Instituto Politécnico de Viseu
Escola Superior de Tecnologia e Gestão de Viseu

Mateus Kreutz Pauli

Utilização do BIM na otimização de vigas parede em betão armado por meio de algoritmo genético

Dissertação de Mestrado
Engenharia de Construção e Reabilitação

Professor Doutor Paulo A. da Silveira Costeira Marques da Silva
Professor Doutor Daniela Gutstein

Março de 2022

RESUMO

Em situações de escassez de recursos, bem como na busca de soluções mais sustentáveis, os sistemas e elementos estruturais devem ser projetados levando em consideração, além da funcionalidade e segurança, o custo total de construção e operação da estrutura. Verifica-se, portanto, que a minimização do custo total de um sistema estrutural envolve necessariamente uma otimização de parâmetros. Na análise estrutural, o projeto de elementos especiais (ou de regiões D, onde a Hipótese de Bernoulli não se aplica), embora amplamente estudado, é um tópico ainda discutido e tem padronização relativamente recente. Entre os métodos comumente usados para dimensionar essas regiões está o Método das Escoras e Tirantes (ou Strut and Tie Method - STM). O objetivo desta dissertação é estudar as possibilidades de desenvolvimento de um método computacional para dimensionamento de vigas-parede através do STM e a introdução de uma otimização inteligente e automatizada desses elementos estruturais na fase inicial de projeto. A dissertação aborda a utilização de conceitos da metodologia BIM, interoperabilidade entre programas e programação visual, além de utilizar conceitos de otimização baseados em algoritmos genéticos. Exemplos práticos disponíveis na literatura técnico-científica são utilizados como base para a validação da metodologia e dos resultados obtidos. O resultado desta dissertação é uma ferramenta que, graças à sua interoperabilidade, é capaz de otimizar a análise estrutural de vigas-parede de betão armado por meio de modelos de escoras e tirantes através da ferramenta de programação visual Dynamo. O driver de otimização é o algoritmo genético NSGA-II que torna as iterações mais inteligentes, trabalhando em direção a uma solução ótima mais rápidamente do que os métodos convencionais.

ABSTRACT

In situations of scarcity of resources, as well as in the search for more sustainable solutions, systems and structural elements must be designed taking into account, in addition to functionality and safety, the total cost of construction and operation of the structure. It appears, therefore, that the minimization of the total cost of a structural system necessarily involves an optimization of parameters. In structural analysis, the design of special elements (or D-regions, where the Bernoulli Hypothesis does not apply), although widely studied, is a topic still discussed and has relatively recent standardization. Among the methods commonly used to design these regions is the Strut and Tie Method (STM). The aim of this dissertation is to study the possibilities of developing a computational method for dimensioning deep beams using STM and the introduction of an intelligent and automated optimization of these structural elements in the initial design phase. The dissertation addresses the use of BIM methodology concepts, interoperability between programs and visual programming, in addition to using optimization concepts based on genetic algorithms. Practical examples available in the literature are used as a basis for validating the methodology and the results obtained. The result of this dissertation is a tool that, thanks to its interoperability, is capable of optimizing structural analysis of reinforced concrete deep beams through strut and tie models and Dynamo visual programming tool. The optimization driver is the NSGA-II genetic algorithm that makes iterations smarter, working towards an optimal solution faster than conventional methods.

PALAVRAS-CHAVE

BIM
Dynamo
Método de escoras e tirantes
Viga-parede
Algoritmo genético
Programação visual
Otimização estrutural

KEYWORDS

BIM
Dynamo
Strut and tie method
Deep beam
Genetic algorithm
Visual programming
Structural optimization

AGRADECIMENTOS

Agradeço à minha família, em especial à minha mãe e às minhas irmãs, por todo o apoio e suporte durante este período. Agradeço também toda ajuda que recebi dos amigos que fiz no decorrer do mestrado. Agradeço também aos meus orientadores, o Professor Doutor Paulo Costeira e a Professora Doutora Daniela Gutstein por toda a ajuda e suporte. Por último, agradeço ao Instituto Politécnico de Viseu e à Universidade Tecnológica Federal do Paraná pela oportunidade.

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE QUADROS

1. INTRODUÇÃO

1.1 Enquadramento

O Building Information Modeling (BIM) tem sido um dos tópicos mais comentados nos últimos anos na indústria da construção e já foi introduzida em várias áreas da Arquitetura, Engenharia e Contrução (AEC). Essa metodologia não só altera a forma de trabalhar entre os projetistas, como também revoluciona o modo como toda a estrutura organizacional de trabalho é aplicada. Esta metodologia permite uma melhor coordenação e colaboração entre os intervenientes no projeto, bem como uma rápida deteção de conflitos e consequentemente uma otimização de tempo e custos [1].

No campo da pesquisa de engenharia, métodos de dimensionamento rápidos, precisos e confiáveis são cada vez mais procurados. Conforme novas tecnologias e ideias são geradas, melhores métodos de dimensionamento podem ser estabelecidos. Nos últimos anos, a otimização estrutural tornou-se uma área emergente que tem a possibilidade de modificar o projeto clássico [2].

Ao automatizar o processo de avaliação, acredita-se que mais alternativas possam ser avaliadas dando um melhor entendimento e conhecimento do projeto. Acredita-se ainda, que com a introdução da automação de processos tediosos seja possível combater o esquecimento e auxiliar na tomada de decisões.

Para limitar o escopo desta dissertação, o foco recairá no campo da engenharia estrutural. A considerar as informações anteriormente citadas, o objetivo desta dissertação é desenvolver uma ferramenta semiautomática que, através da utilização de algoritmos genéticos, entregue ao utilizador propostas otimizadas para o pré-dimensionamento de vigas-parede de betão armado pelo método de escoras e tirantes, atendendo às normas em vigor no Brasil. O modelo ideal é gerado pela remoção de barras ineficazes de uma malha inicial previamente estabelecida pelo utilizador. Posteriormente, com o auxílio de algoritmos genéticos, faz-se o pré-dimensionamento das escoras e dos tirantes.

Por último, é feita uma validação das rotinas e módulos desenvolvidos neste trabalho a partir da comparação com exemplos disponíveis na literatura técnico-científica.

1.2 Objetivos e metodologia

Com este trabalho pretende-se:

- Demonstrar as possibilidades da utilização de ferramentas de programação visual e de algoritmos genéticos durante as fases de conceção e de projeto;
- Compreender como estabelecer a integração do Revit com os *softwares* de análise estrutural (Robot) e com ferramentas de programação visual, especificamente o Dynamo;
- Aplicar as ferramentas Dynamo e Optimo no pré-dimensionamento de vigas-parede a partir da obtenção de um modelo de escoras e tirantes;
- Propor uma possível abordagem no pré-dimensionamento de estruturas laminares de betão armado, que combine a metodologia BIM com as mais recentes ferramentas de programação visual e de otimização.

O trabalho é realizado com a ferramenta de programação visual (VP) Dynamo, que pode interagir com o Revit e o Robot Structural Analysis (Robot) por meio da Interface de Programação de Aplicativos (API), o que garante a simultaneidade das ações, e possibilita a realização de vários tipos de Otimização Multi-Objetivo (MOO) em vários tipos de estruturas. Face aos objetivos, a metodologia de otimização genética do pré-dimensionamento de vigas-parede pelo método de escoras e tirantes, pretende:

- Permitir ao utilizador inserir os valores para criação geométrica da viga-parede de acordo com suas necessidades;
- Permitir ao utilizador inserir informações inerentes ao processo de análise estrutural, como, por exemplo, os casos de carga e os materiais;
- Determinar os valores condicionantes do processo de otimização genética, como, por exemplo, a população inicial e o número de iterações;

- Após o processo de otimização, fornecer ao utilizador uma pré-solução estrutural otimizada.

1.3 Estrutura do texto

Este trabalho está subdividido em 5 capítulos, complementados pelos apêndices e bibliografia. Os dois primeiros capítulos são de caráter teórico e os seguintes são dedicados ao desenvolvimento do modelo semiautomático e às conclusões, conforme descrito abaixo:

Capítulo 1 – INTRODUÇÃO: No presente capítulo é realizada uma introdução ao trabalho realizado, contextualizando o tema (enquadramento), apresentando os objetivos a serem alcançados e a metodologia inicial idealizada para tal.

Capítulo 2 – ESTADO DA ARTE: Neste capítulo, apresenta-se o enquadramento teórico da metodologia BIM, de como ela deve ser implantada em projetos estruturais e como ela se relaciona com a programação visual. Para além da referência aos conceitos de interoperabilidade e conceitos relacionados a otimização genética, são descritos, ainda, os conceitos relativos às estruturas laminares de betão armado, caso particular de vigas-parede, bem como sobre os diferentes métodos, frequentemente, utilizados para o dimensionamento deste tipo de elementos estruturais.

Capítulo 3 – DESENVOLVIMENTO DA MODELAÇÃO AUTOMATIZADA: No referido capítulo é abordado todo o procedimento realizado para a criação de um *workflow* no Dynamo que, através da metodologia BIM interaja com as demais aplicações, em tempo real, e possibilite a obtenção do modelo de escoras e tirantes adequado ao dimensionamento de uma viga-parede pelo método das escoras e tirantes (Strut and Tie Method – STM). São caracterizados suscintamente os *softwares* utilizados e, para fins de validação da metodologia proposta, é realizada a comparação da topologia obtida para sistemas estruturais em treliça, considerando apenas um tipo de material, através do procedimento aqui desenvolvido aplicado a exemplos disponíveis na literatura técnico-científica.

Capítulo 4 – DIMENSIONAMENTO DE VIGAS-PAREDE – CASO DE ESTUDO: Neste capítulo, é feita uma comparação da topologia obtida para sistemas estruturais em treliça, considerando dois tipos de materiais (betão e aço), através do procedimento apresentado no capítulo anterior, aplicado num exemplo disponível na literatura técnico-científica. É

apresentada a implementação da otimização estrutural por meio de algoritmos genéticos, que permite otimizar a quantidade de materiais (betão e aço) a considerar no modelo de escoras e tirantes de vigas-parede, de modo a respeitarem as disposições regulamentares. O capítulo termina com a apresentação de caso de estudo sobre uma viga-parede, exemplo disponível na literatura técnico-científica, comparando tanto a topologia obtida para o modelo de escoras e tirantes como o pré-dimensionamento dos respetivos materiais.

Capítulo 5 – CONCLUSÃO E DESENVOLVIMENTOS FUTUROS: Neste capítulo, são apresentadas as conclusões gerais do trabalho, as principais dificuldades sentidas, as limitações encontradas e são feitas sugestões para o desenvolvimento de trabalhos futuros.

REFERÊNCIAS: Nesta secção são apresentadas as referências teóricas e práticas utilizadas como base para desenvolvimento desta dissertação.

APÊNDICE 1: Nesta secção apresentam-se os aspetos mais relevantes dos procedimentos e códigos desenvolvidos.

2. ESTADO DA ARTE

2.1 Building information modeling (BIM)

Atualmente, o processo de implementação de um empreendimento ainda é fragmentado e depende de meios de comunicação baseados em papel e em CAD 2D e 3D. Erros e omissões nestes documentos geralmente causam custos de imprevistos e eventuais ações jurídicas entre os vários participantes de um empreendimento. Esses problemas causam atrito, despesas financeiras e atrasos. Um dos problemas mais comuns associados a este tipo de comunicação durante a fase de projeto é o tempo considerável e os gastos necessários para gerar informações críticas para a avaliação de uma proposta de projeto, a incluir as estimativas de custo, análise de uso de energia, detalhes estruturais, entre outros [3].

O BIM é uma metodologia inovadora de trabalho colaborativo baseada na elaboração de um modelo virtual que vem proporcionar uma nova abordagem à gestão da informação na construção, à qual destinam empreendimentos e outras obras de Engenharia Civil [4]. Essa metodologia assenta-se, essencialmente, na partilha da informação entre todos os intervenientes durante as fases do ciclo de vida de um edifício (projeto, construção, manutenção e desconstrução), ver Figura 1, nomeadamente entre a arquitetura, as especialidades, os construtores e os donos de obra [5].

Um modelo BIM fiável e quantificável exige sempre um esforço de modelação que cresce exponencialmente em relação ao grau de detalhe que se pretende atingir [6].

Diferente do CAD, cujo principal objetivo é a automação dos aspetos da produção do desenho tradicional, o BIM é uma mudança de paradigma como se pode verificar na Figura 2. Pela automação parcial do detalhe de modelos de um edifício ao nível da construção, o BIM distribui a concentração dos esforços, o que proporciona mais ênfase à fase de conceção do projeto [3].

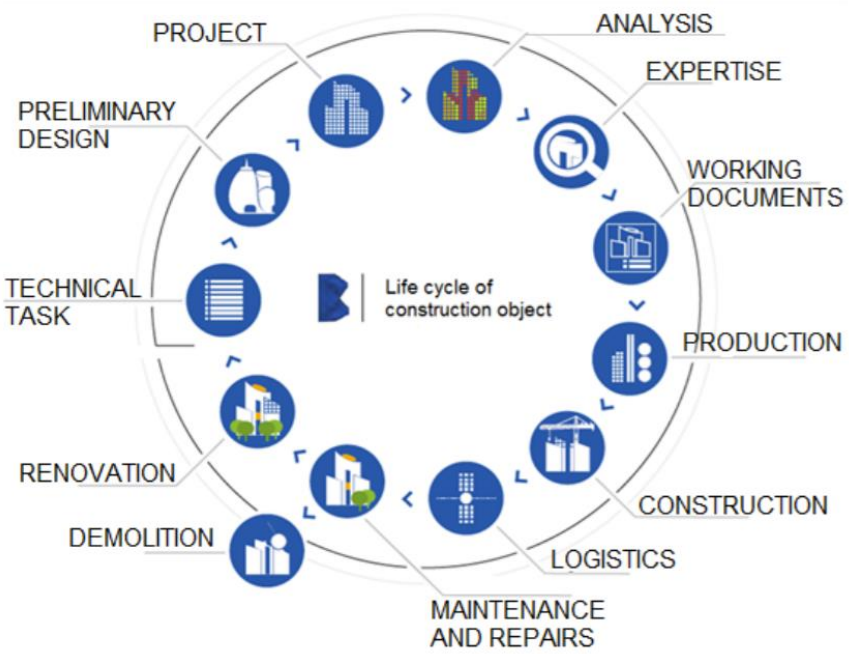


Figura 1: Ciclo de vida de um edifício (adaptado de [7]).

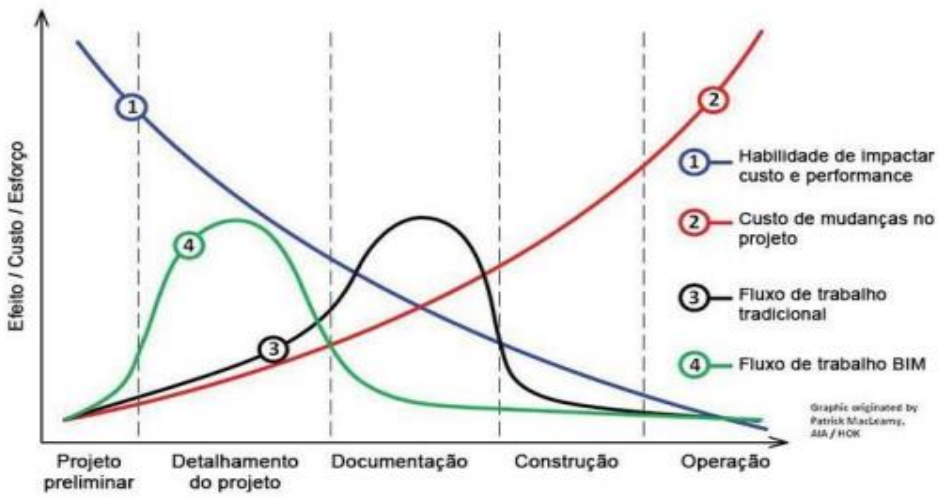


Figura 2: Fluxo de trabalho tradicional vs BIM [8].

O BIM está a deixar de ser encarado como apenas uma mera mudança tecnológica e de *Software* para passar também a uma mudança de procedimentos [5]. Sendo bem implementado, vem a possibilitar melhorias na fase de vida útil das edificações e após a conclusão da obra, garantindo ao utilizador ou proprietário da obra uma melhor gestão das manutenções preventivas, com segurança e menor custo-benefício.

2.1.1 Implementação do BIM em projetos de estruturas

Tradicionalmente, os engenheiros projetistas estruturais iniciam um novo projeto pela análise dos desenhos feitos pelo arquiteto e, ao seguirem as instruções sobre materiais, layout, propriedades da secção e carregamento, criam uma documentação de projeto junto com vários modelos analíticos [10]. O BIM vem oferecer a estes engenheiros um processo de organização e partilha de informação apoiado numa base de dados global com informação estruturada num formato normalizado, onde consta toda a informação considerada necessária para definir uma estrutura [11].

Segundo Ferreira et al. [10], no âmbito do projeto de estruturas existem 3 campos principais para a implementação de um processo BIM: a coordenação interdisciplinar, a análise e dimensionamento de estruturas e a documentação do projeto (Figura 3).

1. **Coordenação interdisciplinar:** a adopção de um processo BIM permite um ganho de eficiência face às práticas correntes por duas vias:

   - Desenvolvimento e utilização de famílias de *software* comercial que contenham funções colaborativas que permitam, por exemplo, a partilha de modelos de trabalho de forma a atender a um conjunto de permissões dadas a cada interveniente.

   - O desenvolvimento e utilização de plataformas abertas para a troca de modelos (projetos) completos, em que cada interveniente contribui para a construção de um modelo global que incorpora um conjunto alargado de informação.

2. **Análise e dimensionamento:** a partir de modelos BIM será uma das áreas de desenvolvimento em que o contributo dos engenheiros civis é mais necessário, pois esta etapa compreende as metodologias, modelos e tipos de análises estruturais adotadas, bem como as etapas de dimensionamento estrutural de acordo com normas técnicas e critérios de projeto pré-definidos.

3. **Documentação:** a documentação do projeto é porventura o domínio de aplicação do processo BIM onde os benefícios imediatos obtidos pelos envolvidos são mais evidentes. A facilidade com que é possível gerar peças desenhadas a partir do modelo tridimensional é uma função de base das ferramentas BIM mais populares [12].

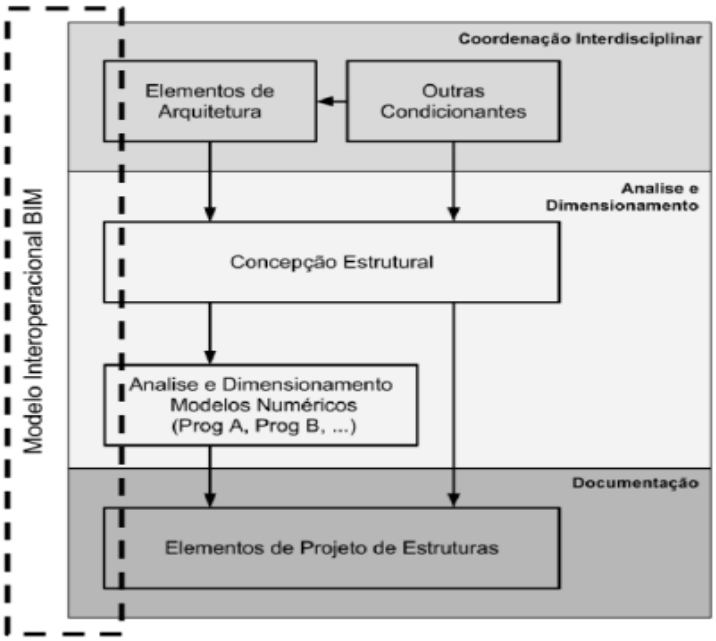


Figura 3: Esquema de integração de modelos BIM no projeto de estruturas [12].

