 4 vezes nas páginas 42, 52, 57 e 58.

KAVEH, A.; TALATAHARI, S. Optimum design of skeletal structures using imperialist competitive algorithm. *Computers & Structures*, v. 88, n. 21–22, p. 1220 – 1229, 2010. ISSN 0045-7949. Citado na página 58.

KAVEH, A.; ZOLGHADR, A. Advances in Engineering Software Comparison of nine meta-heuristic algorithms for optimal design of truss structures with frequency constraints. *Advances in Engineering Software*, Elsevier Ltd, v. 76, p. 9–30, 2014. ISSN 0965-9978. Disponível em: <http://dx.doi.org/10.1016/j.advengsoft.2014.05.012>. Citado na página 49.

LEET, K. et al. *Fundamentos da análise estrutural*. 3. ed. São Paulo: McGraw-Hill, 2009. Citado na página 30.

LOGAN, D. *A First Course in the Finite Element Method, SI Version*. Stamford, US: Cengage Learning, 2011. ISBN 9780495668275. Citado na página 30.

MAHERI, M. R.; NARIMANI, M. An enhanced harmony search algorithm for optimum design of side sway steel frames. *Computers & Structures*, Elsevier Ltd, v. 136, p. 78–89, 2014. ISSN 00457949. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0045794914000558>. Citado 4 vezes nas páginas 45, 47, 58 e 64.

MEZURA-MONTES, E.; COELLO, C. A. C. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, v. 1, n. 4, p. 173 – 194, 2011. ISSN 2210-6502. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2210650211000538>. Citado na página 18.

MURREN, P.; KHANDELWAL, K. Design-driven harmony search (ddhs) in steel frame optimization. *Engineering Structures*, v. 59, n. 0, p. 798 – 808, 2014. ISSN 0141-0296. Citado na página 51.

PEZESHK, S.; CHEN, D. Design of nonlinear framed structures using genetic optimization. *Journal of Structural Engineering*, n. March, p. 382–388, 2000. Citado 4 vezes nas páginas 42, 48, 51 e 52.

PFEIL, W.; PFEIL, M. *Estruturas de aço : dimensionamento prático*. 8. ed. Rio de Janeiro: LTC, 2014. Citado na página 32.

RIBEIRO, L. A. D. *Otimização estrutural de treliças utilizando o algoritmo Firefly*. 114 f. Monografia (Trabalho de conclusão de curso) — Departamento de Engenharia Civil, Universidade Federal de Santa Catarina, Florianópolis, 2014. Citado na página 13.

SILVESTRE, N.; CAMOTIM, D. Elastic buckling and second-order behaviour of pitched-roof steel frames. *Journal of Constructional Steel Research*, v. 63, n. 6, p. 804 – 818, 2007. ISSN 0143-974X. Citado na página 31.

STEEL Construction Manual Shapes Database. 2015. Disponível em: <http://www.aisc.org/content.aspx?id=2868>. Acesso em: 05 de maio de 2015. Citado na página 43.

TOGAN, V. Design of planar steel frames using Teaching-Learning Based Optimization. *Engineering Structures*, 2012. ISSN 01410296. Citado 4 vezes nas páginas 42, 48, 51 e 58.

Apêndices

APÊNDICE A – Rotinas computacionais - Função objetivo

A.1 Análise do Pórtico

```
function [Pu,Mu,Cbs,secoes,vetorKx,n_pilares,restricao_desloc] = ...
    Portico_nt_lt(perfis)

% Número de nós e elementos
n_nos=22;
n_el=30;
% Número de cada nó e coordenada x
x  = [0 30 0 30 0 30 0 30 0 30 0 30 0 30 0 30 0 30 0 30 0 30]*12;
y  = [0 0 15 15 27 27 39 39 51 51 63 63 75 75 87 87 99 99 111 111 123 123]*12;

pilaresporpav = 2;
n_pilares = 20;
n_vigas  = n_el-n_pilares;

% Conectividade: [elemento   seção    nó_1    nó_2]
conec = [1     1     1     3
         2     1     2     4
         3     1     3     5
         4     1     4     6
         5     2     5     7
         6     2     6     8
         7     2     7     9
         8     2     8    10
         9     3     9    11
        10     3    10    12
        11     3    11    13
        12     3    12    14
        13     4    13    15
        14     4    14    16
        15     4    15    17
        16     4    16    18
        17     5    17    19
        18     5    18    20
        19     5    19    21
        20     5    20    22
```