A implementação prática desta metodologia junto dos projetistas de estruturas começa a ser evidente, porém, encontram-se ainda algumas dificuldades, nomeadamente, as inerentes ao investimento, à curva de aprendizagem, aos direitos de propriedade e de responsabilização pelo modelo, ao diferente estado de maturidade em que se encontram os diversos intervenientes num projeto e, também, a complexidade da modelação estrutural de formas menos correntes [5].

2.1.2 Interoperabilidade

O processo de projeto pode envolver muitas fases e diversos colaboradores, portanto, a troca de informações de maneira rápida e eficiente ao longo do ciclo de vida do projeto faz-se necessária. Segundo Eastman et al. [3], esta troca de dados e informações entre aplicativos computacionais no processo de projeto e a capacidade de identificação é denominado como interoperabilidade (Figura 4). Com a interoperabilidade evita-se a réplica de dados que já tenham sido gerados e facilita-se, de forma automatizada e sem obstáculos, o fluxo de trabalho durante o projeto. Ainda segundo Eastman et al. [3], para a passagem de dados recorre-se ao uso de arquivos baseados em formatos de trocas de dados. As trocas de dados entre dois aplicativos BIM são efetuadas basicamente de quatro maneiras diferentes: ligação direta; formato de arquivo de troca proprietário; formatos de trocas de dados de domínio público; formatos de troca de dados

baseados em eXtensible Markup Language (XML). Sendo o CIMsteel Integration Version 2 (CIS/2) e o Industry Foundation Classes (IFC) os dois principais modelos de dados do produto da construção civil.

Dentre as fases de análise e desenho de um projeto, a abordagem do processo do modelo pode ser dividida em duas, nomeadamente o modelo global e o modelo local. Esses modelos podem ter a necessidade de serem desenvolvidos em *softwares* diferentes, que podem ter quase nenhuma interoperabilidade entre eles [12]. Uma única mudança no dimensionamento do projeto requer uma alteração um a um em todos os modelos, o que pode resultar em ambos os projetos defeituosos e demorados [7], junto com os custos adicionais resultantes. Isso resulta num fluxo de trabalho altamente perturbado, que depende fortemente de recursos humanos para o manipular e coordenar [12]. Ao aproveitar-se das vantagens do BIM, esses modelos podem ser combinados num modelo central, que contenha a representação física e analítica do modelo estrutural. As informações físicas contêm dados usados nas aplicações de análise, enquanto as informações analíticas correspondem ao modelo usado na análise estrutural. Estas assumem duas visões distintas do mesmo modelo e interligadas, permitindo não só as análises estruturais do projeto, mas também a produção de documentos de construção [13].

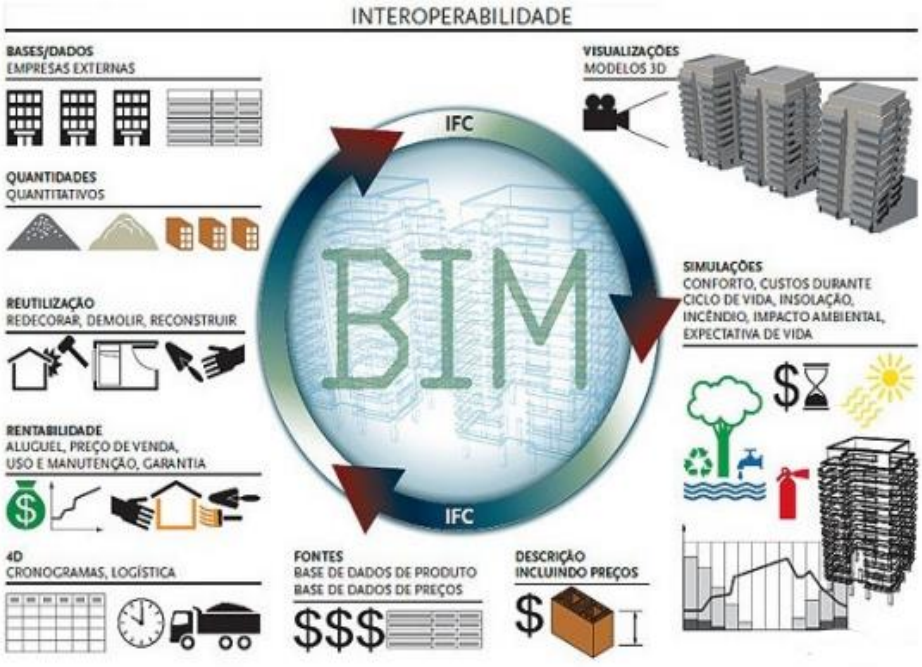


Figura 4: Interoperabilidade [14].

Num fluxo BIM a partilha de informação entre colaboradores está dependente da interoperabilidade dos seus sistemas. O acesso a uma metodologia colaborativa avançada, materializada na centralização e compatibilização de todas as especialidades num só modelo, aumenta as exigências ao nível dos requisitos de interoperabilidade [15].

Partindo de uma perspectiva global, o desenvolvimento de padrões para implementação de projetos colaborativos tem sido mais lento do que o esperado, e a maior barreira técnica é a necessidade de ferramentas de interoperabilidade maduras.

2.2 Estruturas laminares de betão armado

As estruturas laminares têm variadas aplicações na área da Engenharia Civil, a sua aplicação vai desde a utilização em pavimentos, reservatórios, muros, coberturas, etc. Essas estruturas são caracterizadas por terem uma espessura reduzida quando comparada com as outras duas dimensões, e esta designação vem a ser normalmente utilizada para estruturas com uma relação espessura/comprimento superior a 1:4 [16].

De acordo com a Mecânica Estrutural, em função da geometria (plana ou curva) e condições de carga (no plano e/ou fora do plano), as estruturas laminares podem ser geralmente divididas em três tipos: membrana (chapa ou parede), laje (ou placa) e casca (Figura 5). As estruturas de membrana são planas e realizam o transporte das forças no plano; as estruturas de laje também são planas mas realizam o transporte de cargas no plano normal ao plano da estrutura, e finalmente, as estruturas de casca podem apresentar a forma plana ou curva com cargas no plano e fora deste [17].

Normalmente, o processo de dimensionamento de estruturas de betão armado pode ser dividido em dois procedimentos inter-relacionados:

1) Determinar o estado de tensão existente nos elementos estruturais;

2) Determinar as quantidades dos materiais de forma a obter a configuração de rotura plástica do elemento estrutural no estado limite último.

Para a primeira parte, é prática comum realizar uma análise linear elástica através de programas de elementos finitos (Finite Element Method - FEM), enquanto para a segunda parte é usual a utilização de um método de concepção baseada nas equações de equilíbrio e nas relações constitutivas dos materiais utilizados para prever o comportamento da estrutura em carga

máxima. O desenvolvimento de um método de dimensionamento que anteveja como o betão armado vai realmente responder a um conjunto específico de tensões é uma tarefa complexa, mesmo para o mais simples elemento estrutural (elemento de membrana). As dificuldades decorrem do comportamento não-linear do betão armado e a possível correlação entre o conjunto de tensões instaladas com a correspondente capacidade de resistência do betão e da armadura [17].

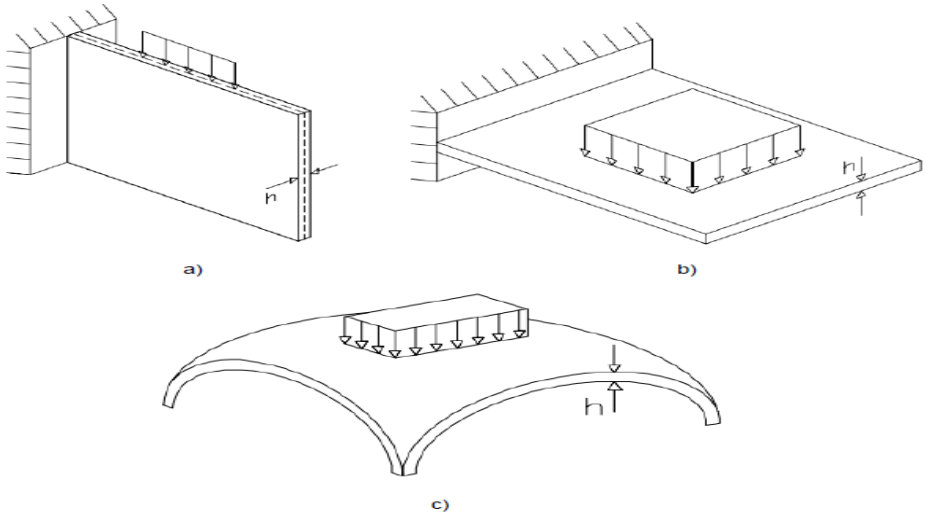


Figura 5: Tipos de estruturas laminares: a) elemento de membrana (parede ou chapas); b) elemento de placa (lajes); c) elemento de casca (adaptado) [17].

2.3 Vigas-parede

Um exemplo de peças laminares são as vigas-parede, usadas como elementos estruturais de distribuição de carga, como vigas de transferência, blocos de estacas, paredes de fundação e estruturas *offshore*. A ABNT NBR 6118:2014 [18] no item 22.4.1. caracteriza as vigas-parede (Figura 6) como "as vigas altas em que a relação entre o vão e a altura ℓ/h (vão teórico sobre altura) é inferior a 2 em vigas biapoiadas e inferior a 3 em vigas contínuas". Essa regra geral é apresentada por Leonhardt e Mönnig [19]. Schlaich et al. [20], no entanto, afirmam que regras simples como relações ℓ/h podem levar a classificações indevidas, sendo necessário considerar tanto a geometria quanto o carregamento para a classificação de vigas-parede.

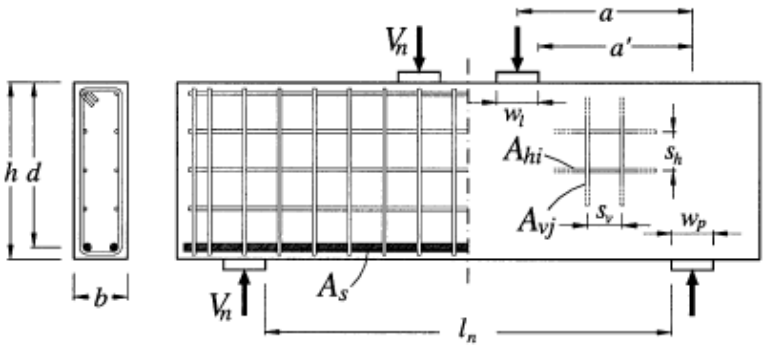


Figura 6: Geometria de uma viga-parede [21].

Ainda no contexto da classificação, o Eurocódigo 2 [16] preconiza que são vigas-parede aquelas com $\ell/h < 3$, sendo, portanto, mais abrangente e conservador que a norma brasileira. O ACI 318-14 [21], por exemplo, define as vigas-parede como membros cuja relação $\ell/h \leq 4$ ou nos quais exista uma carga concentrada com uma distância de até 2 vezes a altura h entre duas cargas concentradas ou entre uma carga e um apoio. Desse modo, a norma requer que esses elementos sejam dimensionados considerando a distribuição não linear de tensões ao longo da altura da viga.

De facto, não existe consenso acerca de uma relação mínima ℓ/h para a qual as vigas se comportem como vigas-parede.

Numa viga esbelta sob carregamento, o estado de tensão é tal que se pode estimar a sua capacidade de carga pela sua resistência à flexão, caso não ocorra alguma rotura prematura por corte (geralmente evitada com armadura transversal). Porém, em vigas-parede, devido ao pequeno valor da relação ℓ/h, esta resistência nem sempre é atingida, o que leva a peça a atingir a rotura por corte, para níveis de carga menores que os esperados para a flexão [22] [23]. Além disso, de acordo com J.K. Oh e S.W. Shin [24], a resistência ao corte das vigas-parede é significativamente maior do que a das vigas esbeltas, devido à sua capacidade especial de redistribuir as forças internas antes da rotura e porque os seus mecanismos de desenvolvimento e de transferência de forças são bastante diferentes das vigas esbeltas.

A ABNT NBR 6.118:2014 [18], classifica as vigas-parede como sendo elementos estruturais especiais, que para serem entendidos é necessário conhecerem-se os conceitos da hipótese de Bernoulli e do Princípio de Saint-Venant.

A hipótese de Bernoulli ou hipótese das seções planas, enunciada por Jacob Bernoulli (1654-1705), estabelece que as seções planas permanecem planas e perpendiculares à linha neutra de

uma barra, após a deformação. Portanto, pode-se assumir que a distribuição de deformações é mantida linear, desde o início do carregamento até à ruína [25].

Apesar de desprezar as deformações de distorção provocadas pelo esforço de corte, a hipótese de Bernoulli possibilita o dimensionamento da maioria dos elementos estruturais. Porém, aos elementos estruturais especiais, esta hipótese não pode ser estendida, uma vez que eles possuem, em sua constituição, regiões onde não se observa a linearidade na distribuição de deformações. Tais regiões podem ser entendidas ao se estudar o Princípio de Saint-Venant [25].

Pelo Princípio de Saint-Venant pode-se concluir que, apenas, em regiões suficientemente afastadas do ponto de aplicação da carga a hipótese de Bernoulli é válida, pois nestas regiões não ocorrem grandes perturbações de tensão. Já na região próxima do carregamento, as deformações por esforço de corte apresentam valores significativos, obrigando à sua consideração no dimensionamento dos elementos estruturais [25].

A partir da hipótese de Bernoulli e do Princípio de Saint-Venant, os engenheiros alemães Schlaich e Schäfer [20] classificaram as regiões das estruturas em regiões B e D (Figura 7). As regiões nas quais a hipótese de Bernoulli, de distribuição linear de tensões e deformações, é válida, foram designadas de regiões B. As regiões onde essa hipótese não é válida foram chamadas de regiões D, onde a letra D refere-se a Descontinuidade. Em geral, o limite entre as regiões B e D pode ser considerado localizado a uma distância h (altura da secção transversal do elemento estrutural considerado) da secção efetiva da descontinuidade. Porém, observa-se que em alguns elementos estruturais a descontinuidade é generalizada, logo, a estrutura como um todo é definida como região D. É o que ocorre, por exemplo, nas vigas-parede [25].

Na Figura 8 está representada a distribuição de tensões normais à secção transversal e a esbeltez ℓ/h, na secção do meio do vão, para uma viga-parede submetida a carregamento uniformemente distribuído.

O comportamento estrutural de vigas-parede começa a se diferenciar gradualmente do comportamento de vigas esbeltas a partir da relação entre o vão e a altura igual a 2 (dois), no caso de vigas com um vão único. Quanto menor é esta relação, mais o diagrama das deformações deixa de ser linear, consequentemente, mais a linha neutra da secção é empurrada para baixo [20].

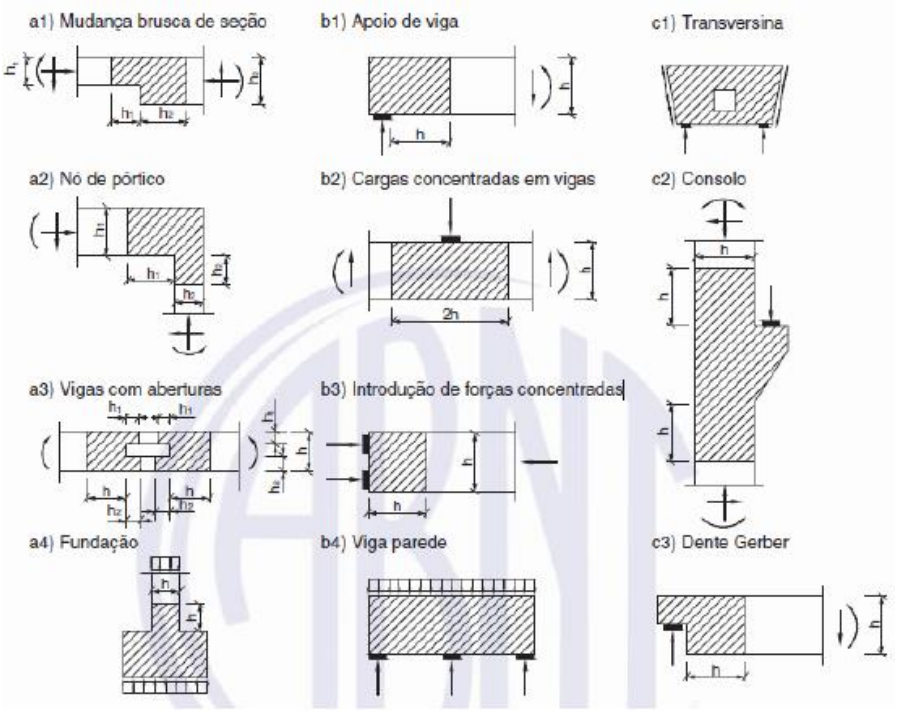


Figura 7: Situações típicas de regiões D [18].

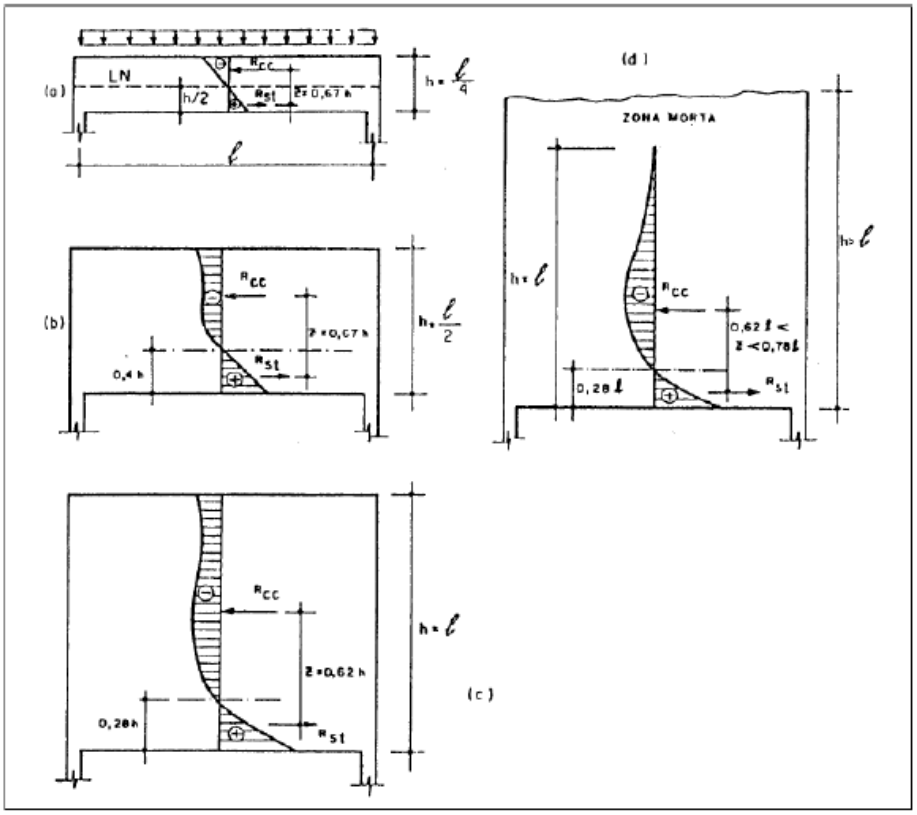


Figura 8: Distribuição das tensões normais na secção do meio do vão, em vigas de vão único, para diferentes valores de ℓ/h, com carregamento uniforme [19].

Percebe-se que para diferentes relações de ℓ/h, ter-se-ão distribuições singulares para as tensões, o que leva a comportamentos resistentes distintos. Esta é uma das inúmeras variáveis que dificultam o dimensionamento de vigas-parede, assim como podem ser citadas: a posição do carregamento (superior ou inferior) e do bordo onde está aplicado; a espessura da viga; a presença ou não de enrijecedores de apoio, de espessamentos locais e da existência de pilares nas extremidades, da resistência do betão utilizado e das taxas e distribuições das armaduras na peça [26].

Na Figura 9, observa-se a trajetória de tensões proposta por Leonhardt e Mönnig [19] para a mesma viga com relação ℓ/h = 1. Esse estudo da trajetória das tensões é importante, uma vez que é o instrumento por meio do qual podem ser elaborados os modelos de escoras e tirantes.

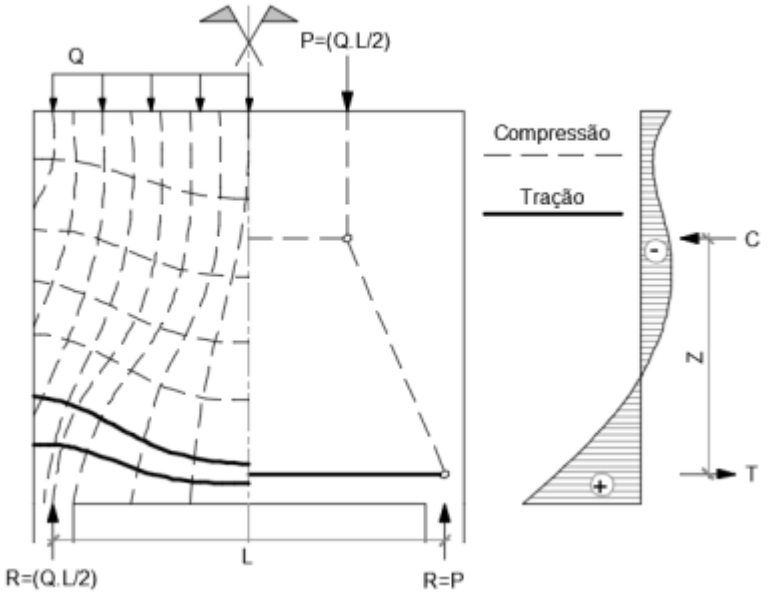


Figura 9: Exemplo de modelo de escoras e tirantes seguindo a trajetória das forças numa viga-parede [27].

Devido à variabilidade na natureza dos modos de rotura, é bastante complexa a identificação dos mecanismos de rotura e a avaliação da resistência ao corte de vigas-parede. Até mesmo os manuais de projeto mais recentes, como o ACI e o Eurocódigo oferecem poucas orientações sobre o projeto de vigas-paredes, em particular quando existem complexidades como por exemplo aberturas [2]. Entretanto, a norma brasileira ABNT NBR 6118:2014, em suas últimas versões vem abordando o tema com mais detalhe, onde a Secção 22 intitulada “Elementos

Especiais", apresenta os critérios de projeto de vigas-parede, para fins de dimensionamento e pormenorização.

No que diz respeito ao dimensionamento, a ABNT NBR 6118:2014 admite que sejam utilizados modelos planos elásticos lineares ou não lineares baseados em métodos numéricos adequados, como o método dos elementos finitos (FEM) ou o método Corda-Painel. Admite também, para o dimensionamento das vigas-paredes no estado limite último, modelos concebidos a partir do método das escoras e tirantes (STM). Na definição destes modelos, de forma a assegurar um comportamento adequado em serviço, a geometria das treliças deve ser tal que os valores das forças nos tirantes resultem o mais próximo possível dos obtidos num modelo linear elástico plano.

O Método Biela-Painel (também chamado "Método Corda-Painel", "Stringer Panel Method (SPM)" ou "Stringer Method") é um meio de análise de estruturas bidimensionais do tipo parede. Conforme afirmam Blaauwendraad e Hoogenboom [28], o Método Biela-Painel é uma alternativa ao Métodos de Escoras e Tirantes e ao Método dos Elementos Finitos. Também como este último, o SPM baseia-se no Método dos Deslocamentos, sendo um dos motivos para a sua boa aceitação e para implementação computacional.

O SPM baseia-se na divisão da estrutura em bielas, que são elementos lineares dispostos horizontal e verticalmente e em painéis, que possuem geometria retangular e são limitados por quatro bielas nas extremidades. Estas visam suportar os esforços axiais (compressão e tração), enquanto os painéis são sujeitos aos esforços de corte [28].

Já o Método dos Elementos Finitos baseia-se na ideia de representação de um domínio contínuo a partir de partes discretas. Entre as vantagens do FEM está a possibilidade da análise de estruturas de geometria complexa, a previsão da sua vida útil e a diminuição de custos relacionados à utilização de protótipos. Além disso, pode-se modelar a estrutura em diferentes situações, possibilitando a sua otimização [29].

Uma característica do FEM, de acordo com Azevedo [29], é a grande quantidade de cálculos necessários para a sua aplicação. Dessa forma, o método só é útil de forma prática a partir da utilização de um computador que possibilite o processamento dos dados envolvidos nas análises.

O Método de Escoras e Tirantes será utilizado neste trabalho e, portanto, será descrito de forma detalhada no Capítulo 4.

2.4 Otimização estrutural

A otimização é a busca da melhor solução para uma operação, enquanto certas restrições são satisfeitas. O conceito está bastante presente no cotidiano [30]. Uma grande vantagem de otimizar projetos é o abandono de parâmetros baseados na intuição ou na experiência dos engenheiros, enfatizando a busca por determinação probabilística. A otimização também tem como destaque a possibilidade de sistematização de procedimentos para o dimensionamento ótimo das estruturas.

Nos últimos tempos, alguns desafios têm surgido, e um deles está relacionado ao consumo de energia e de massa, que antes não grande relevância frente aos esforços envolvidos, devido ao seu baixo custo e abundância, mas que agora tem sido um assunto muito discutido e colocado em destaque, no cenário nacional e mundial [31]. Com os avanços computadionais recentes, a otimização nos projetos de engenharia, por sua vez, torna-se imprescindível, nesse cenário de escassez de recursos e crescimento da demanda.

De acordo com Maia [32], a ideia de aperfeiçoar ou otimizar uma estrutura implicitamente pressupõe alguma liberdade de alterá-la. Os elementos para um problema de otimização são as ferramentas para tais mudanças. A formulação segue um processo praticamente comum para todo e qualquer problema.

Para se definir um problema de otimização, é necessário:

a) um conjunto de variáveis, ditas de projeto, que variam na busca do ótimo;

b) uma função objetivo (ou função de desempenho);

c) um conjunto de restrições (exigências) que são respeitadas.

O nível de sucesso da otimização é determinado em função dos objetivos a serem alcançados. Os objetivos típicos da otimização são: minimizar a tensão, a deformação, o peso, o custo e a energia incorporada enquanto se maximiza a repetição de componentes [30]. Em geral, os objetivos desejados estão em conflito, como a minimização do custo e a manutenção das condições mínimas de segurança, caracterizando assim um problema multiobjetivo. Devido a esse conflito, a resolução de um problema multiobjetivo obtém um conjunto diversificado de soluções de onde o(s) tomador(es) de decisão tem a oportunidade de selecionar aquela que melhor se adapta às suas preferências [33].

O projeto de estrutura ideal é um tópico muito interessante e importante no campo da engenharia e construção. A resolução de problemas de otimização estrutural geralmente é definida por um

processo de encontrar um projeto ideal da estrutura com o mínimo de material, sujeito a algumas restrições de projeto e condições de carregamento específicas. A metodologia de projeto ideal de estruturas pode ser categorizada por dimensionamento, forma e otimizações de topologia [34]. A otimização da forma visa obter uma forma ideal da estrutura, alterando a configuração geométrica sem alterar o dimensionamento e os parâmetros do material. Na otimização da topologia, é feita uma configuração estrutural inicial que satisfaz um conjunto de critérios de projeto sem configuração geométrica predefinida da estrutura. Assim, ao otimizar os parâmetros da forma e tamanho, procura-se obter uma geometria estrutural nova e inovadora. Recentemente, vários mecanismos bioinspirados na natureza ganharam popularidade na solução de problemas complexos de otimização combinatória de estruturas [35-38].

Atualmente, muitos são os estudos sobre a otimização de uma estrutura em treliça [35, 37, 38]. A maioria dos estudos de otimização em problemas de treliça é baseada na suposição de variáveis de projeto discretas, onde cada membro da estrutura de treliça é tratado como tendo variáveis de projeto separadas (isto é, comprimento, espessura, área de secção transversal, etc.). A desvantagem associada aos problemas de otimização de treliça é a determinação das funções objetivo e penalidade, que são necessárias para estimar a solução viável e penalizar a solução inviável que é inicialmente desconhecida [39].

Em linhas gerais, os problemas respondidos pela otimização são puramente a maximização ou a minimização de uma função objetivo com uma ou mais variáveis presentes em um determinado domínio [40]. Frequentemente existe um conjunto de restrições ou limitações nas variáveis que caracterizam as necessidades do projeto, e tais variáveis podem ter valores de igualdades ou desigualdades, sendo os projetos de valores desiguais os mais complexos [41].

Os algoritmos de otimização podem ser divididos em cinco grupos principais: Algoritmos Determinísticos (DA), Algoritmos Estocásticos (SA), Recozimento Simulado (*Simulated Annealing - SA*), Algoritmos Evolutivos (EA) e Enxame de Partículas (PS) [42]. Os algoritmos de interesse neste trabalho são definidos a seguir.

2.4.1 Algoritmos evolucionários

Os algoritmos evolutivos (*EA – Evolutionary Algorithm*) são utilizados para resolver desafios do mundo real com base no comportamento das espécies e incluem algoritmos de enxame de

partículas e algoritmos genéticos. Esses algoritmos geralmente são escolhidos por sua capacidade de resolver problemas muito complexos com muitos parâmetros e vários objetivos a serem alcançados [42].

2.4.1.1 Algoritmos genéticos

Os algoritmos genéticos (*Genetic Algorithms - GA*) são um tipo particular de algoritmo evolutivo, que também se basearam na natureza, na teoria da evolução das espécies proposta por Darwin [43], o que inclui operadores genéticos como seleção, mutação e conceitos de *crossover* (Figura 10). Não seguem nenhuma linearização ou simplificação enquanto procuram um ótimo global, dado pela minimização ou maximização de uma função de aptidão predefinida. Essa função considera todos os critérios importantes de dimensionamento com o respetivo peso para a pontuação global da solução [42].

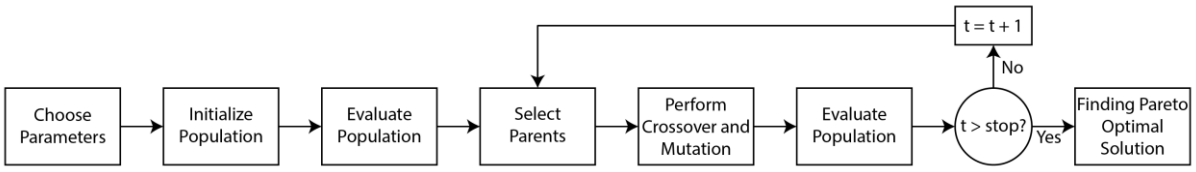


Figura 10: Fluxo geral de otimização [44].

2.4.1.2 Algoritmo genético de seleção natural

O Algoritmo Genético de Seleção Natural abreviado por NSGA-II é um algoritmo de otimização genética evolutivo, elitista, multi-objetivo e não dominado. O NSGA-II consiste em 6 etapas como pode ser visto na Figura 11. Pelo uso de elitismo, o NSGA-II não permite que uma solução ótima de Pareto seja excluída, enquanto mantém uma complexidade máxima de O (MN^2) onde M = número de funções objetivo, N = tamanho da população, e O = limite superior [45].

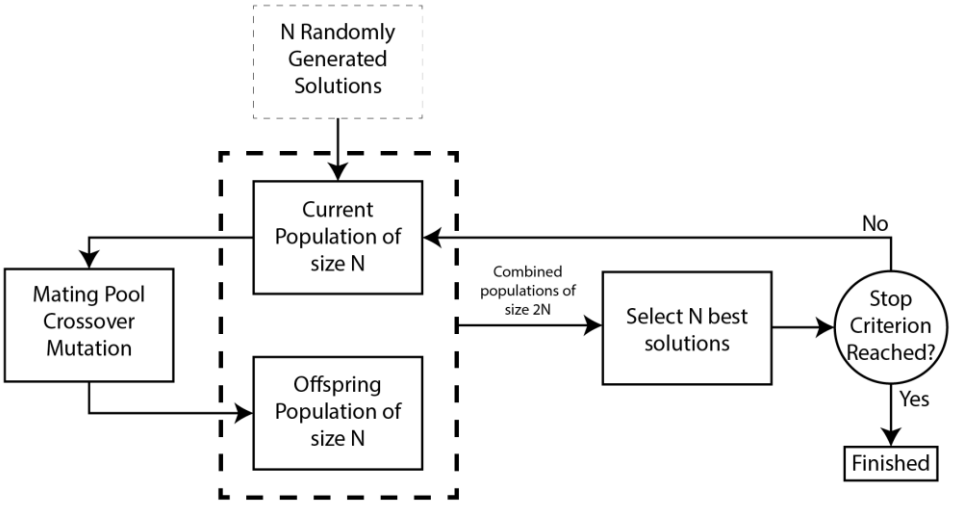


Figura 11: NSGA-II algoritmo [1].

As vantagens do algoritmo NSGA-II são a preservação explícita da diversidade, a complexidade limitada e o elitismo. As desvantagens são a possibilidade de uma comparação aglomerada que restringe a convergência e uma classificação não dominada com o tamanho de 2N [1].

2.5 Programação visual

A programação gráfica ou também chamada de programação visual usa o conceito de programação textual, mas simplifica a abstração ao substituir trechos de código textual por componentes gráficos. "*No sentido convencional, [...] programação gráfica também têm sintaxe e semântica bem definidas*" [46], mas apresentadas de uma forma mais conveniente e menos abstrata para não-programadores.

Com esta abordagem flexível, não é mais necessário memorizar uma sintaxe abstrata para criar algoritmos. Assim, as principais fontes de erros em algoritmos gráficos são erros lógicos.

Uma característica da programação gráfica é a combinação de nós (módulos de programa gráfico) para criar novas funcionalidades adequadas à tarefa desejada. Em alguns sofwares que utilizam a programação visual, como o Dynamo, os nós de códigos textuais (definidos na Secção 3.2.3) podem ser usados como um substituto para os nós predefinidos do programa, apenas para ser mais rápido ou para resolver tarefas especiais nas quais os nós existentes não

são suficientes ou inexistentes. Depois que um nó de código textual é criado, pode ser convertido num nó gráfico com a mesma funcionalidade.

A programação gráfica pode ser comparada ao desenvolvimento de *software* baseado em componentes. Os componentes podem ser criados com programação tradicional ou através da combinação de nós existentes num nó personalizado [47].

2.5.1 Programação visual e BIM

O processo de *design* utilizado na programação baseada em texto ou programação gráfica também é conhecido como *design* computacional. "*A computação é o procedimento de calcular, ou seja, determinar algo por métodos matemáticos ou lógicos [...]. Trata-se de racionalização, raciocínio, lógica, algoritmo, dedução, indução, extrapolação, exploração e estimativa*" [48].

No contexto do BIM, um Script de programação visual usa as informações de um modelo, seus objetos e outras fontes externas para gerar novas informações com algoritmos computacionais.

Os *Scripts* podem usar os dados para "gerar novas soluções de *design*" e "*analisar, avaliar e otimizar as decisões de design selecionadas relacionadas à forma e ao desempenho de um edifício*" [49].

Semelhante à transição de CAD 2D para BIM, trabalhar com programação visual e dados "exigirá novas atitudes, fluxos de trabalho e especialização em uma tradição que luta contra a mudança" [50].

Atualmente, o ambiente de programação gráfica mais famoso da indústria de Arquitetura, Engenharia e Construção (AEC) é o Grasshopper para o Rhinoceros (Rhino). O Rhinoceros tem fortes capacidades para uso da geometria e o Grasshopper tem uma biblioteca madura de centenas de nós com funções exclusivas.

De acordo com Shu [51], linguagens de programação gráfica modernas, como Grasshopper, podem ser classificadas como linguagens de alto nível, porque a linguagem "requer menos etapas, e com menos detalhes do usuário" e são amplamente aplicáveis, pois podem "gerenciar operações multitarefas".

Algumas interfaces de programação gráfica que podem ser usadas no sector da AEC:

- Componentes Gerativos;
- Grasshopper;
- Hipergrafo (Maya);
- Dynamo;
- Marionete;
- Flux.

A interface de programação gráfica de interesse neste trabalho é o Dynamo, cujo funcionamento é definidos na Secção 3.2.2.

3. DESENVOLVIMENTO DA MODELAÇÃO AUTOMATIZADA

Neste capítulo descreve-se o processo proposto de design desenvolvido ao longo desta dissertação. Este procedimento semiautomático destina-se à análise de vigas-paredes de betão armado, através da obtenção de um modelo de escoras e tirantes adequado ao elemento estrutural e posterior pré-dimensionamento das respetivas escoras e tirantes.

3.1 Metodologia adotada

Tanto a programação Python quanto a visual são ferramentas essenciais no processo de design. Essas ferramentas funcionam para semiautomatizar o trabalho e diminuir a duração do projeto. Isso permite que mais iterações sejam analisadas sempre em busca de um design mais inteligente. Entretanto, o material disponível na literatura para realizar a otimização estrutural através do Dynamo e do Robot é pouco e insuficiente. A interação com Robot por meio do Dynamo ainda não é amplamente usada devido ao número limitado de nós disponíveis e à curva de aprendizado acentuada para trabalhar diretamente na Robot - Application Programming Interface (API). A API Robot é acedida somente através da programação, seja ela em Python, C++, C#, Visual Basic, etc. Devido à complexidade das tarefas realizadas nesta dissertação, o uso da API para interagir com Robot fez-se essencial e, como consequência, houve um aumento significativo no grau de dificuldade do trabalho.

Para justificar a metodologia adotada neste trabalho, o processo de desenvolvimento será brevemente resumido a seguir.