Utilizando-se como exemplo a implementação do Problema 2

```
        21     6     3     4  %inicio das vigas
        22     6     5     6
        23     6     7     8
        24     7     9    10
        25     7    11    12
        26     7    13    14
        27     8    15    16
        28     8    17    18
        29     8    19    20
        30     9    21    22];

global W;
global W12e14;
E=29000; %200GPa
secoes = zeros(n_el,9);

%dicionário que avalia quais elementos estão conectados por cada nó
%ex. nó 3 conecta os elementos 1 3 e 21
%será usado para o cálculo de G de cada nó (função acharG)
el_conectados= containers.Map('KeyType','double', 'ValueType','any');

for el=1:n_el
    i_perfil = perfis(conec(el,2));
    %el i_perfil A E Ix rx ry L peso linear
    if el≤n_pilares
        secoes(el,:) = [el i_perfil W12e14(i_perfil,1) E W12e14(i_perfil,9) ...
        W12e14(i_perfil,12) W12e14(i_perfil,16) 0 W12e14(i_perfil,21)];
    else
        secoes(el,:) = [el i_perfil W(i_perfil,1) E W(i_perfil,9) ...
        W(i_perfil,12) W(i_perfil,16) 0 W(i_perfil,21)];
    end
    %será usado para o cálculo de G (função acharG)
    if isKey(el_conectados,conec(el,3))
        el_conectados(conec(el,3)) = [el_conectados(conec(el,3)) el];
    else
        el_conectados(conec(el,3)) = el;
    end
    if isKey(el_conectados,conec(el,4))
        el_conectados(conec(el,4)) = [el_conectados(conec(el,4)) el];
    else
        el_conectados(conec(el,4)) = el;
    end
end

% Carregamentos, forças=[nó   intensidade_x  intensidade_y   M_z]
n_forcas=10;
forcas=[ 3    10     0     0
```

```
          5    10     0     0
          7    10     0     0
          9    10     0     0
         11    10     0     0
         13    10     0     0
         15    10     0     0
         17    10     0     0
         19    10     0     0
         21     5     0     0];

% Carregamentos equivalentes=[elemento  tipo  intensidade]
n_eq=10;
w_eq=[ 21    1    6/12
       22    1    6/12
       23    1    6/12
       24    1    6/12
       25    1    6/12
       26    1    6/12
       27    1    6/12
       28    1    6/12
       29    1    6/12
       30    1    3/12];

% Apoios
n_rest=2;  %Numero de nós restringidos
GDL_rest=[1   1   1   1
          2   1   1   1];  %[nó   restringido_x   restringido_y   restringido_theta]
                           %(1 para restringido, 0 para livre)

% CALCULO DA ESTRUTURA
GDL=3*n_nos;  %graus de liberdade da estrutura
K=zeros(GDL,GDL); %matriz rigidez global

% Cálculo da matriz de cada elemento
for el=1:n_el
    %calculo do comprimento do elemento el
    no1=conec(el,3);
    no2=conec(el,4);
    %L=abs(x(no2)-x(no1));
    L = sqrt((x(no2) - x(no1))^2 + (y(no2) - y(no1))^2);
    secoes(el,8)=L;
    %Propriedades
    A = secoes(el,3);
    E = secoes(el,4);
    Iz = secoes(el,5);
    %Cossenos diretores a partir das coordenadas dos nós do elemento
    c = (x(no2) - x(no1))/L;    % cosseno
```

```
    s = (y(no2) - y(no1))/L;    %  seno
    % Matriz de rotação do elemento "el"
    T=[c  s  0  0  0  0
      -s  c  0  0  0  0
       0  0  1  0  0  0
       0  0  0  c  s  0
       0  0  0 -s  c  0
       0  0  0  0  0  1];
       % Construção da matriz de rigidez em coordenadas locais
       k1=E*A/L;
       k2=12*E*Iz/L^3;
       k3=6*E*Iz/L^2;
       k4=4*E*Iz/L;
       k5=k4/2;
       k=[k1   0    0   -k1   0   0
           0   k2   k3   0   -k2  k3
           0   k3   k4   0   -k3  k5
         -k1   0    0    k1   0   0
           0  -k2  -k3   0   k2  -k3
           0   k3   k5   0  -k3   k4];
       % Matriz de rigidez em coordenadas globais
       kg=T'*k*T;

      %Determinando matriz de incidência cinemática:
      b = zeros(6,GDL);
      i=no1;
      j=no2;
      b(1,3*(i-1)+1) = 1;
      b(2,3*(i-1)+2) = 1;
      b(3,3*(i-1)+3) = 1;
      b(4,3*(j-1)+1) = 1;
      b(5,3*(j-1)+2) = 1;
      b(6,3*(j-1)+3) = 1;
      %Expandindo e convertendo a matriz do elemento para coordenadas globais:
      Ki = b'*kg*b;
      %Somando contribuição do elemento para a matriz de rigidez global:
      K = K + Ki;
end


%% Separação da estrutura em lt e nt (com e sem deslocamento lateral)
% calcula esforços considerando somente cargas em y
% depois recalcula considerando somente os esforços laterais
% isto para posteriormente poder calcular os fatores de amplificação para a
% consideração dos efeitos de segunda ordem

% Vetor de forças Global
```

```
% aqui tem-se as forças aplicadas lateralmente

F=zeros(GDL,1);
for i=1:n_forcas
    F(3*forcas(i,1)-2)=forcas(i,2);
    F(3*forcas(i,1)-1)=forcas(i,3);
    F(3*forcas(i,1))=forcas(i,4);
end
% Construção do vetor de forças equivalentes
% aqui tem-se os carregamentos distribuidos verticais que não causam
% deslocamento lateral da estrtura

Feq=zeros(GDL,1);
for i=1:n_eq
    tipo=w_eq(i,2);   %tipo de força equivalente
    el=w_eq(i,1);     %elemento onde está aplicada
    if tipo==1
        f=zeros(6,1);
        no1=conec(el,3);
        no2=conec(el,4);
        L = secoes(el,8);
        w=w_eq(i,3);
        f(1)=0;
        f(2)=-w*L/2;
        f(3)=-w*L^2/12;
        f(4)=0;
        f(5)=-w*L/2;
        f(6)=+w*L^2/12;
        %Cossenos diretores a partir das coordenadas dos nós do elemento
        c = (x(no2) - x(no1))/L;    % cosseno
        s = (y(no2) - y(no1))/L;    %  seno
        % Matriz de rigidez do elemento "el"
        T=[c  s  0  0  0  0;
          -s  c  0  0  0  0
           0  0  1  0  0  0
           0  0  0  c  s  0
           0  0  0 -s  c  0
           0  0  0  0  0  1];
        feq=T'*f;
        Feq(3*no1-2)=Feq(3*no1-2)+feq(1);
        Feq(3*no1-1)=Feq(3*no1-1)+feq(2);
        Feq(3*no1)=Feq(3*no1)+feq(3);
        Feq(3*no2-2)=Feq(3*no2-2)+feq(4);
        Feq(3*no2-1)=Feq(3*no2-1)+feq(5);
        Feq(3*no2)=Feq(3*no2)+feq(6);
    end
end
```

```

% guardamos os originais de K e F
% aqui nota-se que a matriz de rigidez continua a mesma, porém
% os vetores de força são separados em nt e lt (no lateral e lateral
% translation)

Kg=K;

Fg_lt=F;
Fg_nt=Feq;

% Aplicar Restrições (condições de contorno)
for k=1:n_rest
    % Verifica se há restrição na direção x
    if GDL_rest(k,2)==1
        j=3*GDL_rest(k,1)-2;
        %Modificar Matriz de Rigidez
        for i=1:GDL
            Kg(j,i)=0;   %zera linha
            Kg(i,j)=0;   %zera coluna
        end
        Kg(j,j)=1;       %valor unitário na dianogal principal
        Fg_lt(j)=0;
        Fg_nt(j)=0;
    end
    % Verifica se há restrição na direção y
    if GDL_rest(k,3)==1
        j=3*GDL_rest(k,1)-1;
        %Modificar Matriz de Rigidez
        for i=1:GDL
            Kg(j,i)=0;   %zera linha
            Kg(i,j)=0;   %zera coluna
        end
        Kg(j,j)=1;       %valor unitário na dianogal principal
        Fg_lt(j)=0;
        Fg_nt(j)=0;
    end
    % Verifica se há restrição na rotação
    if GDL_rest(k,4)==1
        j=3*GDL_rest(k,1);
        %Modificar Matriz de Rigidez
        for i=1:GDL
            Kg(j,i)=0;   %zera linha
            Kg(i,j)=0;   %zera coluna
        end
        Kg(j,j)=1;       %valor unitário na dianogal principal
        Fg_lt(j)=0;
```

```
        Fg_nt(j)=0;
    end
end

% Calculo dos deslocamentos
desloc_lt = Kg\Fg_lt;
desloc_nt = Kg\Fg_nt;

% Reações
reacoes_lt=K*desloc_lt-Fg_lt;
reacoes_nt=K*desloc_nt-Fg_nt;

% Esforços nos elementos
f_el_nt=zeros(el,6);
f_el_lt=zeros(el,6);

%Fator de segunda ordem B1
B1 = zeros(1,el);

for el=1:n_el
    %calculo do comprimento do elemento el
    no1=conec(el,3);
    no2=conec(el,4);
    %L=abs(x(no2)-x(no1));
    L = secoes(el,8);
    %Propriedades
    A = secoes(el,3);
    E = secoes(el,4);
    Iz = secoes(el,5);
    %Cossenos diretores a partir das coordenadas dos nós do elemento
    c = (x(no2) - x(no1))/L;    % cosseno
    s = (y(no2) - y(no1))/L;    %  seno
    % Matriz de rigidez do elemento "el"
    T=[ c  s  0  0  0  0
       -s  c  0  0  0  0
        0  0  1  0  0  0
        0  0  0  c  s  0
        0  0  0 -s  c  0
        0  0  0  0  0  1];
    % Construção da matriz de rigidez em coordenadas locais
    k1=E*A/L;
    k2=12*E*Iz/L^3;
    k3=6*E*Iz/L^2;
    k4=4*E*Iz/L;
    k5=k4/2;
    ke=[k1   0    0   -k1   0   0
        0   k2   k3   0   -k2  k3
```

```
        0   k3   k4   0   -k3  k5
        -k1   0    0    k1   0   0
        0  -k2  -k3   0   k2  -k3
        0   k3   k5   0  -k3   k4];
    %pega os valores dos deslocamentos dos nós do elemento "el"
    u1_nt = desloc_nt(no1*3-2);
    u2_nt = desloc_nt(no2*3-2);
    v1_nt = desloc_nt(no1*3-1);
    v2_nt = desloc_nt(no2*3-1);
    th1_nt= desloc_nt(no1*3);
    th2_nt= desloc_nt(no2*3);
    d_g_nt=[u1_nt  v1_nt  th1_nt  u2_nt  v2_nt  th2_nt]';
    d_el_nt=T*d_g_nt;

    u1_lt = desloc_lt(no1*3-2);
    u2_lt = desloc_lt(no2*3-2);
    v1_lt = desloc_lt(no1*3-1);
    v2_lt = desloc_lt(no2*3-1);
    th1_lt= desloc_lt(no1*3);
    th2_lt= desloc_lt(no2*3);
    d_g_lt=[u1_lt  v1_lt  th1_lt  u2_lt  v2_lt  th2_lt]';
    d_el_lt=T*d_g_lt;

    %% forças equivalentes: recalcula vetor de feq. no sistema local
    aux=find(w_eq(:,1)==el);
    if isempty(aux)
        feqq=0;
    else
        tipo=w_eq(aux,2);   %tipo de força equivalente
        if tipo==1
            w=w_eq(aux,3);
            feqq=zeros(6,1);
            feqq(1)=0;
            feqq(2)=-w*L/2;
            feqq(3)=-w*L^2/12;
            feqq(4)=0;
            feqq(5)=-w*L/2;
            feqq(6)=+w*L^2/12;
        end
    end

   %% força e tensão atuante no elemento "el", cada linha da matriz f_el
    %contem os esforços de um elemento = [Fx_1 Fy_1 Mz_1 Fx_2 Fy_2 Mz_2]
    % os esforços são armazenados
    f_el_nt(el,:) = ke*d_el_nt-feqq;
    f_el_lt(el,:) = ke*d_el_lt;