Inicialmente, pretendia-se utilizar a ferramenta Optimo - disponível para o Dynamo - tanto na obtenção do modelo de escoras e tirantes quanto no pré-dimensionamento dos seus elementos. Entretanto, devido às diversas dificuldades encontradas na introdução do código Python, responsável pela obtenção do modelo STM, dentro dos nós do Optimo, decidiu-se desenvolver, de forma separada, um código Python para obter o modelo de escoras e tirantes e utilizar o Optimo apenas para o pré-dimesionamento das bielas.

Para a obtenção do modelo de escoras e tirantes recorreu-se aos conceitos de otimização estrutural, mais precisamente de estruturas treliçadas, uma vez que, o Structural Analysis for Dynamo não fornece acesso à metodologia de análise pelo método dos elementos finitos.

Definida a metodologia a ser utilizada para obter o modelo de escoras e tirantes, surgiram novas dificuldades. Apesar de acessível por meio da programação, a API do Robot não possibilita o acesso a todas as suas funcionalidades, como por exemplo, às matrizes de rigidez geradas no processo de cálculo da estrutura. Desta forma, não se pode modificá-las e, consequentemente, utilizá-las como base para a otimização da topologia das estruturas. Essa particularidade do *software*, de certa forma, acabou por condicionar o trabalho desenvolvido. A solução proposta para ultrapassar essas limitações consiste no desenvolvimento de um procedimento para otimização da topologia de estruturas treliçadas com base nas tensões atuantes e no número de barras que se conectam em cada nó.

Contextualizada a metodologia e as tomadas de decisão necessárias para desenvolvê-la, pode-se dividir o trabalho em duas etapas. A primeira etapa consiste na inserção dos dados de entrada, criação da geometria (malha) no Dynamo e no Robot, cálculo da estrutura e análise dos resultados obtidos. É nesta etapa que o modelo de escoras e tirantes é definido. Já a segunda etapa consite na obtenção, por meio da otimização com algoritmos genéticos, das áreas das secções das escoras e dos tirantes que atendam aos critérios de verificação regulamentares.

Deve-se salientar que o processo de criação é transparente, o que permite ao utilizador visualizar todas as etapas e fazer alterações nos gráficos/*Scripts* do *workflow*.

O Quadro 1 apresenta de forma resumida as etapas e as ferramentas utilizadas no desenvolvimento deste procedimento, desde a criação dos elementos no Dynamo até ao processo de otimização genética para o pré-dimensionamento do modelo de escoras e tirantes. Grande parte dos códigos criados neste trabalho podem ser encontrados no Apêndice 1.

Toda a informação contida no Quadro 1, está sintetizada em forma de organograma na Figura 12.

Quadro 1: Visão geral de gráficos e códigos desenvolvidos.

Descrição	Ferramenta
1. Criar uma malha paramétrica	Dynamo
Cria uma malha de barras e nós com base nos dados de entrada	
1.1 Criar modelo analítico a partir do modelo paramétrico	Dynamo
Crie o modelo analítico no Robot	
1.2 Cálculo	Robot
Executa o cálculo da estrutura no Robot	
2. Resultados	Python/Dynamo
Utiliza o Python para extrair os resultados	
2.1 Avaliação dos Resultados	Python/Dynamo
Utiliza o Python para avaliar os resultados e obter o modelo de escoras e tirantes	
3. Início do processo de otimização genética	Dynamo
Dimensionamento das bielas e dos tirantes	
3.1 População inicial	Dynamo
Usa valores iniciais para gerar a primeira população de soluções (pais)	
3.2 Loop de geração e classificação	Dynamo
Cruza os pais da lista de população inicial com a pontuação do Fitness e as opções ideais de filhos são escolhidos como os novos pais	
4. Envio do modelo final de escoras e tirantes para o Revit	Python

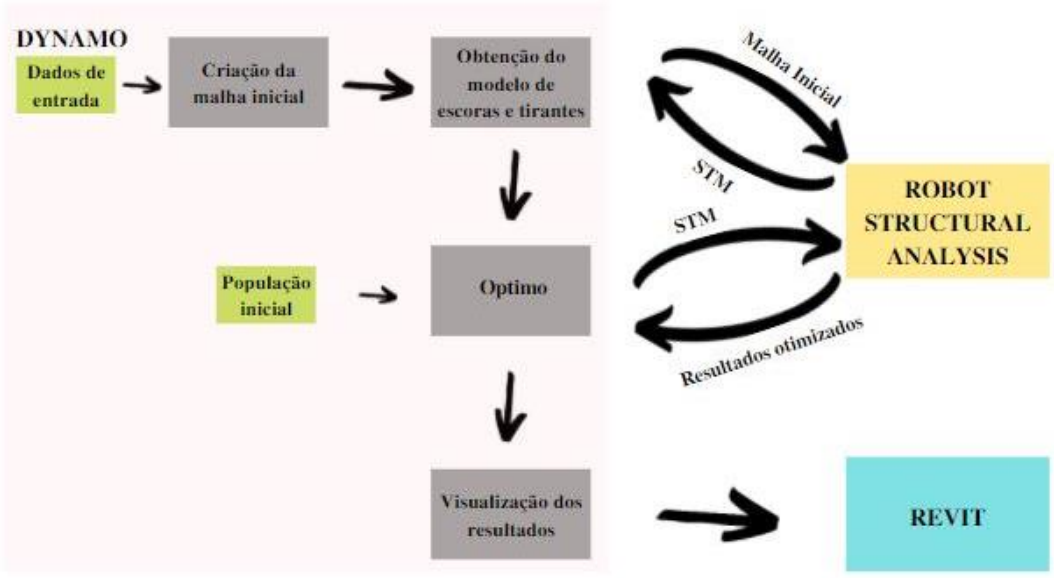


Figura 12: Organogrâma do trabalho desenvolvido.

Já no Quadro 2, está retratado de forma detalhada a primeira etapa do *workflow* do Dynamo, desde a inserção dos dados até à obtenção do modelo de escoras e tirantes (Figura 13), correspondente aos itens 1 e 2 do Quadro 1.

Quadro 2: Processo de criação da malha inicial até a obtenção do modelo de escoras e tirantes.

1. Preparação do modelo
São definidos pelo utilizador os valores de entrada para os diferentes dados utilizados na criação da geometria
2. Geração da malha de pontos
É criada uma malha de pontos com base nos dados iniciais
3. Geração da malha de barras
É criada uma malha retangular de barras de forma a conectar todos os pontos
4. Criação dos apoios, secções e ligação das barras
Estas propriedades devem ser criadas antes de serem atribuídas a estrutura
5. Geração do modelo de cálculo
É gerado um modelo completo para análise estrutural em 4 etapas
5.1 Gerar os nós analíticos no Robot
São criados nós analíticos a partir da malha de pontos citada em 2
5.2 Gerar barras analíticas no Robot
São criadas barras analíticas a partir da malha de barras citadas em 3
5.3 Atribuir apoios
São atribuidos aos nós desejados da estrutura os apoios criados
5.4 Atribuir secções, ligações e materiais
São atribuidas as propriedades às barras analíticas
5.5 Criar os casos de carga e as cargas
São gerados os tipos de carregamentos e o valor das cargas para o modelo
6. Calcular o modelo
É iniciado o cálculo da estrutura
7.Modificar o modelo
São extraídos os resultados e em função dos dados inseridos pelo utilizador, com o uso da programação Python, a estrutura é avaliada para verificar a possível exclusão das barras
8. Guardar o modelo
É guardado o modelo gerado num arquivo “.rtd”

Na Figura 13 é possível visulizar as etapas enumeradas e descritas no Quadro 2 diretamente no nó do Dynamo desenvolvido para tal. Percebe-se que o referido nó é composto por uma série de outros nós que em conjunto executam as atividades propostas.

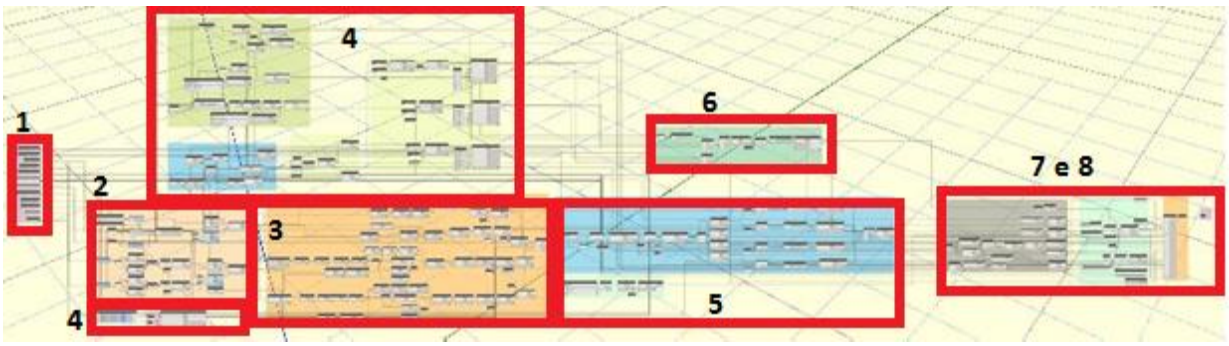


Figura 13: Detalhe da criação da malha inicial até a obtenção do modelo de escoras e tirantes.

3.2 Caracterização dos *softwares* utilizados

Para o desenvolvimento do procedimento utilizam-se três *softwares* (Quadro 3), o Revit, o Dynamo (extensão do Revit) para programação visual, programação em Python e otimização, enquanto o Robot é utilizado como ferramenta para análise estrutural.

Quadro 3: *Softwares* utilizados.

Software	Versão
Autodesk Revit	2021
Dynamo	2.5
Autodesk Robot Structural Analysis Professional	2021

Deve-se salientar que o Python não é propriamente um *software*, entretanto, é uma ferramenta útil que possibilita interagir com os demais *softwares* utilizados.

De acordo com Borges [52], o Python é uma linguagem de altíssimo nível orientada para objetos, de tipagem dinâmica e forte, interpretada e interativa. A linguagem foi criada em 1990 por Guido van Rossum, tendo como foco original utilizadores como físicos e engenheiros. O Python baseou-se noutra linguagem existente na época chamada ABC.

O Python funciona como uma linguagem de programação multiparadigma, usada para muitos propósitos gerais em uma variedade de campos, incluindo, mas sem limitar-se à Matemática, Data-Science e desenvolvimento de web. Além de utilizada para o desenvolvimento de sistemas, é muito utilizada como linguagem *script* em vários *software*s.

Nesta dissertação, o Python foi utilizado como ferramenta para superar as limitações da programação visual com o Dynamo, de forma a possibilitar o acesso ao Robot- Application Programming Interface (API). Como a API pode ser complicada para não programadores, e não há na literatura qualquer tipo de documentação para auxiliar no acesso à API via linguagem Python, a comunidade Dynamo e o suporte da Autodesk tornaram-se necessários e essenciais para o desenvolvimento desta dissertação. Os aspetos mais relevantes do código desenvolvido estão descritos no APÊNDICE 1.

3.2.1 Revit

O Revit é um *software* de modelação 3D que permite aos utilizadores modelar diferentes elementos de uma construção. O programa assenta na utilização de uma base de dados que descreve todos os objetos que compõem um edifício, a qual "alimenta" de forma paramétrica a representação dos diferentes géneros de objetos. Esta nova tecnologia paramétrica em BIM não assenta na representação dos objetos por si próprios, mas, sobretudo, na identificação de relações entre eles, de forma que, quando uma alteração é efetuada num determinado elemento construtivo, a mesma seja automaticamente refletida nos outros elementos afetados.

A tecnologia BIM do Revit não tem por base a execução direta de desenhos (como no CAD), mas sim a criação de um modelo tridimensional que represente o edifício, não só do ponto de vista gráfico, mas também da caracterização de todos os aspetos que possam ser relevantes para a sua construção e gestão.

A partir do modelo podem ser obtidas todas as peças desenhadas (plantas, cortes e alçados) e as listagens de componentes que possam vir a ser necessárias para a avaliação/quantificação dos trabalhos. Além disso, a existência de um modelo 3D permite aos projetistas uma melhor avaliação dos componentes construtivos e do espaço, o que facilita tanto a análise de soluções alternativas quanto a comunicação entre todos os intervenientes no projeto.

O ambiente de trabalho deste *software* encontra-se representado pela Figura 14.

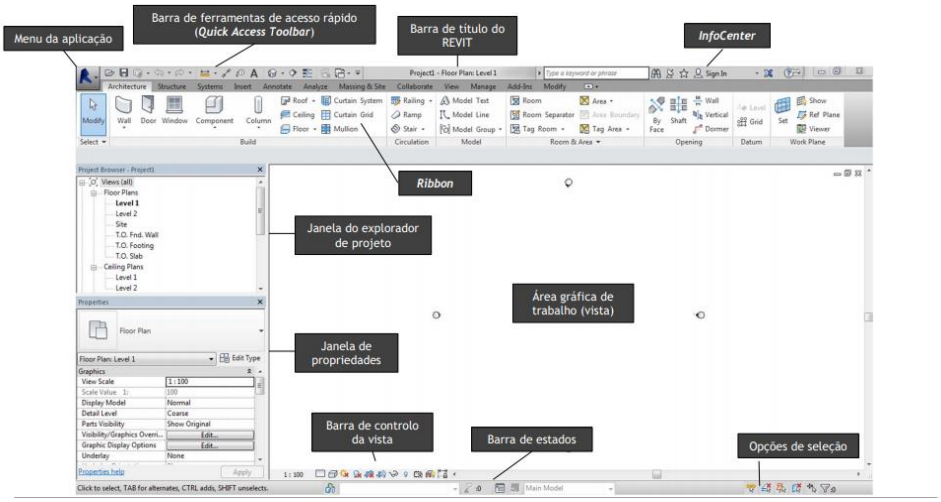


Figura 14: Ambiente de trabalho do Revit.

3.2.2 Dynamo

O Dynamo é um *software* de programação visual que permite criar rotinas. Em comparação com outros *softwares*, tem a vantagem de não ser necessário proceder à escrita de dezenas ou centenas de linhas de código. O que torna a interface gráfica com o utilizador muito mais iterativa, ou seja, substitui a complexa sintaxe da linguagem de programação por blocos e/ou nós, conectados entre si por *strings* virtuais [53], conforme esquematizado na Figura 15.

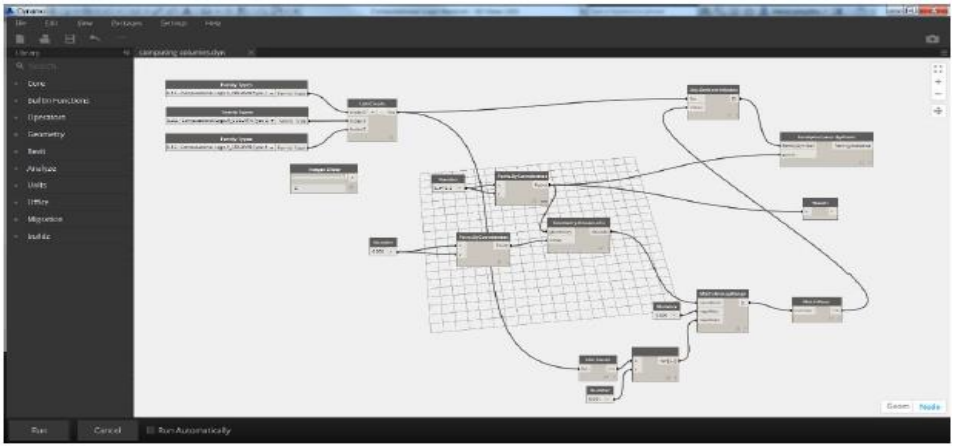


Figura 15: Aréa de trabalho do Dynamo.

A programação no Dynamo é realizada através de nós. Esses nós podem ser criados manualmente através da linguagem Python ou, mais frequentemente, usados como nós predefinidos, integrados pelo Dynamo ou criados por terceiros, para representar vários comandos e funcionalidades. Na Figura 16 temos a explicação simples do funcionamento de um nó. O nó neste exemplo é o "Point.ByCoordinates" nó. Cada nó consiste em entradas e saídas. Neste caso, o X, Y e o Z são entradas que podem ser conectadas a nós numéricos e a saída é onde o ponto estará localizado na visualização 3D [54].

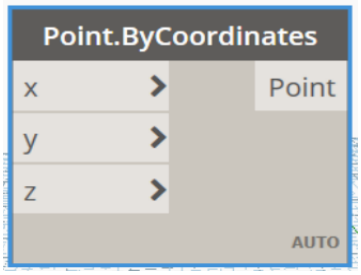


Figura 16: Exemplo de nó do Dynamo.

O Dynamo é um *plugin* do Revit, o que facilita a ligação e a interoperabilidade entre estas duas ferramentas. O Dynamo está conectado ao Revit de forma que se o utilizador mudar algo no Revit, isso muda no Dynamo também e vice-versa. Por exemplo, todos os elementos que existem no Revit, o Dynamo pode lê-los diretamente e ser capaz de usá-los [54].

Existem vários pacotes disponíveis para descarregar na página *dynamopackages* [55] com uma vasta gama de utilidades. Para a realização deste trabalho foram utilizados vários pacotes, dentre os quais se destacam o Structural Analysis for Dynamo e o Optimo.

3.2.3 Optimo

O Optimo é um pacote para o Dynamo desenvolvido por Rahmani et al. [56]. Optimo é uma Otimização Multi-Objetivo (MOO) baseada em programação visual de código aberto para operações baseadas em BIM por meio da interação com o Dynamo. Esta ferramenta foi publicada em 2015 na biblioteca Dynamo. O Optimo funciona com base em uma versão modificada do Nondominated Sorting Genetic Algorithm-II (NSGA-II), conforme apresentado na Secção 2.4.1.2. O mecanismo de funcionamento do algoritmo consiste na criação de um conjunto de

números aleatórios de um intervalo e usa-os como pais para o algoritmo de otimização, baseado na teoria da evolução. Em cada iteração, os filhos ideais criados são usados como pais que trabalham em direção à solução ideal, desde que o critério de paragem ainda não tenha sido alcançado. Geralmente, a configuração do Optimo pode ser descrita em cinco etapas, como pode ser visto na Figura 17.

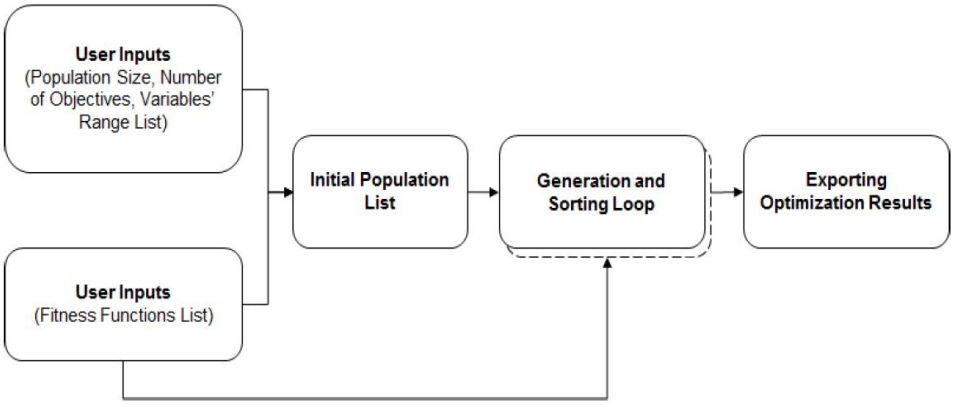


Figura 17: Configuração geral do Optimo [47].