```

```
    %% cálculo do fator de segunda ordem B1

    Pe1= pi^2*E*Iz/L^2;
    m1 = f_el_nt(el,3);
    m2 = f_el_nt(el,6);
    if abs(m2) > abs(m1)
        cm = 0.6 + 0.4*m1/m2;
    else
        cm = 0.6 + 0.4*m2/m1;
    end
    B1(el) = cm/(1-f_el_nt(el,4)/Pe1);
end
% B1 ≥ 1
B1 = max(1,B1);

%% cálculo do fator de segunda ordem B2

%pega os valores de desloc. na extremidade direita e subtrai 2 a 2 cada nó
auxiliar = reshape(desloc_lt,[3,n_nos]);
interstorydrift = diff(auxiliar(1,pilaresporpav:pilaresporpav:end));

%soma das forcas verticais, nos pilares, para cada andar
soma_nd = w_eq(:,3).*secoes(n_pilares+1:end,8);

%soma das forcas em x para cada andar, pega a coluna pq só tem 1 força por
%andar
soma_hd = forcas(:,2);

%B2 calculado usando a soma das cargas verticais e horizontais, o desvio
%entre pavimentos,e a altura dos pavimentos(y(4))
h_andares = diff(y(pilaresporpav:pilaresporpav:end));
B2 = 1./(1 - 1./(0.85*h_andares).*interstorydrift.*(soma_nd./soma_hd)');

restricao_desloc = sum((interstorydrift ./ (h_andares/300))-1);

%% função para cálculo de G em cada nó (cálculo de Kx)
function G = acharG(vetor)
    if numel(vetor)==1
        G= 10;
    else
        ehpilar = vetor(vetor ≤ n_pilares);
        ehviga  = vetor(vetor >  n_pilares);
        Ip = secoes(ehpilar,5);
        Lp = secoes(ehpilar,8);
        Iv = secoes(ehviga,5);
        Lv = secoes(ehviga,8);
        G = sum(Ip./Lp)/sum(Iv./Lv);
```

```
    end
end

vetorG = zeros(1,n_nos);
for no=1:n_nos
    conectados = el_conectados(no);
    vetorG(no) = acharG(conectados);
end

%% construção dos vetores Mu, Pu consdierando segunda ordem além do vetorKx

%B1*Mnt
%transforma o B1 em duas colunas para multiplicar o momento no topo e base de ...
    cada elm
B1 = [B1;B1]';
f_el_nt(:,[3,6]) = B1 .* f_el_nt(:,[3,6]);

%B2*Mlt e B2*Nlt
%ajuste de B2 transformando em 4 colunas para multiplicar tp e bs de M e N
B2 = repmat(B2,[pilaresporpav,1]);
B2 = B2(:);
B2 = repmat(B2,[1,4]);
f_el_lt((1:n_pilares),[1,3,4,6]) = B2 .* f_el_lt((1:n_pilares),[1,3,4,6]);

f_el_2ord = f_el_lt + f_el_nt;

%pega o valor de compressão, coluna 4 dá o sinal correto do esforço,
%positivo = tração e negativo = compressão
Pu = f_el_2ord(:,4); %unidade kip


function kx = acharkx (Ga,Gb)
    kx = sqrt((1.6*Ga*Gb+4*(Ga+Gb)+7.5)/(Ga+Gb+7.5));
end

vetorKx = zeros(n_el,1);

Cbs = zeros(1,n_el);
despreza2ord = all(B2<1.1);
M_vao = zeros(n_el,1);
for i=1:n_el
    if i≤n_pilares %pilares - momento linear
        Mi = f_el_2ord(i,3);
        Mf = f_el_2ord(i,6);
        Mb = (Mi+Mf)/2;
        Ma = (Mi+Mb)/2;
        Mc = (Mb+Mf)/2;
```

```
        Mmax = max(abs(Mi),abs(Mf));
        Cbs(i) = 12.5*Mmax/(2.5*Mmax+3*abs(Ma)+4*abs(Mb)+3*abs(Mc));
        %se 2ord < 1.1 os pilares ficam com K=1
        if despreza2ord
            vetorKx(i) = 1;
        else
            vetorKx(i) = acharkx(vetorG(conec(i,3)),vetorG(conec(i,4)));
        end
    else %vigas - momento quadrático
        Mi = f_el_2ord(i,3);
        Mf = abs(f_el_2ord(i,6));
        carga_q = w_eq(w_eq(:,1)==i,3);
        %momento no vão calculado pelo cortante M1 - V^2/(2*q)
        M_vao(i) = abs(Mi - f_el_2ord(i,2)^2/carga_q/2);
        Ma = abs(Mi-f_el_2ord(i,2)*secoes(el,8)*0.25 + ...
            carga_q/2*(secoes(el,8)*0.25)^2);
        Mb = abs(Mi-f_el_2ord(i,2)*secoes(el,8)*0.5 + ...
            carga_q/2*(secoes(el,8)*0.5)^2);
        Mc = abs(Mi-f_el_2ord(i,2)*secoes(el,8)*0.75 + ...
            carga_q/2*(secoes(el,8)*0.75)^2);
        Mmax = max([Mi,Mf,M_vao(i)]);
        Cbs(i) = 12.5*Mmax/(2.5*Mmax+3*abs(Ma)+4*abs(Mb)+3*abs(Mc));
        vetorKx(i) = acharkx(vetorG(conec(i,3)),vetorG(conec(i,4)));
    end
end

%pega o máximo entre os momentos de cada barra, colunas 3 e 6 correspondem
%as colunas do momento [fx1,fy1,{M1},fx2,fy2,{M2}]
Mu = max(abs(f_el_2ord(:,[3,6])),[],2);
%pega o máximo entre momentos extremidade e vão
Mu = max(Mu,M_vao);
end
```