As entradas do utilizador contêm o tamanho da população, número de objetivos e variáveis. O tamanho da população é o número de valores aleatórios criados entre os limites superior e inferior. Quanto maior o tamanho da população, melhor será o resultado. O tempo da otimização está diretamente relacionada à capacidade computacional, uma vez que, quanto maior a população, maior é o número de iterações necessárias. De acordo com Rahmani et. al [56] não há limite para o tamanho da população inicial. O número de objetivos define o número de funções de aptidão que devem ser incluídas na otimização. Cada um dos objetivos conterá um nó personalizado e uma função de adequação. As variáveis são o intervalo de soluções, que variam do limite inferior ao superior. A Fitness Function (função de avaliação) é uma lista que contém as funções de aptidão criadas com os nós personalizados no Dynamo. O *Fitness* deve ser entendido como um valor único que representa a adequação geral da solução, o que permite ao utilizador usar funções externas sem a necessidade de alterar o código-fonte de otimização.

- Lista de População Inicial

A população inicial é uma lista de valores entre os limites superior e inferior. Esses valores iniciais são usados para gerar aleatoriamente a primeira população de soluções que atribui os valores de *Fitness* à lista de população.

- *Loop* de geração e classificação

Esta é a parte central do Optimo, o *loop* de geração e classificação processa até atingir a contagem de iterações definida pelo utilizador. Os pais da lista de população inicial são cruzados com a pontuação do *Fitness* e as opções ideais de filhos são escolhidos como os novos pais pelo algoritmo.

3.2.4 Robot structural analysis

O Robot Structural Analysis (Robot) é um *software* da Autodesk usado para efetuar a análise estrutural automática de modelos com a capacidade distinta, em comparação com outro *software* de natureza semelhante, de interagir com outros aplicativos da Autodesk frequentemente usados no sector, como o AutoCAD ou o Revit. A Figura 18 contém a área inicial de trabalho do Robot.

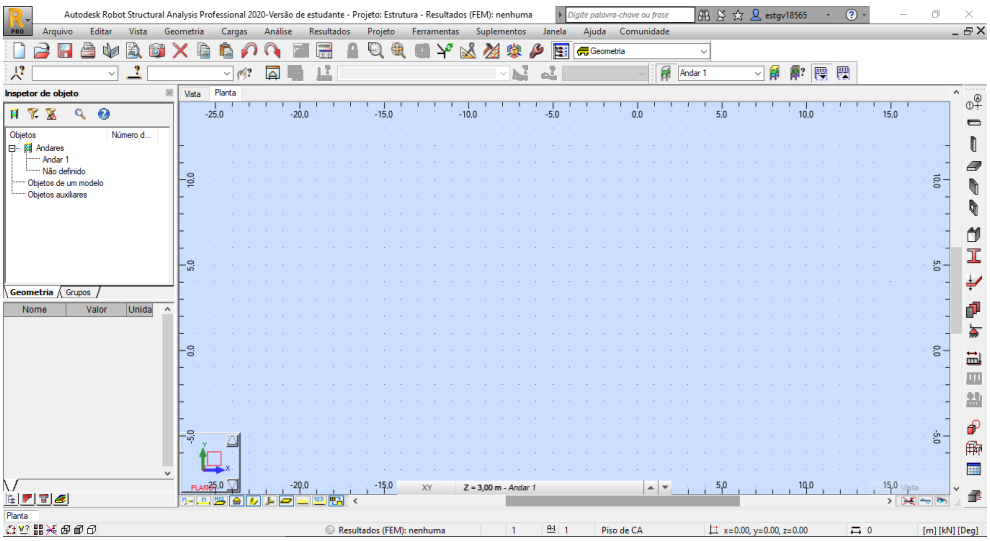


Figura 18: Área de trabalho do Robot Structural Analysis.

O Robot oferece acesso total a uma aplicação aberta - API (Application Programming Interface). A API Robot Structural Analysis Professional possibilita o controlo programático através de *softwares* externos.

A API Robot Structural Analysis Professional permite o controlo da geração da geometria, geração da malha, atribuição do material estrutural e propriedades seccionais, execução da

análise e avaliação de resultados, ajudando a fornecer uma redução rápida no tempo de iteração do projeto [57].

A API usa a tecnologia de modelo de objeto de componente (Component Object Model - COM) da Microsoft. Isso significa que o utilizador pode desenvolver o código em Visual Basic para Aplicativos (VBA) de dentro de aplicativos como o Excel, o Dynamo, o Microsoft Word, o AutoCAD ou desenvolver aplicativos .Net sofisticados para vincular o *software* de análise do Robot a *software*s externos [57].

O *software* possui informações incorporadas e verificações codificadas ditadas por diferentes normas internacionais.

3.2.5 Limitações

1. Dynamo

Como o Dynamo é um programa dependente do Revit, os requisitos de *hardware* são substanciais. Foi descoberto que até 3 GB de RAM são usados para executar gráfico. Se um gráfico rodar por muito tempo, haverá aumento na quantidade de RAM usada como também observado por [30].

2. Optimo

Foi observado um problema entre o Dynamo, a ferramenta Optimo e o Robot. A complexidade da execução é limitada pela quantidade de RAM disponível. A seguinte correlação foi observada: 8 GB de RAM ≈ 50 runs enquanto 16 GB de RAM ≈ 100 runs. A única solução para isso é guardar a população atual e reiniciar o Robot, o Dynamo e o Revit e usar a população guardada como população inicial. Isso aumenta o desenvolvimento e tempo de execução [1].

3. Robot

Os nós lançados pela Autodesk para o Robot são predeterminados e limitados na interação com o Robot. Isso reduz a possível complexidade dos modelos estruturais disponíveis. Nesta dissertação, o autor propôs soluções alternativas menores que fazem uso de Python *Scripts* para interação direta com a API Robot. No entanto, atualmente, não existe documentação para o uso de Python e da API Robot, o que torna o processo de Script muito demorado, pois é executado por tentativa e erro.

A interação Robot/Dynamo é sensível ao tempo. Como a estrutura está a ser criada e calculada no Robot diretamente através do Dynamo, o tempo entre as diferentes ações é essencial, pois, por exemplo, não é possível definir apoios que ainda não foram criados ou adicionar cargas a uma barra que não existe [1].

4. Tempo de Execução

O tempo necessário para a execução de um gráfico de otimização depende da complexidade da estrutura. Estruturas complexas podem levar vários minutos para serem geradas no Robot. Ao combinar o fluxo de trabalho em um nó personalizado, esse tempo pode ser reduzido. Com o acesso a mais poder de processamento, esse tempo pode ser ainda mais reduzido [1].

5. Interoperabilidade

O *workflow* desenvolvido nesta dissertação só poderá ser executado se respeitada a ordem descrita no Quadro 4, caso contrário, a interoperabilidade entre os *software*s poderá não funcionará corretamente.

Quadro 4: Passo a Passo para executar os programas necessários.

1. Executar o Revit
2. De forma independente, executar o Robot Structural Analysis
3. Aguardar a abertura do Robot Structural Analysis
4. Novamente no Revit, selecionar a aba “Manage” e então o ícone do Dynamo
5. Aguardar a abertura do Dynamo
6. No Dynamo, selecionar o arquivo “.dyn” que se deseja abrir

3.3 Geração do modelo

3.3.1 Criação da geometria

Para que seja possível realizar a análise estrutural no Robot, primeiramente, deve-se criar a geometria da estrutura desejada no Dynamo, para só então criá-la no Robot. A geometria inicial é uma malha composta por nós ligados por uma série de barras verticais, horizontais e diagonais.

Para a criação da malha inicial, alguns parâmetros devem ser inseridos pelo utilizador por meio de nós de entrada. Os parâmetros escolhidos como dados de entrada, e, portanto, variáveis, são: altura, comprimento, recobrimentos e formato da malha. Entretanto, há ainda outros dados variáveis que devem ser preenchidos pelo utilizador, como: propriedades dos materiais (betão e aço), a posição e a magnitude dos carregamentos.

O betão e o aço são utilizados para obtenção de valores de referência que, por sua vez, serão utilizados como parâmetros de avaliação do dimensionamento da estrutura, conforme descrito no Capítulo 4. Os recobrimentos são utilizados para obtenção da altura útil, a. A altura utíl é obtida descontando-se duas vezes o valor do recobrimento nas faces verticais, uma vez que a malha inicial deve estar posicionada dentro dos limites da secção geométrica da viga-parede. Já os demais parâmetros são utilizados para a criação da malha em si. A Figura 19 contém alguns dos parâmetros escolhidos como dados de entrada.

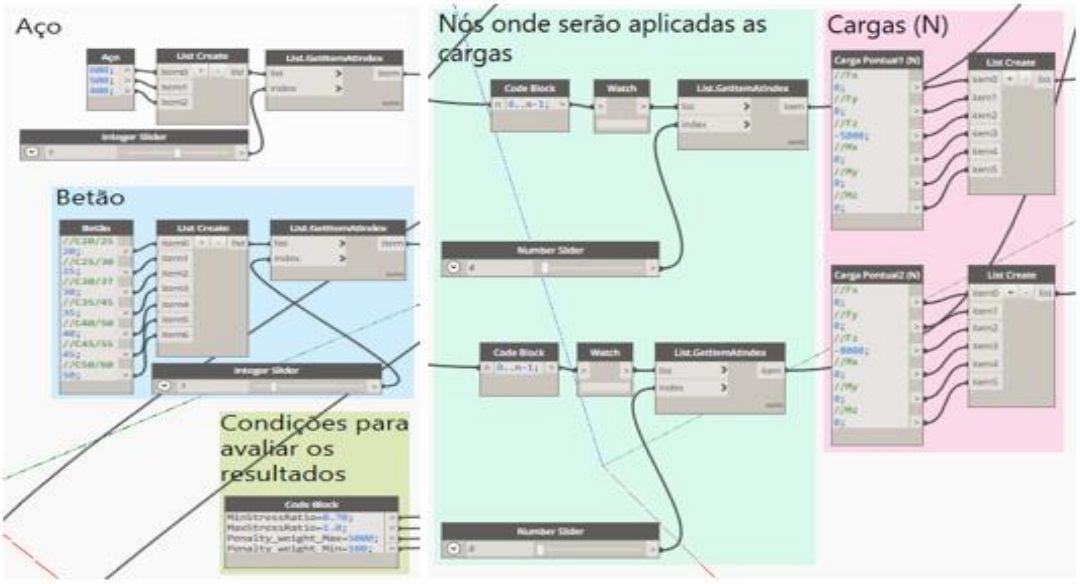


Figura 19: Nós de entrada.

Definidos os valores de entrada, o programa está pronto para ser executado. Logo após a execução, os dados inseridos pelo utilizador passam por um nó personalizado criado com o intuito de garantir que a malha a ser gerada atenderá os critérios de dimensões necessários para ser considerada uma viga-parede, e que as barras componentes da malha atendam aos critérios de inclinação dos modelos de escoras e tirantes descritos no capítulo seguinte. Caso algum destes critérios não seja atendido, os dados iniciais não são transferidos para a próxima etapa e, no nó

personalizado aparece um alerta visual que sugere ao utilizador a alteração dos dados de entrada. Se atendidos todos os critérios, os dados seguem e dá-se início à criação da malha.

3.3.2 Modelo analítico no Robot

A Autodesk possuí um pacote chamado Structural Analysis for Dynamo com alguns nós predefinidos que trabalham para criar um *link* entre o Dynamo e Robot. Os nós são simples e podem lidar com as tarefas mais comuns e com a criação de objetos do Dynamo no Robot. Entretanto, como citado no início desta secção, quando tarefas mais avançadas se fazem necessárias, como no caso desta dissertação, é preciso trabalhar diretamente com a API Robot através da linguagem Python.

Após a execução do *workflow*, as informações do Dynamo são transferidas ao Robot. Este processo pode demorar alguns minutos, de acordo com a complexidade da estrutura. Para evitar o surgimento de erros no envio da informação, foram criados e adicionados Python *Scripts* ao longo de todo o *workflow* com o intuito de criar uma pausa de alguns segundos no processo de transferência. O tempo de pausa e o local de inserção podem ser ajustados de acordo com as necessidades do utilizador.

3.3.3 Secção, material, apoios e cargas

Na Figura 20 está representado o *workflow* no Dynamo para criação das condições de contorno e também da geometria no Robot. A explicação do *workflow* encontra-se no Quadro 5.

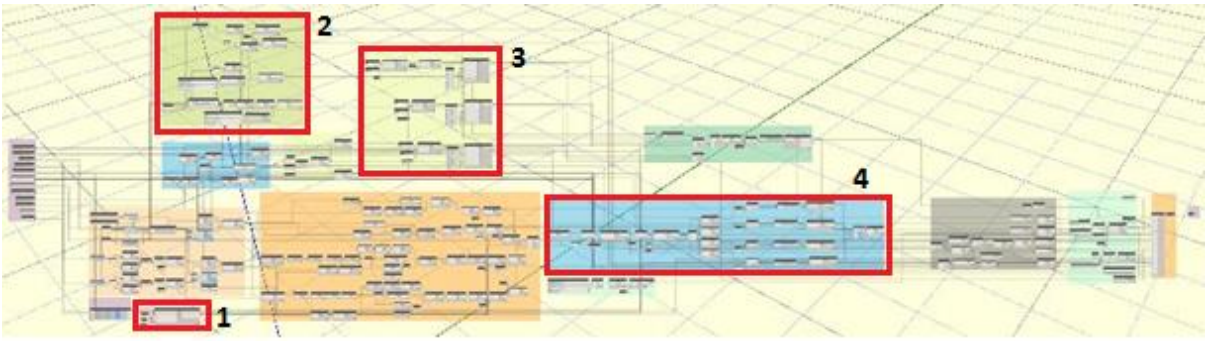


Figura 20: Criação das condições de contorno e da geometria no Robot.

Quadro 5: Etapas no Dynamo para a criação das condições de contorno e da geometria no Robot.

1. Criação das secções no Robot
2. Criação dos apoios da estrutura
3. Criação dos casos de carga
4. Criação das barras no Robot e atribuição das secções, materiais, etc.

As secções inicialmente atribuídas às barras dependem dos dados de entrada fornecidos pelo utilizador, e, portanto, são variáveis. A secção inicial é utilizada para obter as tensões atuantes em cada uma das barras. A partir das tensões obtidas, adota-se um critério de verificação para a classificação das barras como possíveis ou não de exclusão. Para isso é preciso definir essas secções no Robot para somente então atribuí-las às barras desejadas. E para tal, criou-se um nó personalizado onde, com o auxílio da linguagem Python, é feito o acesso diretamente à API Robot e através dela as secções são criadas. Na Figura 21 é possível visualizar os nós que compõem o nó de criação das secções e também o Python *Script*.

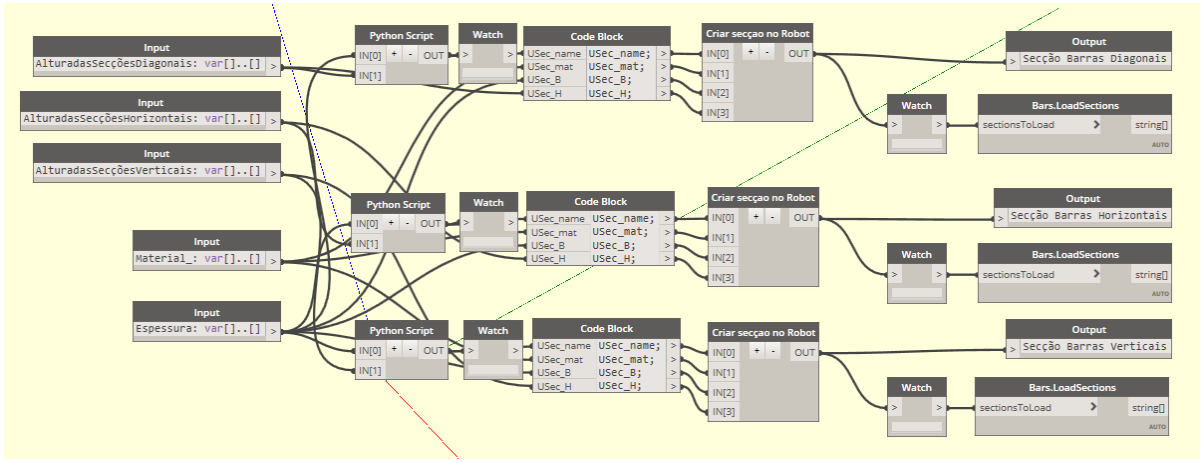


Figura 21: Conteúdo do nó de criação das secções no Robot.

Os apoios são introduzidos nos nós a partir dos modelos padrão do Robot, como se pode ver no Quadro 6. Se outro tipo de apoio ou ligação for necessário, assim como as secções, é necessário criá-los no Robot antes de atribuí-los à estrutura. Como o *workflow* do Dynamo se destina à criação semiautomática de modelos, deve-se novamente fazer uso da API Robot.

Quadro 6: Modelos padrão de apoios e ligações disponíveis no Robot.

Apoios	Ligações
Rotulado	Rotulado-Rotulado
Fixo	Fixo-Rotulado
	Rotulado-Fixo

Na Figura 22 é possível visualizar o nó personalizado para criação de um tipo de apoio e também um *code block* com as informações necessárias para tal.

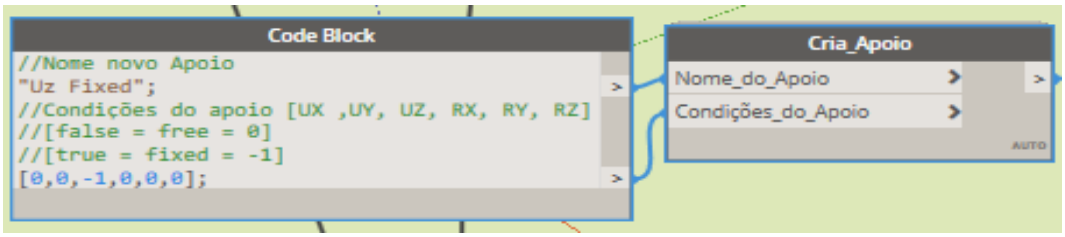


Figura 22: Code Block com os dados para criação de um novo apoio no Robot.

A Figura 23 contém o nó personalizado para criação de um tipo de ligação, um *Code Block* com as informações necessárias e os nós necessários para enviar a informação para a próxima etapa do *workflow*.

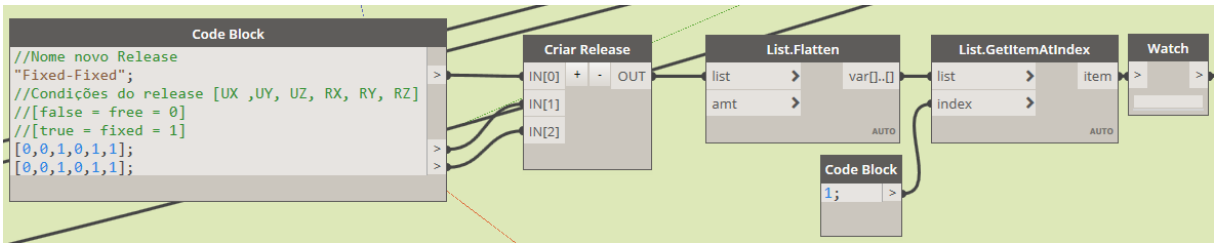


Figura 23: *Code Block* com os dados para criação de um novo apoio no Robot.

Os casos de carga e os carreamentos, são criados através de nós pré-definidos existentes dentro do pacote Structural Analysis for Dynamo. Como se trata de uma estrutura composta por barras e nós, as cargas são do tipo nodal e aplicadas no máximo em dois nós escolhidos pelo utilizador. Além dos nós, o utilizador também deve definir o valor do carregamento a ser aplicado.

É importante ressaltar que existe um problema de tempo entre o Dynamo e o Robot, conforme descrito na Secção 3.2.5, e, portanto, além do Python *Script* utilizado para parar a transferência de informação, é necessário também fazer uso de um nó que atrase a atribuição das ligações, apoios ou secções recém-criadas. O atraso é realizado por meio do uso do comando "{waitfor, pass} [1]" dentro de um *Code Block* (Figura 24). Este código faz com que o programa aguarde a entrada de "*waitfor*" antes de dar continuidade aos dados que constam em "*pass*". Isso garante a criação dos elementos como secções, ligações, etc., antes que estes sejam atribuídos as barras, o que previne possíveis problemas na execução do *workflow*.

Figura 24: *Code Block* utilizado para atrasar a execução do *workflow*.

3.4 Cálculos, extracção e análise dos resultados

As etapas de extração dos resultados, análise e a exclusão das barras, foram feitas com auxílio da programação em linguagem Python e estão contidos dentro de um nó próprio para Pyhton *Scripts*. Este *Script* foi criado para executar todas as etapas de forma automática, e encontra-se parcialmente disponível no Apêndice A.

3.4.1 Cálculos

A etapa de cálculo é realizada por meio do nó pré-definido “Analysis.CalculateWithSave” do Structural Analysis for Dynamo, o qual deve ser alimentado com as seguintes informações: barras analíticas, apoios, casos de cargas e as cargas a serem aplicadas na estrutura.

Para que não haja nenhum erro no processo de cálculo, as informações descritas na Secção 3.3.3 devem estar todas presentes no Robot. Portanto, faz-se uso novamente do Python *Script* que atrasa o processo de transmissão dos dados, para garantir que toda a estrutura já esteja pronta antes de iniciar o processo de cálculo.

3.4.2 Extração e análise dos resultados

Após utilizar o nó "Analysis.CalculateWithSave", é possível extrair os resultados desejados através dos nós disponíveis no Structural Analysis for Dynamo, por exemplo, com o uso do nó "Barstresses.GetMaxValuesList". Os resultados são recebidos no Dynamo em unidades do Sistema Internacional - SI (definido internamente no Robot).

Entretanto, para tornar o *workflow* mais rápido e facilitar o processo de avaliação dos resultados, optou-se por extraí-los através da API Robot. Após a obtenção dos esforços atuantes em cada barra da estrutura, inicia-se então o processo de avaliação destas. Assim como na extração dos resultados, o processo de avaliação foi feito com auxílio da programação em Python.

A primeira etapa de avaliação consiste em separar as barras em conjuntos de acordo com os esforços a que estão submetidas, sendo esforços de tração ou compressão. Após separadas, inicia-se o processo de filtragem das barras. Para tal, a tensão a que a barra está submetida é comparada com um valor de referência. Este valor de referência, para as barras comprimidas, é obtido de acordo com a Equação (1), conforme prescrições do Eurocódigo 2 [58] e da ABNT NBR 6118:2014 [18]:

$$f_{cd} = \frac{f_{ck}}{\gamma_c} \tag{1}$$

Onde:

fcd = valor de cálculo da resistência do betão à compressão;

fck = valor característico da resistência do betão à compressão;

γ_c *= coeficiente parcial relativo ao betão.*

Para as barras sujeitas a forças de tração, o valor de referência é obtido de forma similar, considerando-se o valor de cálculo da tensão de cedência à tracção do aço, conforme descrito na Equação (2), conforme prescrições do Eurocódigo 2 [58] e da ABNT NBR 6118:2014 [18], considerando-se a resistência à tração do betão como sendo nula.

$$f_{syd} = \frac{f_{syk}}{\gamma_s} \tag{2}$$

Onde:

fsyd = valor de cálculo da tensão de cedência à tracção do aço das armaduras para betão armado;

fsyk = valor característico da tensão de cedência à tracção do aço das armaduras para betão armado;

γ_s *= coeficiente parcial relativo ao aço das armaduras para betão armado.*

Este único critério foi adotado em função da impossibilidade, citada no início deste capítulo, em aceder às matrizes de rigidez utilizadas pelo Robot, de modo a garantir que a eliminação de barras não dá origem a fenómenos de instabilidade local ou global da treliça. Além disso, esta etapa tem como finalidade apenas a obtenção de um modelo de escoras e tirantes e não o dimensionamento da estrutura, considera-se que as simplificações adotadas são aceitáveis. Contudo, se for necessário, o utilizador tem liberdade para modificar os valores de referência alterando os coeficientes parciais dos materiais ou os valores característicos de resistência dos materiais, para diferentes condições normativas por exemplo.

As barras que apresentarem tensão menor do que o valor de referência são armazenadas numa lista e, na sequência, serão submetidas aos critérios de avaliação definidos a seguir.

Também foram adotados critérios para garantir a isostaticidade da modelação em análise, uma vez que, ao excluir barras da estrutura, é possível provocar o surgimento de instabilidades globais e/ou locais. Portanto, a exclusão deve ser feita de forma a garantir que a maior quantidade de barras seja excluída sem causar nenhum tipo de instabilidade. Como alternativa a este problema, a próxima etapa do processo de filtragem baseia-se no número de elementos que se ligam aos nós inicial e final de uma determinada barra. A seleção das barras dá-se através dos nós da estrutura, que são selecionados no sentido das extremidades para o centro. Sempre que um nó é selecionado, todas as barras ligadas a ele são analisadas. Se a barra selecionada estiver, com base nos critérios anteriores, apta a ser excluida e possuir seus nós inicial e final conectados a 3 ou mais barras, este elemento torna-se de facto possível de exclusão. Porém, para casos especiais como, por exemplo, nos nós com cargas aplicadas e nos nós restringidos (apoios) o critério anterior não é aplicado. Entretanto, a barra só será excluída se o critério de estaticidade global descrito na Equação (3) for atendido. Desta forma, a possibilidade de surgirem instabilidades é consideravelmente diminuída. É importante referir que todo este processo de exclusão foi obtido de forma empírica através de inúmeros testes, tendo em vista que, critérios mais rigorosos de avaliação da instabilidade local/global não podem ser aplicados devido às limitações do *software* de cálculo anteriormente referidas.

De acordo com os critérios de exclusão aqui estabelecidos, é possível que durante o processo, alguns nós fiquem ligados a somente duas barras. A fim de evitar o surgimento de erros durante

o processo de cálculo, criou-se um *Script* que identifique e exclua estes nós, bem com as barras ligadas a eles. Além disso, crie em seu lugar uma nova barra com as mesmas propriedades das que existam, de maneira a garantir que tanto a estaticidade quanto a topologia da estrutura permaneçam inalteradas.

Terminadas as etapas citadas anteriormente, são excluídos os nós que porventura ainda estejam na estrutura ligados a uma ou nenhuma barra, para que se garanta novamente a condição de estaticidade descrita na Equação (3).

$$\mathrm{NB} + \mathrm{NR} \geq 2 \times NN \tag{3}$$

Onde:

NB = Número de barras;

NR = Número de reações;

NN= Número de nós.

Devido à possibilidade da criação de novas barras durante o processo, foi adicionado ao Python *Script* um código que selecione todas as barras presentes no Robot e lhes atribua o material e as secções já criados anteriormente.

A próxima e última etapa do código consiste em guardar o modelo resultante como arquivo no formato ".rtd". Isto dá ao utilizador a possibilidade, posterior, de acesso à estrutura resultante através do Robot.

3.5 Validação do modelo de escoras e tirantes para secções compostas por um único material

Com o intuito de validar o processo de geração de modelos de escoras e tirantes (sistemas estruturais de treliça) desenvolvido neste trabalho, optou-se, nesta fase, por buscar na literatura técnico-científica exemplos cujo objetivo fosse a otimização da topologia de estruturas treliçadas. Para isso, são adotados dois exemplos descritos a seguir.

3.5.1 Exemplos

Para validação do modelo desenvolvido foram considerados dois exemplos. O Exemplo 1, foi retirado do trabalho desenvolvido por Petrovic et al. [59]. Em seu trabalho, o autor realiza otimizações de topologia, de forma e de secções transversais de um modelo de treliça. Entretanto, para fins de validação deste trabalho, será considerado apenas o resultado obtido no processo de otimização da topologia da estrutura. O modelo inicial de treliça pode ser visualizado na Figura 25.

O material dos elementos de treliça é alumínio 6063-T5 cujas características são: Módulo de Elasticidade igual a 68,947 MPa e peso específico de 26,48 kN/m³. A carga pontual de F = 444,82 kN, é aplicada nos nós 2 e 4, conforme se mostra na Figura 26. O modelo está limitado a um deslocamento máximo de ± 0,0508 m em todos os nós e em todas as direções, a uma tensão normal de ± 172,3689 MPa para todas as barras e a uma área mínima para todas as secções das barras de 0,6426 cm².

A área da secção transversal inicial de todas as barras é de 452,3893 cm². O modelo inicial com essas barras tem um peso de 13019,482 kgf (130.19 kN). Os nós 5 e 6, por se tratarem de apoios não são definidos como variáveis, ou seja, não podem ser removidos.

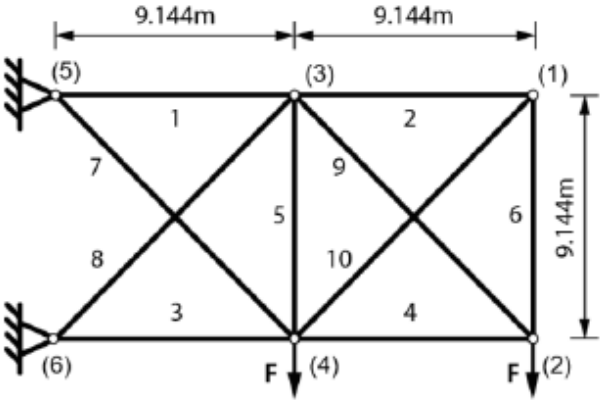


Figura 25: Treliça a ser otimizada [Extraído de 59].

Para a reprodução do exemplo desenvolvido por Petrovic et al. [59], optou-se por adotar secções transversais retangulares com área equivalente às secções circulares utilizadas pelo autor em seu modelo. Como neste exemplo não é verificada a carga critíca de Euller, a forma da secção

transversal da barra não tem influência, bastando garantir a equivalência de áreas. Os demais parâmetros foram reproduzidos com exatidão.

No Exemplo 2, desenvolvido por Faustino et al. [60], a estrutura inicial correspondente consiste em pontos nodais e barras com a geometria representada na Figura 27. Os autores apresentam no seu trabalho diferentes tipos e tamanhos de malhas iniciais e diferentes forças nodais aplicadas. Entretanto, para validação do modelo desenvolvido neste trabalho considerou-se somente o exemplo designado por Pt3, conforme as características a seguir:

- Malha composta por 29 barras e 12 nós.
- As barras possuem secção transversal constante e igual a 3 cm².
- Duas cargas nodais são aplicadas simultaneamente nos nós 3 e 12 (fx3 = −45,9619 kN, fy3 = 45,9619 kN, fx12 = −45,9619 kN, fy12 = −45,9619 kN), em que:
    - fxn é a carga nodal em (kN) aplicada no nó n na direção Ox;
    - fyn é a carga nodal em (kN) aplicada no nó n na direção Oy.
- Os deslocamentos e limites de tensão máximos das barras considerados são ± 50 cm e ± 355 MPa, respectivamente, e o fator de segurança parcial λ é igual a 1,5.

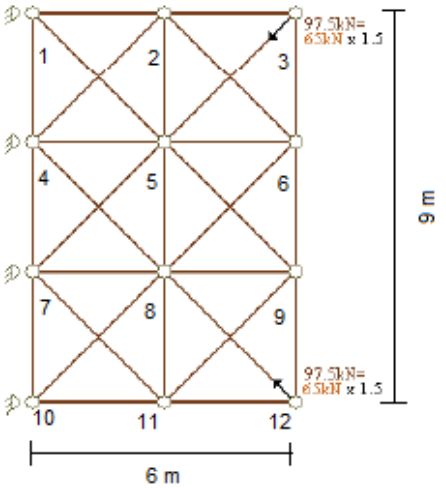


Figura 26: Exemplo 2 - malha inicial [Extraído de 60].

Assim como no Exemplo 1, para a reprodução do modelo desenvolvido por Faustino et al. [60], fez-se necessária uma pequena alteração. Devido à forma como a estrutura é criada no Dynamo, houve um aumento no número de barras e de nós, uma vez que, as barras inclinadas foram

subdivididas e um nó foi adicionado no ponto onde as mesmas se intercetam. Sendo assim, o número de nós passou de 12 para 18 e o de barras passou de 29 para 41. Nesta análise, ao contrário de Faustino et al. [60], não foram considerados os limites de deslocamentos. Os demais parâmetros foram reproduzidos com exatidão.

3.5.2 Resultados

As Figuras 28 e 29 apresentam, respectivamente, as topologias obtidas pelo modelo desenvolvido e pelos autores para os exemplos de validação 1 e 2. Analisando-se os resultados obtidos por este trabalho, em comparação com os resultados obtidos na literatura de referência, é visível a semelhança obtida tanto com relação ao Exemplo 1, quanto ao Exemplo 2. Ressalta-se que ambos os trabalhos de referência utilizaram uma programação baseada na análise dos deslocamentos nodais como um dos critérios de restrição no processo de otimização, o qual não foi possível de ser utilizado neste trabalho.

Na Figura 28 ilustram-se os resultados obtidos para o Exemplo 2.

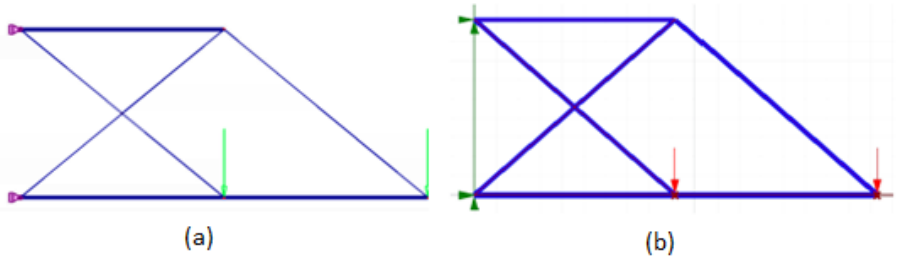
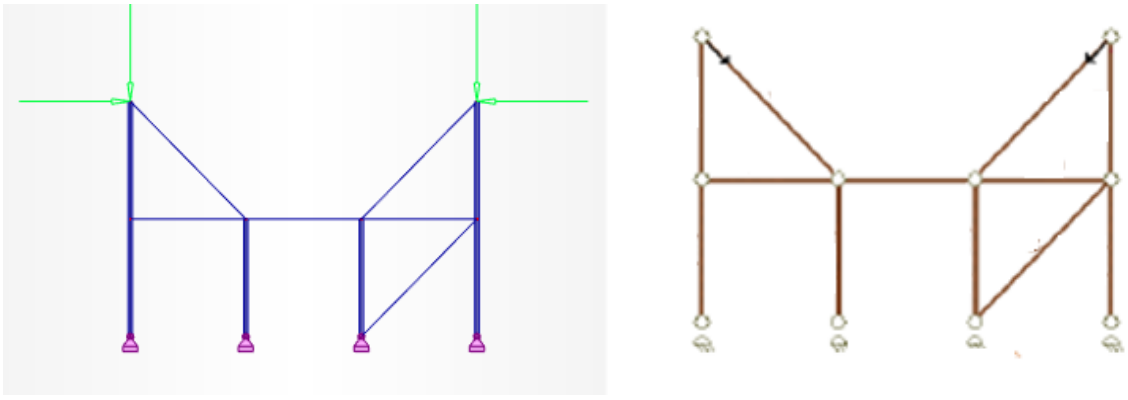


Figura 27: (a) Solução obtida pelo modelo desenvolvido; (b) Solução obtida por [59].



(a) (b)

Figura 28: (a) Solução obtida pelo modelo desenvolvido; (b) Solução obtido por [60].

Pode-se perceber ainda que em ambos os casos, assim como nos exemplos base, a estrutura ideal requer uma quantidade inferior de material e, ainda assim, se mantém cinematicamente estável.

Sendo assim, devido a satisfatória semelhança obtida ao comparar-se os resultados com aqueles disponíveis na literatura, assume-se que o modelo desenvolvido neste trabalho pode ser utilizado para otimização da topologia em outras estruturas treliçadas com as mais diferentes configurações iniciais.

A fase seguinte será considerar este módulo para a obtenção de um modelo de escoras e tirantes que permita o dimensionamento de vigas-parede em betão armado, conforme será abordado no capítulo seguinte.

4. DIMENSIONAMENTO DE VIGAS-PAREDE – CASO DE ESTUDO

Após o satisfatório desempenho obtido com o código Python desenvolvido nesta dissertação na obtenção da otimização da topologia de estruturas treliçadas de aço (Capítulo 3), pretende-se, no decorrer deste capítulo, utilizar o mesmo código para, desta vez, obter modelos de escoras e tirantes (STM) de betão armado com o objetivo de dimensionar vigas-parede. Por fim, após obtido os modelos STM, estes terão seus elementos otimizados através de algoritmos genéticos, de modo a obter as dimensões do elemento de betão e da quantidade de armaduras de aço.

Para tal, apresenta-se em seguida o método baseado nos modelos de escoras e tirantes.

4.1 Método de escoras e tirantes

Um método usualmente sugerido para o projeto de vigas vigas-paredes é o modelo de escoras e tirantes (Strut and Tie Method - STM), que consiste basicamente em prever e projetar uma treliça interna à estrutura. Esta treliça é constituída por tirantes de aço e escoras de compressão de betão que são interligadas em nós, para suportar o carregamento imposto nas regiões de descontinuidade (também denominadas de Regiões D, em que a hipótese da secção plana não mais se aplica, conforme discutido na Secção 2.3). De acordo com o ACI [21], o procedimento de aplicação do STM inclui as etapas gerais resumidas abaixo:

i. Definir os limites da Região D (Figura 7) e determinar as forças locais e seccionais impostas;

ii. Desenhar a treliça de suporte interna, encontrar as cargas equivalentes e calcular as forças nas barras da treliça;

iii. Escolher o aço de reforço para fornecer a capacidade necessária ao tirante e certificar-se de que este reforço de tirante está adequadamente ancorado nas zonas dos nós;

iv. Avaliar as dimensões dos nós e das escoras, de forma que as capacidades desses componentes sejam adequadas para suportar os valores das forças de projeto;

v. Selecionar a armadura distribuída para garantir o comportamento dúctil das Regiões D. O método clássico de dimensionamento do nó é pelo arranjo da forma do nó, de modo que as tensões aplicadas em todos os lados do nó sejam iguais. Organizar o nó nesta forma pode ser feito ao dimensionar os limites do nó para que se tornem proporcionais e perpendiculares às forças que atuam sobre eles [16].