A.2 Cálculo das resistências

```
function [Pn,Mn,resist_tracao] = esforcos_nominais(secoes,Cbs,vetorKx,n_pilares)
E=29000; %200GPa
Fy=36;   %248.2MPa
refy = sqrt(E/Fy);

global W;
global W12e14;



n_el = numel(Cbs);
n_vigas = n_el - n_pilares;
```

```

% itera todos os pilares e vigas
% e computa a resistência dos elementos
%usa uma função que recebe o indice e calcula a resistência
%retorna:
% Pn = [Pn_pilar Pn_viga];
% Mn = [Mn_pilar Mn_viga];

%% Cálculo da resistência à tração
resist_tracao = secoes(:,3)*Fy;

%% recebe a seção do pilar e da viga e calcula o kx de cada elem.

ky_pilares = 1;
ky_vigas   = 0.2;
%constroi vetorKy para ky constante entre vigas e pilares
vetorKy = [ky_pilares*ones(n_pilares,1);ky_vigas*ones(n_vigas,1)];

%verfica a máxima relação KL/r entre eixo X e Y
klr = max(vetorKx./secoes(:,6), vetorKy./secoes(:,7)).*secoes(:,8);

function [Pn,Mn] = achar_resistencia(indice, klr, L, ky, Cb,el)
   %% Propriedades da seção
   if el≤n_pilares
        A   = W12e14(indice,1);
        d   = W12e14(indice,2);
        tw  = W12e14(indice,3);
        bf  = W12e14(indice,4);
        tf  = W12e14(indice,5);
        W_p = W12e14(indice,6);
        bt  = W12e14(indice,7);
        htw = W12e14(indice,8);
        Ix  = W12e14(indice,9);
        Zx  = W12e14(indice,10);
        Sx  = W12e14(indice,11);
        rx  = W12e14(indice,12);
        Iy  = W12e14(indice,13);
        Zy  = W12e14(indice,14);
        Sy  = W12e14(indice,15);
        ry  = W12e14(indice,16);
        J   = W12e14(indice,17);
        Cw  = W12e14(indice,18);
        h0  = W12e14(indice,19);
        rts = W12e14(indice,20);

   else
        A   = W(indice,1);
```

```
        d   = W(indice,2);
        tw  = W(indice,3);
        bf  = W(indice,4);
        tf  = W(indice,5);
        W_p = W(indice,6);
        bt  = W(indice,7);
        htw = W(indice,8);
        Ix  = W(indice,9);
        Zx  = W(indice,10);
        Sx  = W(indice,11);
        rx  = W(indice,12);
        Iy  = W(indice,13);
        Zy  = W(indice,14);
        Sy  = W(indice,15);
        ry  = W(indice,16);
        J   = W(indice,17);
        Cw  = W(indice,18);
        h0  = W(indice,19);
        rts = W(indice,20);
   end

    %% Compressão - sem elementos esbeltos
    % cálculo padrão
    % usado para o caso de elementos esbeltos

    Fe = pi^2*E/klr^2;
    if klr ≤ 4.71*refy
        Fcr = 0.658^(Fy/Fe)*Fy;
    else
        Fcr = 0.877*Fe;
    end

    %% Compressão - com elementos esbeltos
    % verificação da esbeltez na compressão da alma
    % nota: todas as mesas da série W são compactas para compressão
    Qa=1;
    if  htw > 1.49*refy   % h/tw > lim alma não compacta
        ref = sqrt(E/Fcr);
        if htw ≥ 1.49*ref
            be = 1.92 * tw * ref*(1- 0.34/htw*ref); %nova alma ->Hnovo
            if be < htw*tw
                Aeff = A - tw*(htw*tw - be);  % novoA = velhoA - tw*redução
                Qa = Aeff/A;
            end
        end
    end
```

```
%% Cálculo de Pn
% alteração de Fcr com base na esbeltez

if Qa==1
    Pn = Fcr*A;
else
    if Fe ≥ 0.44*Qa*Fy
        Fcr = Qa*(0.658^(Qa*Fy/Fe))*Fy;
    else
        Fcr = 0.877*Fe;
    end
    Pn = Fcr*A;
end

%% Flexão
% nota: somente W6x15 tem mesa não-compacta na série W
% considerando fy=36ksi
% todos os outros tem mesa compacta na flexão
% além disso, todas as almas são compactas na flexão
% resultando em dois casos:
% 1) alma e mesa compactas -- F2.[1,2]
% 2) alma compacta e mesa não compacta (W6x15) -- F2.2 e F3.2

% Caso 1 -- alma e masa compactas
% a) escoamento da seção bruta

Mp = Zx*Fy;

% b) flambagem lateral com torção (FLT)
Lb = ky*L; %considera o contraventamento
    Lp = 1.76*ry*refy; %resultado em polegadas
Lr = 1.95*rts*E/(0.7*Fy)*sqrt(J/(Sx*h0) + sqrt( (J/(Sx*h0))^2 + ...
    6.76*(0.7*Fy/E)^2));

if Lb ≤ Lp
    Mn = Mp;
elseif (Lp < Lb) && (Lb ≤ Lr)
    Mn = Mp - (Mp - 0.7*Fy*Sx)*(Lb-Lp)/(Lr-Lp);
else
    Fcr = Cb*pi^2*E/(Lb/rts)^2*sqrt(1+0.078*J/(Sx*h0)*(Lb/rts)^2);
    Mn = Fcr*Sx;
end

% Caso 2 -- alma compacta e mesa não compacta (W6x15)
if indice == 6
    lamb_p = 0.38*refy;
    lamb_r = refy;
```

```
        Mn = Mp - (Mp-0.7*Fy*Sx)*(bt-lamb_p)/(lamb_r-lamb_p);
    end

    % Mn nunca pode ser maior que Mp (escoamento da seção bruta)
    if Mn > Mp
        Mn = Mp;
    end
end

%pré-alocação do tamanho do vetor para melhorar a velocidade
Pn = zeros(1,n_el);
Mn = zeros(1,n_el);

%cálculo das resistências Pn e Mn de cada elemento
for el=1:n_el
    [Pn(el),Mn(el)] = ...
        achar_resistencia(secoes(el,2),klr(el),secoes(el,8),vetorKy(el),Cbs(el),el);
end
end
```