A modelação das escoras e tirantes fornece ao analista estrutural alguma liberdade para o desenvolvimento de múltiplas soluções para o mesmo problema. Desta forma, a escolha da solução pode ser feita para visar uma determinada solução otimizada, seja ela a solução mais segura, a mais barata, etc. A modelação, portanto, requer alguma experiência de projeto [20].

A norma brasileira ABNT NBR 6118:2014 [18] , para o dimensionamento de Regiões D através do método de Escoras e Tirantes, estabelece que os modelos devam ser exclusivamente isostáticos e autoequilibrados, sendo as reações de apoios obtidas a partir de um modelo linear ou não linear externo ao das bielas e tirantes. Assim como a teoria geral, os modelos devem ser compostos somente por escoras, tirantes e nós, sendo as condições de contomo exclusivamente as forças aplicadas nos nós, ativas ou reativas.

Outra limitação imposta à aplicação da teoria é a restrição ao ângulo de inclinação entre as escoras inclinadas e a armadura longitudinal do elemento estrutural.

O Quadro 7 apresenta os intervalos permitidos para estes ângulos, segundo recomendações da norma brasileira, americana e europeia. Sabe-se que quanto menor este ângulo, menor será a resistência à compressão da escora.

Quadro 7: Intervalos permitidos para o ângulo θ entre as escoras comprimidas e a armadura longitudinal no modelo de bielas e tirantes

NORMA	Ângulo θ (º)
ACI 318/2014	$25º \leq \theta \leq 65º$
ABNT NBR 6118:2014	$30º \leq \theta \leq 63º$
EUROCÓDIGO 2	$21º \leq \theta \leq 45º$

Os valores utilizados para a criação do nó de verificação descrito na Secção 3.3.1, atende aos critérios estabelecidos na norma ABNT NBR 6118:2014.

Para determinação da função objetivo utilizada pelo algoritmo de otimização genética na busca por uma solução ideal, consideraram-se os critérios estabelecidos pela ABNT NBR6118:2014

no que diz respeito à avaliação das regiões nodais, das escoras e dos tirantes que compõem o modelo STM.

4.1.1 Tipos de escoras

De acordo com a ABNT NBR 6118:2014, a partir da forma do campo de tensões idealizado por uma dada escora em uma estrutura plana, podemos classificá-la como prismática, em formato de leque ou em formato engarrafado (semelhante ao gargalo de uma garrafa), conforme ilustradas pela Figura 29, sendo que estes três tipos de escoras englobam todos os casos. As escoras do tipo prismático e em leque representam campos de tensão com nenhuma ou pequena curvatura, de modo que as tensões ortogonais a este campo são desprezíveis, em oposição, o campo engarrafado gera tensões significativas de tração devido às mudanças de direção do campo de tensões de compressão.

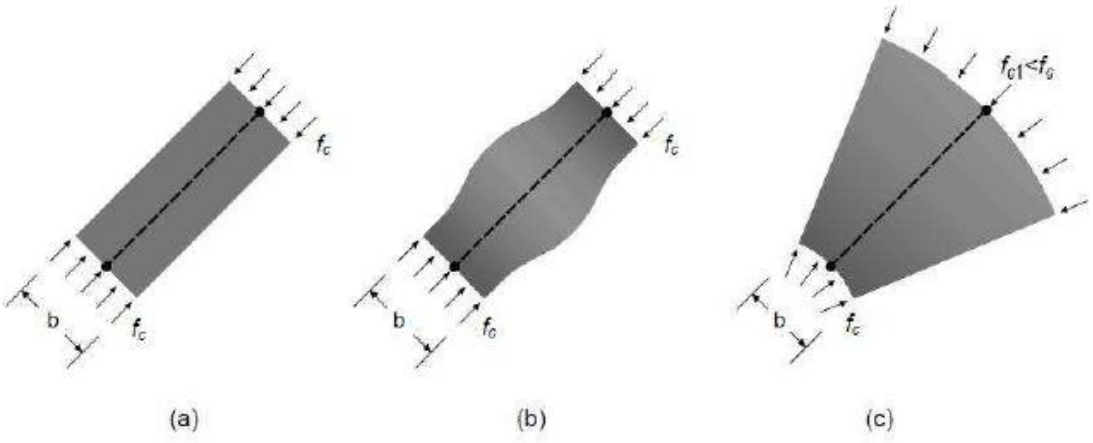


Figura 29: Tipos de escoras segundo a geometria: a) prismática; b) formato engarrafado; c) formato de leque [61].

Como regra geral, a tensão resistente das escoras ou bielas de compressão, não é a mesma da resistência característica do betão quando ensaiado através de provetes cilíndricos, como forma geral pode-se exprimir a resistência da escora por [62]:

$$f_{cdb} = \upsilon f_{cd} \tag{4}$$

Sendo:

f_{cdb} = tensão resistente da escora;

fcd = valor de cálculo da resistência do betão à compressão (ver Equação (1));

υ = fator de eficiência da escora, sendo menor ou igual a 1.

O valor de υ varia de acordo com o tipo de escora e outros fatores relativos ao confinamento e tensões transversais. Além disso, as referências normativas apresentam alguma variação nos valores para as mesmas condições. A resistência final da escora é dada por:

$$F_{cdb} = A_c f_{cdb} \tag{5}$$

Onde:

A_c = área contribuinte da escora, dada pelo produto entre profundidade fora do plano analisado, t, e a largura da escora dentro do plano, b.

Portanto:

$$F_{cdb} = tbf_{cdb} \tag{6}$$

4.1.2 Tirantes e critérios de resistência

Os tirantes representam o eixo de um conjunto de armaduras, sendo ela passiva ou pré-esforçada. Em casos especiais podem também caracterizar um campo de tensões de tração no betão ou de alivio de compressão. A tensão limite nos tirantes é dada simplesmente pela tensão de cedência do aço f_y, no caso de armadura passiva ou f_{yp}, no caso de armadura pré-esforçada [62]. Para esta dissertação, considerou-se somente o uso de armaduras passivas, portanto, as verificações para os tirantes são feitas com base apenas na tensão de cedência do aço, conforme a Equação (2).

4.1.3 Classificação dos nós

De acordo com Schlaich et al. [20], os nós podem ser classificados em dois tipos: nós continuos onde o campo de tensões principais apresenta pequena mudança de direção; e nós singulares,

onde ocorrem mudanças abruptas na direção das tensões principais. Os nós singulares devem ser mais profundamente estudados pois neles a ancoragem dos varões pode ser deficiente ou podem também apresentar problemas de ductilidade e deformação excessiva.

Na Figura 30 está representada uma abordagem mais direta aos tipos de nós, que diz respeito aos esforços incidentes sobre eles. Como regra geral os nós podem ser classificados em quatro tipos: CCC, CCT, CTT e TTT, onde C e T representam os esforços de compressão ou de tração.

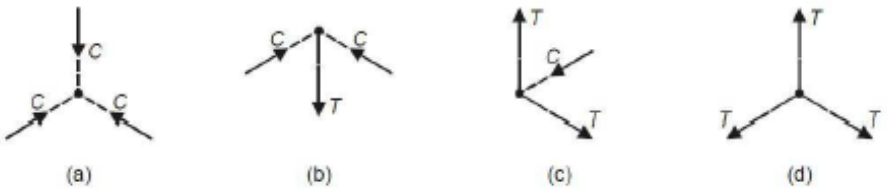


Figura 30: Tipos de nós de acordo com esforços incidentes [61].

Com relação às dimensões do nó, a definição mais direta é através da geometria das barras que nele se encontram por meio da projeção da interseção das diferentes barras conforme ilustrado pela Figura 31.

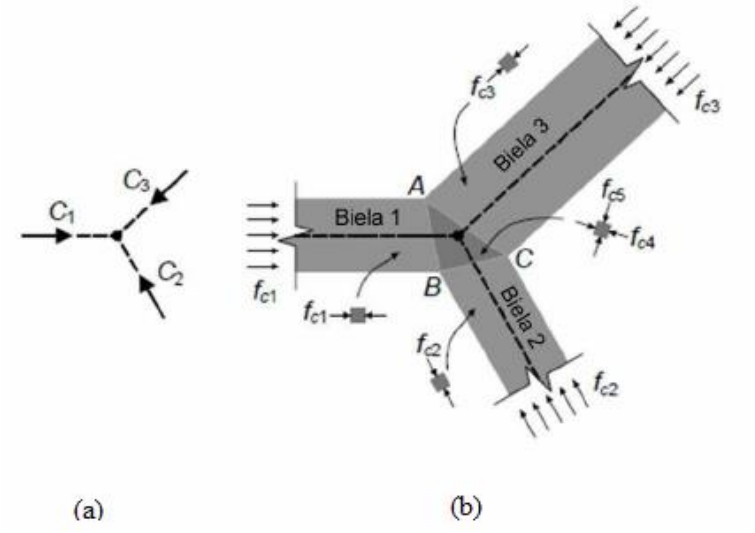


Figura 31: Forças atuantes no nó tipo CCC, sendo (a) geometria do nó e (b) geometria das bielas. [61].

4.1.4 Verificação das escoras comprimidas e regiões nodais

Para a verificação das tensões máximas nas escoras comprimidas e regiões nodais, consideraram-se os critérios estabelecidos pela ABNT NBR 6118:2014, item 22.3.2:

- Escoras prismáticas ou nós CCC (região circundada somente por escoras):

$$f_{cd1} = 0{,}85 \times \alpha_{V2} \times \frac{f_{ck}}{\gamma_c} \tag{7}$$

- Escoras atravessadas por mais de um tirante ou nós CTT ou TTT:

$$f_{cd2} = 0{,}60 \times \alpha_{V2} \times \frac{f_{ck}}{\gamma_c} \tag{8}$$

- Escoras atravessadas por um tirante ou nós CCT:

$$f_{cd3} = 0{,}72 \times \alpha_{V2} \times \frac{f_{ck}}{\gamma_c} \tag{9}$$

Onde:

f_{cd1} = *Tensão resistente à compressão máxima em escoras com compressão transversal ou sem tensões de tração transversal e nós onde se encontram somente escoras;*

f_{cd2} = *Tensão resistente à compressão máxima em escoras com tração e nós onde se encontram dois ou mais tirantes;*

f_{cd3} = *Tensão à compressão resistente máxima em nós com somente um tirante;*

f_{ck} = *Valor característico da resistência à compressão do betão;*

γ_c = *Coeficiente parcial relativo à resistência do betão.*

$$\alpha_{V2} = \left(1 - \frac{f_{ck}}{250}\right) \text{ com } f_{ck} \text{ em MPa} \tag{10}$$

4.2 Modelo de escoras e tirantes em vigas-parede

Com o intuito de verificar a aplicabilidade do processo de geração de modelos de escoras e tirantes (sistemas estruturais de treliça) desenvolvido no Capítulo 3 na obtenção de modelos STM de betão armado, optou-se, nesta fase, por buscar na literatura técnico-científica exemplos

cujo objetivo fosse a obtenção de um modelo de escoras e tirantes para dimensionamento de vigas-parede.

4.2.1 Exemplos

Diferente dos exemplos do Capítulo 3 – Secção 3.5, nos dois exemplos que se apresentam nesta secção, o modelo desenvolvido é testado para situações em que as barras da malha inicial são constituídas por materiais diferentes, sendo eles betão e aço. Isto se faz necessário pois é sabido que nos modelos de escoras e tirantes, aplicados a vigas-parede, os esforços de compressão a que se submetem as escoras são resistidos pelo betão. Nos casos dos tirantes, os esforços de tração são absorvidos pelas armaduras de aço, desprezando-se a resistência à tração do concreto, em se tratando das verificações de Estados Limites Últimos segundo a ABNT NBR6118:2014.

Sendo assim, buscou-se na literatura técnico-científica exemplos cujo objetivo principal fosse a obtenção de modelos de escoras e tirantes para o estudo de vigas-parede. Porém, o exemplo escolhido como base para comparação, quando analisado e comparado com os critérios estabelecidos segundo a ABNT NBR 6118:2014, não se enquadra na categoria de viga-parede. Já do ponto de vista do Eurocódigo 2, o exemplo estaria na transição entre viga e viga-parede. Entretanto, segundo o ACI 318-14, o modelo em questão pode sim ser dimensionado como sendo um elemento de viga-parede. Apesar das divergências em relação as normativas adotadas, optou-se por utilizar o exemplo a seguir, devido à riqueza de informações fornecidas. Dito isto, foram realizadas comparações entre resultados com intuito de validar o código desenvolvido neste trabalho.

O Exemplo 3, desenvolvido por Mello & Souza [63] e representado na Figura 33, como já mencionado, tem como objetivo a definição de um modelo de escoras e tirantes que possibilite o dimensionamento de uma viga-parede e, portanto, não há uma estrutura inicial composta por barras e nós.

Para a reprodução deste exemplo foi criada uma malha composta por 41 barras e 18 nós, ver Figura 34. O material utilizado para criação das barras foi o mesmo utilizado pelo autor, tendo o betão um valor característico da resistência à compressão de 30 MPa e o aço um valor de tensão de cedência característico à tração de 500 MPa. Já os valores de cálculo foram obtidos de acordo com a ABNT NBR 6118:2014: f_{cd} = 21,43 MPa e f_{syd} = 434,78 MPa. Duas cargas

nodais com valor de 693 kN estão aplicadas nos nós de número 6 e 9. A viga-parede possui 5,4 metros de comprimento, 1,8 metros de altura e 0,4 metros de espessura.

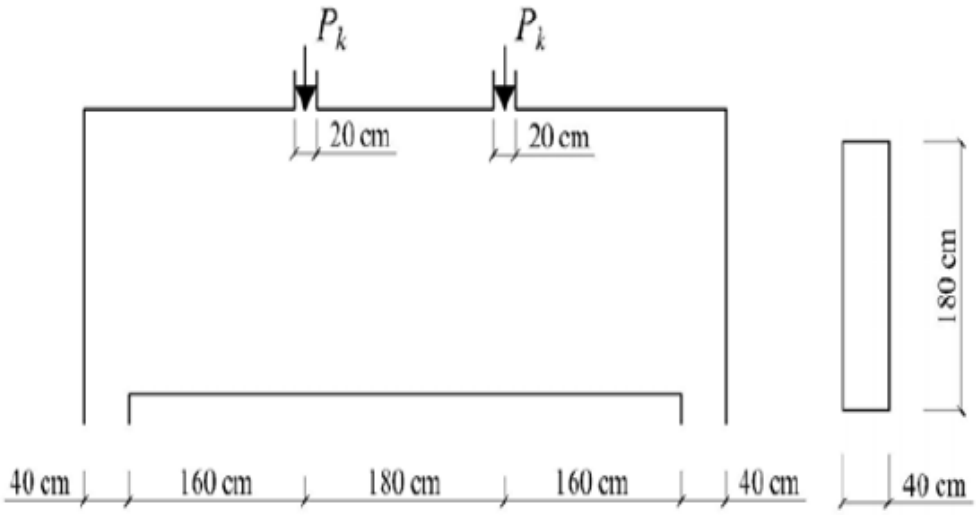


Figura 32: Exemplo 3 - viga-parede [63].

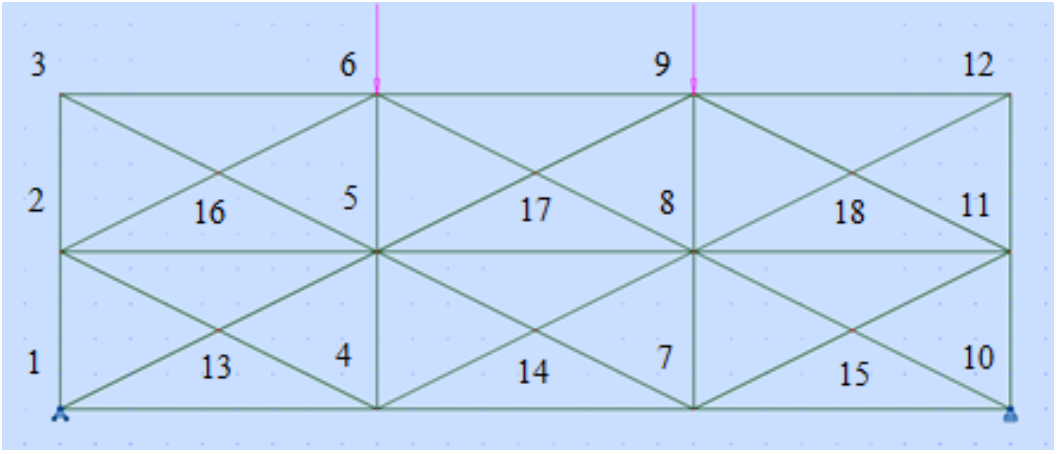


Figura 33: Malha inicial usada no Exemplo 3 - viga-parede, com a identificação dos nós.

4.2.2 Análise de resultados

Assim como ocorreu nos exemplos do Capítulo 3, onde apesar das diferenças utilizadas nos critérios de avaliação da estrutura para obtenção do modelo STM, o exemplo aqui apresentado, teve resultados satisfatórios.

Para fins de comparação, na Figura 34 está representado o modelo de escoras e tirantes obtido por Mello & Souza [63], com auxílio de um *software* CAST desenvolvido por Tjhin and Kuchma [64], como solução para a viga-parede.

Já na Figura 35, encontra-se a solução obtida nesta dissertação para o Exemplo 3. Diferente dos demais casos, apesar da semelhança, é visível que os modelos aqui gerados apresentam maior número de barras. Uma das possíveis causas para tal, está relacionada com os critérios de exclusão estabelecidos no Capítulo 3 (Secção 3.4.2), onde se procura manter os nós da estrutura com pelo menos 3 barras conectadas, visando uma diminuição no número de possíveis instabilidades. Além disso, devido ao arranjo adotado para a malha inicial, não foi possível conectar as cargas nodais diretamente aos apoios por meio de uma única barra.

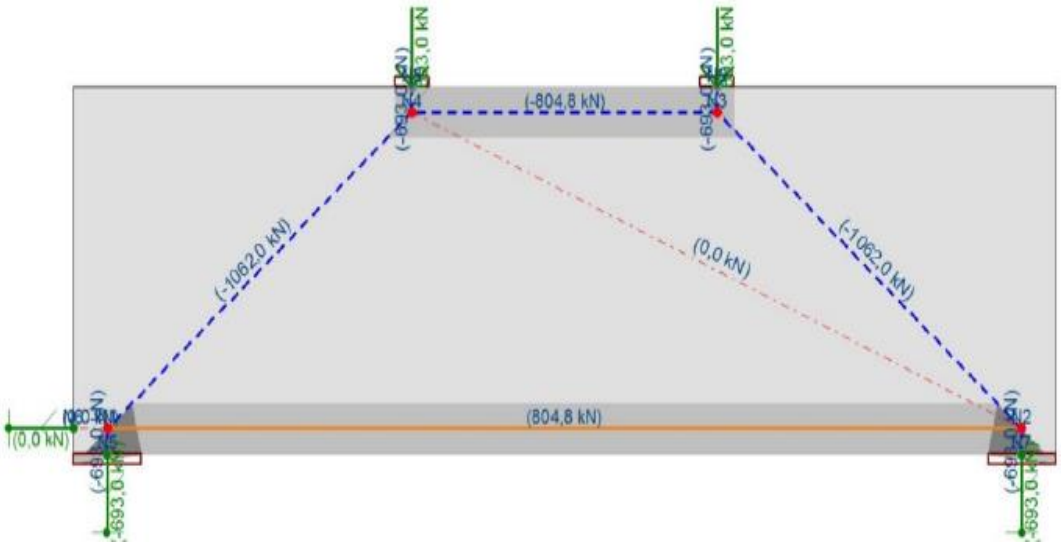


Figura 34: Modelo de escoras e tirantes resultante segundo [63].