A.3 Função objetivo

```
function [ obj ] = fobj_portico(perfis)
% função objetivo recebe índices da viga e pilar:
% retorna peso se ok ou então peso + penalidade
perfis = round(perfis);

%% Esforços requeridos
% chama código de pórtico plano e retorna com a matriz de forças e momentos
% em cada elemento do pórtico

[Pu,Mu,Cbs,secoes,vetorKx,n_pilares,restricao_desloc] = Portico_nt_lt(perfis);

%% Esforços nominais - resistência
% computa a resistência do perfil
% depende de KL/r, logo varia para cada elemento

[Pr,Mr,resist_tracao] = esforcos_nominais(secoes,Cbs,vetorKx,n_pilares);
Pr = Pr';
Mr = Mr';

% a função portico_nt_lt calcula todos os elementos como flexocompressão,
% caso o esforço em algum elemento seja de tração, este "for" coloca a
% resistência do perfil correspondente
for i=find(Pu>0)
    Pr(i) = resist_tracao(i);
```

```
end

% feita a consideração dos sinais (Tr/Comp), pode-se pegar o valor em
% módulo apenas
Pu = abs(Pu);

n_el=numel(Pu);
interacao = zeros(1,n_el);
for i=1:n_el
    UsobreR = Pu(i)/(0.9*Pr(i));
    if UsobreR ≥ 0.2
        interacao(i) = fix((UsobreR + 8/9*(Mu(i)/(0.9*Mr(i))))*100)/100;
    else
        interacao(i) = fix((UsobreR/2 + Mu(i)/(0.9*Mr(i)))*100)/100;
    end
end

%peso = soma de peso linear * L
peso = sum(secoes(:,9).*secoes(:,8)/12);

% resultado da interação de flexo-compressão, deve ser menor que 1
% 0 para ok ou o quanto excedeu de 1.0
restricao1 = interacao-1;
restricao2 = restricao_desloc;
P = sum(max(0,[restricao1 restricao2]));
alfa = 10^15;

obj = peso + alfa * P;
end
```

APÊNDICE B – Rotina computacional - *SGA*

B.1 Rotina principal

```
function [ minimo,coordenadas,dadositeration,diversidade] = RGA(F,alfa0,alfamin)
% Programado para encontrar o conjunto de coordenadas, contido dentro dos
% limites de lsup e linf, ao qual minimizam o valor de fobj, quando
% comparados com qualquer outro conjunto de coordendas que respeite as
% mesmas restrições
% lsup = limite superior do domínio analisado
% linf = limite inferior do domínio analisado
% alfa0 = porcentagem de aleatoriedade principal
% alfamin = porcentagem minima de aleatoriedade, visa garantir que a
% aleatoriedade nunca zere
% funobj = função a ser otimizada

format long
%% Parametros do Otimizador
%aqui são inicializados todos os parâmetros necessários para o
%funcionamento do algoritmo, cada qual sera explicado em particular;

[linf,lsup,fobj,dim] = fobjs(F);

elite = 1;
%Define a porcentagem de individuos que, pelo seu rank, garantem vaga no grupo ...
    de otimização
%sem necessitar participar do torneio

n = 50;
%População

nmax = 150;
%Número máximo de iterações

ng=0.2;
%Define o tamanho do grupo de otimização, representa a porcentagem do tamanho deste
%com relação a população total
nglobal = 0.7;
%Porcentagem das iterações dedicadas à otimização global

```

```
%nlocal = 1 - nglobal;
nlocal = round(100*(1-nglobal))/100;
%Porcentagem das iterações dedicadas à otimização local

npertub = 4;
%Quantidade de individuos inseridos no processo de otmização, por meio de
%pertubação relacionada a media e desvio padrão do grupo total

residuominimo = 0.01;%0.002;
%Porcentagem do alfaminimo que atuara como limitante inferior na
%aleatoriedade na etapa de otimização local

matrizaleatoriedade = [1 -4/(nmax*nglobal);0.25 -1/(4*nmax*nglobal);0 0];
%Coeficientes linear e angular, respectivamente, das retas que definem o
%decaimento da aleatoriedade com o número de iterações, referentes à etapa
%de otimização global

numeroderetas=size(matrizaleatoriedade);
%Número de retas utilizadas para definir o decaimento da aleatoriedade
%descrito anteriormente

tamanhotorneio = 4;
%Define o número de indivíduos que se enfrentara em cada etapa do torneio

alfa = (alfa0+alfamin)*(lsup-linf);
%Define a aleatoriedade de cada iteração, representa a amplitude do domínio
%ao qual cada indíviduo pertencente ao grupo de otimização pode gerar um descendente

if length(lsup) ≠ length(linf)
    disp('Dimensões inválidas');
else %verifica se os limitantes do domínio tem as mesmas dimensões
    %% Gera população inicial
    %Após a inicialização do algoritmo, este começa seu processo de
    %otimização gerando a população inicial, de maneira aleatória em
    %qualquer posição do domínio
    x = bsxfun(@plus,linf,bsxfun(@times,lsup-linf,rand(n,dim)));
    x = round(x);
    %% Avalia fitness da população
    %Avalia o valor da função objetivo em cada indivíduo da população
    %inicial. A matriz bancodedados armazena todos os dados referentes as
    %coordenadas e avaliação da função objetivo de todos os indivíduos
    fertilidade = zeros(n,1);
    for i = 1:n
        fertilidade(i,1) = fobj(x(i,:));
    end
    bancodedados = [fertilidade x];
    bancodedados = sortrows(bancodedados,1);
```

```matlab

%% Seleção do primeiro grupo de otimização
%Aqui inicia-se o processo de seleção de indivíduos para participarem
%do grupo de otimização, que tem por responsabilidade gerar os
%indivíduos para as próximas etapas do processo de otimização. O vetor
%indices contem o indice de cada individuo selecionado. Uma parte esta
%selecionada diretamente pelo rank, definidos pelo parâmetro elite.
%Outra parcela é selecionada pelo algoritmo de torneio

indices = (1:n*ng)';
indicestorneio = torneio(bancodedados(:,1),n*(1-elite)*ng,tamanhotorneio);
dadositeration=  zeros(nmax,dim+1);
indicestorneio = sort(indicestorneio);
indices(elite*ng*n+1:n*ng) = indicestorneio(:,1);

%% Formação do primeiro grupo de otimização
%Aqui efetivamente se monta o grupo de otmizaçao, selecionando os
%membros pelos indices determinados anteriormente
cresceram = zeros(n*ng,dim+1);
for i = 1:n*ng
    local = indices(i);
    cresceram(i,:) = bancodedados(local,:);
end

%% Inicio do processo iterativo
%Inicia-se a seguir o processo iterativo de otimização. Primeiramente
%inicia-se pelo otimização global, onde busca-se explorar o máximo possível ...
    o domínio.
%Após isso inicia-se uma etapa local, onde busca-se otimizar ainda
%mais a função objetivo nas proximidades do ponto ótimo até então

%% Processo de otimização global
%No processo de otimização global, cada indivíduo do grupo de
%otimização pode gerar um determinado número de outros indíviduos. A
%quantidade de filhos que ele pode ter é função do seu rank no grupo.
%Cada família acaba por ser avaliada e apenas o melhor indivíduo fara
%parte do próximo grupo de otimização, até terminarem as iterações.
%Após a geração de cada novo grupo de otimização, um determinado número
%de outros indivíduos substituem alguns membros ja pertencentes deste
%grupo. O número de indivíduos que farão isto é definido pelo parâmetro
%npertub, sendo estes inseridos em função da média e desvio padrão do
%grupo inteiro.
diversidade = zeros(n,1);

disp('Fase Global')
for k = 1:ceil(nglobal*nmax)
    indicespertub = torneioinverso(cresceram,npertub,tamanhotorneio);
```

```
        %Seleciona os membros que serão substituidos, através de um torneio
        %inverso, onde busca-se o perdedor para ser substituido
        for t = 1:npertub
            perturb = round(mean(cresceram(:,2:dim+1))) + ...
                t*round(std(cresceram(:,2:dim+1)).*(rand(1,dim)-0.5));
            perturb = max(perturb,linf);
            perturb = min(perturb,lsup);
            cresceram(indicespertub(t,1),2:dim+1) = perturb;
        end

        %% Processo de geração do grupo de otimização
        cresceram = filhotes(cresceram,n,ng,lsup,linf,alfa,fobj);
        cresceram = sortrows(cresceram,1);
        dadositeration(k,:) = cresceram(1,:);
        %Aqui se utiliza o grupo de otimização da iteração anterior, ja
        %saindo com o próximo grupo e a avaliação da função objetivo em
        %cada um desses indivíduos. vale ressaltar que o ponto ótimo obtido
        %até o momento nunca se perde, pois a participação dele no próximo
        %grupo de otimização depende apenas da sua avaliação da função
        %objetivo e este nunca sera substituido pelo torneio inverso, pois
        %possui rank 1

        %O dado do ponto ótimo até o momento é transferido para a matriz
        %dadositeration, que contem todos os dados de cada iteração
        %% Variação da aleatoriedade
        %Aqui varia-se a aleatoriedade. O decaimento é definido pelas retas
        %inicializadas anteriormente. Obtem-se um valor de ordenada entre 0
        %e 1 para a matriz matrizusada, de acordo com o máximo da ordenada
        %indicada por cada reta, em função da abscissa número de iterações.
        %Este valor representa a porcentagem a porcentagem da aleatoriedade
        %inicial que sera utilizada na próxima iteração.
        %Soma-se a isso o alfamin, a fim de se garantir que nunca se tenha
        %aleatoriedade zero, o que paralisaria o algoritmo.
        matrizusada = zeros(numeroderetas(1,1),1);
        for l = 1:numeroderetas(1,1)
            matrizusada(l,1) = ...
                max(matrizaleatoriedade(l,1)+matrizaleatoriedade(l,2)*k);
        end
        alfa = (alfa0*max(matrizusada)+alfamin)*(lsup-linf);
        disp(dadositeration(k,1));
        disp(alfa(1,1));
    end
    %% Processo de otimização local
    %Assemelha-se ao processo global, a principal diferença é que neste
    %processo não diferenciam-se os individuos por familia, para formar o
    %próximo grupo de otimização. Neste caso avaliam-se todos os indivíduos
    %igualmente, passando para o próximo grupo os indivíduos melhor
```

```
%rankiados, independente da familia ao qual pertencem
disp('Fase Local')
for k = 1:ceil(nmax*nlocal)
    indicespertub = torneioinverso(cresceram,npertub,tamanhotorneio);
    for t = 1:npertub
        perturb = round(mean(cresceram(:,2:dim+1))) + ...
         t*round(std(cresceram(:,2:dim+1)).*(rand(1,dim)-0.5));
        perturb = max(perturb,linf);
        perturb = min(perturb,lsup);
        cresceram(indicespertub(t,1),2:dim+1) = perturb;
        cresceram(indicespertub(t,1),1) = ...
            fobj(cresceram(indicespertub(t,1),2:dim+1));
    end
    %Processo de inserção de indivíduos idêntico ao processo que consta na ...
        etapa global
    bancodedados = filhoteslocais(cresceram,n,ng,lsup,linf,alfa,fobj);
    %Esta função entra com o grupo de otimização desta iteração e
    %retorna todos os indivíduos gerados e geradores, além de suas
    %avaliações da função objetivo
    %% Variação da aleatoriedade
    %Varia de maneira semelhante à variação da etapa anterior. Difere
    %por se tratar de apenas uma reta de decaimento para a
    %aleatoriedade
    alfa = (((nlocal*nmax-k)/(nlocal*nmax)) * ...
     alfamin+residuominimo*alfamin)*(lsup-linf);
    %Resíduominimo garante que a aleatoriedade não zere e também
    %possibilita trabalhar com mínimos de aleatoriedade diferentes para
    %a etapa global e local, bastanto introduzir este parâmetro com
    %valor diferente de 1
    bancodedados = sortrows(bancodedados,1);
    %Contém todos os dados de avaliação da função objetivo e
    %coordenadas de cada indivíduo desta iteração
    dadositeration(round(nglobal*nmax)+k,:) = bancodedados(1,:);
    %O melhor indivíduo da iteração é anexado à matriz dadositeration
    cresceram = zeros(ng*n,dim+1);
    indices = (1:n*ng)';
    indicestorneio = torneio(bancodedados(:,1),n*(1-elite)*ng,tamanhotorneio);
    indicestorneio = sort(indicestorneio);
    indices(elite*ng*n+1:n*ng) = indicestorneio(:,1);
    for i = 1:n*ng
        local = indices(i);
        cresceram(i,:) = bancodedados(local,:);
    end
    %Monta-se o novo grupo de otimização, através dos membros
    %elitizados (garantidos pelo seu rank) e daqueles vencedores do
    %torneio
    disp(alfa(1,1));
```

```
            disp(dadositeration(round(100*(nglobal*nmax))/100+k));
        end
        coordenadas = zeros(1,dim);
        coordenadas(1,:) = dadositeration(nmax,2:dim+1);
        %Coordenadas do ponto ótimo obtido ao fim do processo de otimização
        minimo = fobj(coordenadas);
        %Ponto ótimo obtido ao fim do processo de otimização
    end
end
```

B.2 Chamada da função objetivo

```
% lb is the lower bound: lb=[lb_1,lb_2,...,lb_d]
% up is the uppper bound: ub=[ub_1,ub_2,...,ub_d]
% dim is the number of variables (dimension of the problem)
function [lb,ub,fobj,dim] = fobjs(F)

switch F
    case 'portico'
        fobj = @fobj_portico;
        lb = [1 1 1 1 1 1 1 1 1];
        ub = [66 66 66 66 66 267 267 267 267];
        dim = 9;
    end
end
```

B.3 Criação das famílias - global

```
function [ cresceram ] = filhotes( cresceram,n,ng,lsup,linf,alfa,fobj,v )
%% Family generation code: GLOBAL phase %%
% It receives the current search group and returns their families
d = length(lsup);
indices = v;
for i = 1:ng*n
    tamanho = indices(i,1);
    ajuda = zeros(tamanho+1,d+1);
    x = zeros(tamanho+1,d);
    x(1,:) = cresceram(i,2:d+1);
    ajuda(1,:) = cresceram(i,:);
    for j = 1:tamanho
        x(j+1,:) = x(1,:) + alfa.*(rand(1,d)-0.5);
        for l = 1:d
            if x(1+j,l) < linf(1,l)
                x(1+j,l) = linf(1,l);
```

```
            else
                if x(1+j,l) > lsup(1,l)
                    x(1+j,l) = lsup(1,l);
                end
            end
        end
        ajuda(1+j,2:d+1) = x(j+1,:);
        ajuda(1+j,1) = fobj(x(1+j,:));
    end
    ajuda = sortrows(ajuda,1);
    cresceram(i,:) = ajuda(1,:);
end
end
```

B.4 Criação das famílias - local

```
function [ bancodedados ] = filhoteslocais(cresceram,n,ng,lsup,linf,alfa,fobj,v)
%% Family generation code: local phase %%
% It receives the current search group and returns their families
d = length(lsup);
bancodedados = zeros(n,d+1);
contador = 0;
ajuda = zeros(n,1);
indices = v;
for i = 1:ng*n
    tamanho = indices(i,1);
    x = zeros(n,d);
    x(contador+1,:) = cresceram(i,2:d+1);
    bancodedados(contador+1,:) = cresceram(i,:);
    for j = 1:tamanho
        x(contador+1+j,:) = x(contador+1,:) + alfa.*(rand(1,d)-0.5);
        for l = 1:d
            if x(contador+1+j,l) < linf(1,l)
                x(contador+1+j,l) = linf(1,l);
            else
                if x(contador+1+j,l) > lsup(1,l)
                    x(contador+1+j,l) = lsup(1,l);
                end
            end
        end
        ajuda(contador+1+j,1) = fobj(x(contador+1+j,:));
        bancodedados(contador+1+j,1) = ajuda(contador+1+j,1);
        bancodedados(contador+1+j,2:d+1) = x(contador+1+j,:);
    end
    contador = contador+tamanho+1;
end
```

```
if contador < n
    termo = n-contador;
    for l = 1:termo
        bancodedados(contador+1,2:d+1) = bancodedados(contador,2:d+1)+ ...
            alfa.*(rand(1,d)-0.5);
        bancodedados(contador,1) = fobj(bancodedados(contador,:));
    end
end
end
```

B.5 Torneio

```
function [ d ] = torneio( matriz,vencedores,tamanhotorneio )
%% Tournament code
[a,b] = size(matriz);
d = zeros(vencedores,1);
c = zeros(tamanhotorneio,1);
i = 1;
while i < vencedores+1
    for j = 1:tamanhotorneio
        c(j)=ceil(a*rand(1));
    end
    e = min(c);
    verification = ismember(d,e);
    if sum(verification)==0
        d(i,1) = e;
        i = i+1;
    else
    end
end
end
```

B.6 Torneio inverso

```
function [ d ] = torneioinverso( matriz,perdedores,tamanhotorneio )
%% "Inverse" Tournament code
% It is called here inverse because the "winners" are the worst designs
[a,b] = size(matriz);
d = zeros(perdedores,1);
c = zeros(tamanhotorneio,1);
i = 1;
while i < perdedores+1
    for j = 1:tamanhotorneio
        c(j)=ceil(a*rand(1));
```

```
    end
    e = max(c);
    verification = ismember(d,e);
    if sum(verification)==0
        d(i,1) = e;
        i = i+1;
    else
    end
end
end
```