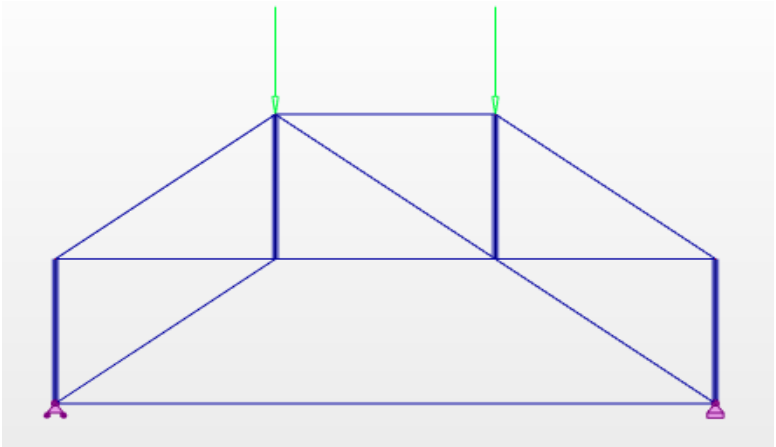


Figura 35: Exemplo 3 - Modelo de escoras e tirantes resultante.

Com o auxílio do programa de cálculo estrutural Ftool [65], foram obtidos para os dois exemplos os esforços de tração e compressão nas barras. Apesar da diferença de topologia, percebe-se através da Figura 36 que em ambos os casos permaneceram na estrutura barras com a finalidade exclusiva de evitar o surgimento de instabilidades, sejam elas globais e/ou locais. Em relação aos esforços, mais precisamente os esforços de compressão que, por convenção do programa utilizado possuem valor negativo, percebe-se que o valor máximo de compressão obtido por Mello & Souza [63] é de 1062,0 kN, já no exemplo aqui desenvolvido, este valor é ligeiramente inferior, de aproximadamente 876,2 kN, mas o valor de compressão obtido na zona superior da viga, entre as cargas aplicadas, é o mesmo. Entretanto, nos esforços de tração – que por sua vez possuem sinal positivo – o valor máximo encontrado para o tirante na base da viga é 804,8 kN e é igual para ambos os casos. Porém, vale ressaltar, novamente, que na aplicação desenvolvida não foi possível a definição de uma barra que ligue o apoio ao nó onde estão aplicadas as cargas. Então, o modelo de escoras e tirantes possui em cada uma das zonas laterais da viga dois montantes e duas escoras, cujos esforços (346,5 kN e 876,2 kN) são equilibrados por um tirante com o mesmo esforço de tração obtido na base da viga-parede.

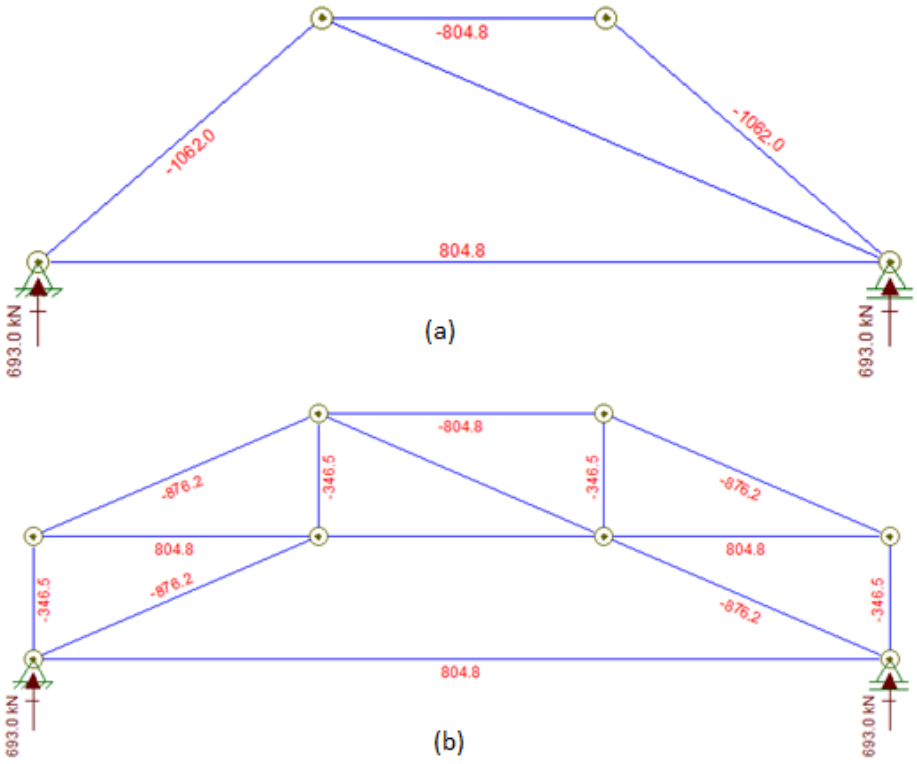


Figura 36: (a) Esforços normais obtidos com o auxílo da ferramenta Ftool para o exemplo desenvolvido por Mello & Souza [63]; (b) Esforços normais obtidos com o auxílo da ferramenta Ftool para o exemplo desenvolvido nesta dissertação.

Sendo assim, apesar da relativa semelhança na topologia do modelo de escoras e tirantes, é evidente que os resultados obtidos foram diferentes. Entretanto, devido a limitações de tempo e de objetivos do trabalho, optou-se por avançar para a implementação do módulo de otimização com recurso a algoritmo genético.

Portanto, assume-se, nesta fase, que o código Python desenvolvido, apesar de ter critérios de avaliação relativamente simples, será utilizado para a obtenção de modelos de escoras e tirantes.

4.3 Implementação da otimização com algoritmo genético

Obtido o modelo de escoras e tirantes, inicia-se a segunda etapa do *workflow*, que consiste no processo de pré-dimensionamento e otimização das secções das escoras e dos tirantes. Para isso, faz-se uso da ferramenta Optimo, disponível na biblioteca do Dynamo.

Para utilização do Optimo é necessária a criação de nós personalizados. Os nós personalizados consistem numa coleção de nós predefinidos existentes no Dynamo e Python *scripts*, que são agrupados e guardados localmente no formato ".dyf" (descrito no Capítulo 3). Os nós personalizados criados para esta otimização serão descritos no decorrer deste capítulo. O *workflow* criado no Dynamo para realizar a otimização com o Optimo pode ser visto na Figura 37.

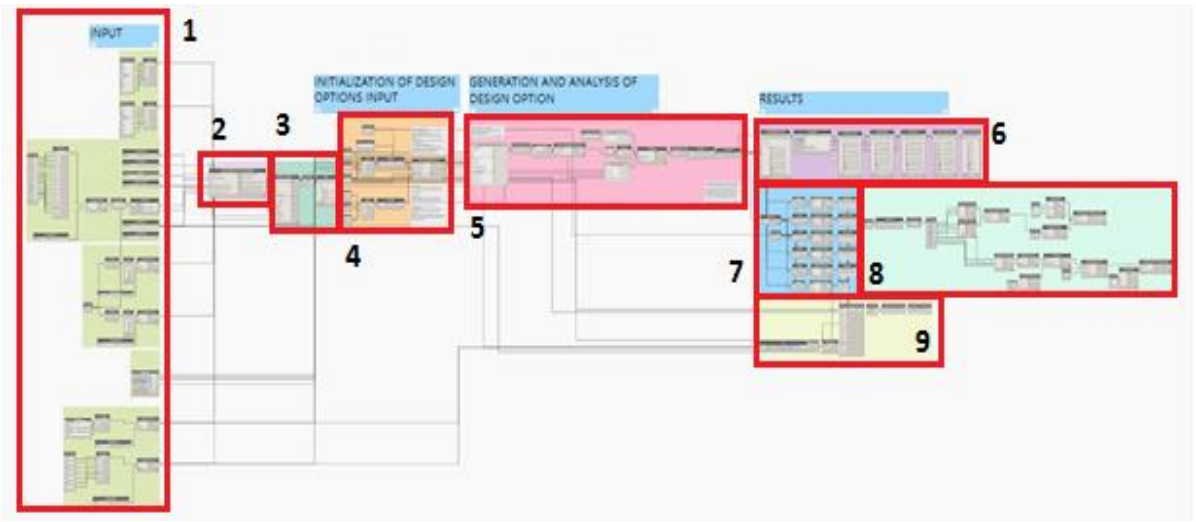


Figura 37: Sequência de funcionamento do pacote Optimo.

Através do Quadro 8, que contém a explicação detalhada de cada um dos itens presentes na Figura 37, é possível ter um melhor entendimento do processo de otimização.

Inicialmente, pretendia-se tratar cada elemento do modelo de escoras e tirantes como tendo variáveis de projeto separadas, porém não foi possível devido a problemas não identificados na relação do Optimo com o Robot. Portanto, as escoras e os tirantes foram otimizados para atender ao elemento mais solicitado.

Quadro 8: Explicação do *workflow* - Optimo.

1. **Entradas das variáveis descritas na Secção 3.3.1**
2. **Nó personalizado descrito na Secção 3.3.1**

Verificar se os dados inseridos pelo utilizador atendem aos critérios de caracterização de uma viga-parede e se atendem aos critérios de inclinação das escoras para a criação do modelo de escoras e tirantes

3. **Nó personalizado descrito na Secção 3.3 e 3.4**

Obter um modelo de escoras e tirantes

4. **Nó NSGA_II.InitialSolutionList**

Determinar o tamanho da população inicial (pais), o número de objetivos da otimização e os limites superiores e os inferiores para a população inicial

5. ***Fitness node* descrito na Secção 4.3.2 *NSGA_II.FitnessFunctionResult, NSGA_II Function, LoopWhile***

5.1 Gerar, através do nó LoopWhile, os filhos que serão utilizados como novos pais, em função dos pais e dos seus resultados

5.2 Definir o número de filhos a gerar (utilizador)

6. **Visualizar todos os resultados da otimização**
7. **Visualizar resultado do processamento do Optimo para cada variável**
8. **Nó personalizado**

Selecionar um elemento pertencente a uma família estrutural que esteja disponível no projeto Revit e posicioná-lo ao longo das linhas que representam as escoras e os tirantes

9. **Nó personalizado**

Verificar quais os nós que atendem aos critérios de verificação

4.3.1 Core node

No *Core Node* (Figura 38) encontra-se um código Python desenvolvido com o intuito de avaliar a estrutura existente e, com base em funções de penalização, fornecer uma nota (*Score*) para a estrutura gerada durante o processo de otimização genética.

A Figura 39 mostra o *workflow* contido no interior do *Core Node* da Figura 38, usado para obter e o resultado (Score) da estrutura para cada uma das iterações, onde as etapas indicadas estão resumidas no Quadro 9.

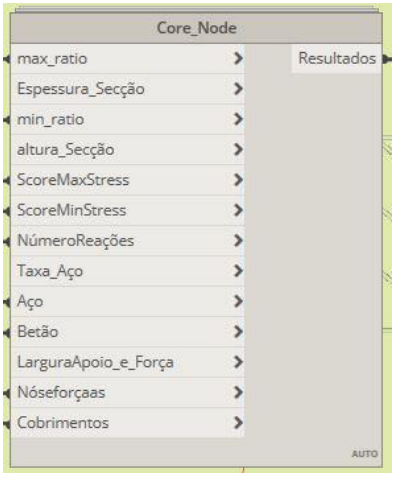


Figura 38: Core Node.

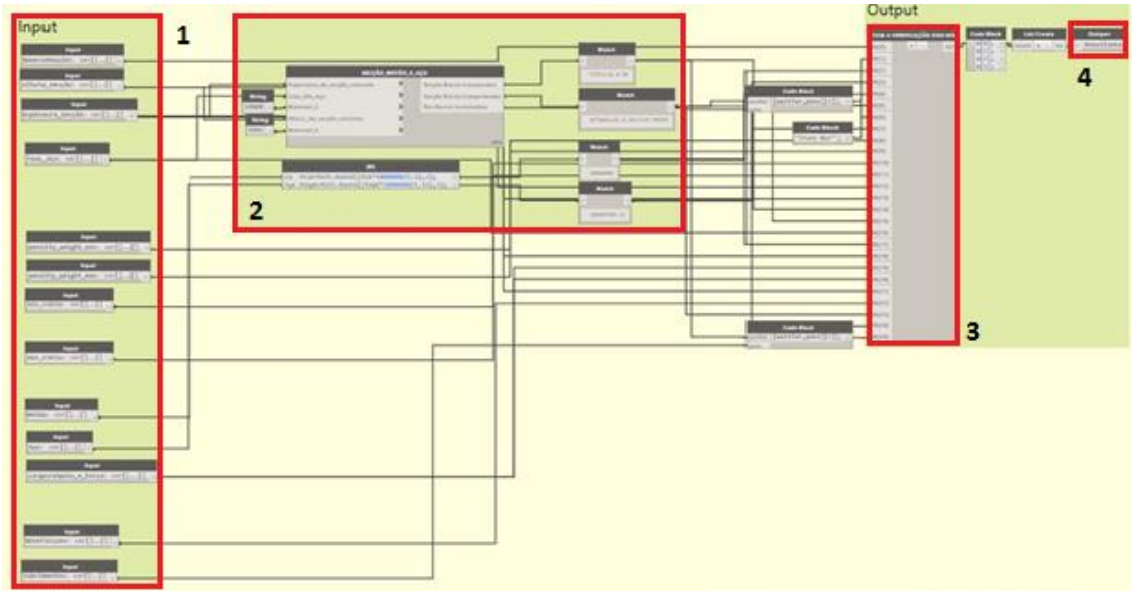


Figura 39: Interior do Core Node.

Quadro 9: Explicação do nó Core Node.

1. Input do nó
2. Criação das Secções das barras (Secção 3.3.1)
3. Código Python para obtenção do "Score" da estrutura.
4. Output contendo o "Score"

O processo de avaliação contido no código Pyhton (item 3 do Quadro 9), inicia-se com o cálculo do modelo de escoras e tirantes resultante descrito na Secção 4.1 e baseia-se na utilização de funções de penalização. Estas funções têm como simples objetivo penalizar as soluções não viáveis. Desta forma, a função objetivo é substituída pela função já penalizada. A função de penalização para o problema de minimização pode ser descrita da seguinte forma:

$$F_p = F + \sum C_i \delta_i \tag{11}$$

Onde:

F_p = Função objetivo penalizada;

F = Função objetivo;

C_i = Coeficiente de penalização imposto para a violação da i-ésima restrição;

δ_i= 1, se a restrição for violada;

δ_i= 0, se a restrição for satisfeita.

Após o cálculo da estrutura, as barras são novamente divididas com base nos esforços aos quais estão submetidas. As barras comprimidas recebem uma secção retangular com dimensões arbitrárias criadas pelo nó "NSGA_II.InitialSolutionList". Já as secções tracionadas, recebem secções circulares, cuja área e diâmetro são definidos em função da área de betão das secções comprimidas e em função do valor da taxa de armadura, também atribuida por "NSGA_II.InitialSolutionList". Além das secções, as barras passam a ser constituídas por materiais diferentes, sendo as comprimidas de betão e as tracionadas de aço.

Depois de atribuídas as novas secções e materiais, a estrutura é calculada mais uma vez para obtenção das tensões atuantes em cada uma das barras. Novamente comparam-se as forças obtidas com os valores de referência descritos no Capítulo 3 Secção 3.3.1, porém, desta vez, obtem-se a razão entre a tensão atuante e o valor de referência. Esta razão é então comparada

com os valores limites pré-estabelecidos pelo utilizador, chamados de "max_ratio" e "min_ratio". Nesta dissertação atribuiu-se o valor 0,95 como valor máximo e o valor de 0,3 como mínimo. Se a razão obtida estiver fora destes limites é então atribuído um peso como penalidade, chamado de "ScoreMaxStress" e "ScoreMinStress", cujos valores também podem ser definidos pelo utilizador. Neste trabalho, para as barras com razão superior ao limite máximo o peso atribuído é de 1000, já para as barras com razão inferior ao limite mínimo o peso atribuído é de 100. A penalidade para os membros subdimensionados é significativamente maior do que para membros sobredimensionados. Desta forma, no final da análise de todas as barras é obtido o "StressScore" da estrutura, que é a soma do peso próprio da estrutura mais as penalidades atribuídas.

O peso próprio da estrutura deve ser considerado pois, com o passar das iterações, o algoritmo pode vir a aumentar demasiadamente o tamanho das secções, uma vez que o intervalo estabelecido para a relação tensão solicitante/tensão resistente vai de 0,3 até 0,95. Portanto, mesmo estando dentro do intervalo desejado, quanto mais próximo do valor 0,3 menos otimizada está a secção. Desta forma, estaríamos a desperdiçar material e a contrariar os princípios da otimização. Sendo assim, o facto de se adicionar o peso próprio na composição do "Score" auxilía o algoritmo na busca pela melhor secção.

Os outros dois métodos de avaliação estão diretamente ligados ao processo de verificação dos critérios de dimensionamento das vigas-parede, mais precisamente em relação à área mínima de aço nos tirantes e na verificação das tensões nos nós. Caso algum tirante satisfaça a condição estabelecida na Equação (12), é-lhe atribuido um peso de 2 (duas) vezes o "ScoreMaxStress". Este valor é somado no "Score" da estrutura.

$$0.04 \times \mathrm{b} \times \mathrm{d} \leq A_S \tag{12}$$

Onde:

b = Largura da face de compressão da escora (corresponde à espessura da viga-parede);

d = Distância da fibra de compressão extrema ao centróide dasarmaduras longitudinais de tração;

A_s = Área da secção de uma armadura para betão armado.

Os nós são classificados de acordo com o descrito na Secção 4.1.3 e posteriormente verificados. Os nós que não atendem ao critério de verificação de tensões é atribuido um peso correspondente ao "ScoreMaxStress". Portanto, quanto mais nós em incumprimento, maior será o "Score" de penalidade da estrutura. Sendo assim, o processo de otimização trabalha com uma função objetivo composta por três valores, o "StressScore". A solução que apresente o "StressScore" com valor mais próximo do peso próprio da estrutura é classificada como a solução ideal.

Quanto às entradas (*inputs*) do nó, podem ser divididas em duas categorias: aquelas cujo utilizador controla, que devem ser preenchidas, e aquelas cujo controlo é feito pelo Optimo e que devem ser deixadas em branco. Os dados que o Optimo controla são na verdade as populações geradas no nó "NSGA_II.InitialPopulationList" e posteriormente em "LoopWhile", que são os valores variáveis utilizados na busca da solução ótima da estrutura. Os nós que o utilizador tem liberdade para modificar são variáveis apenas antes da execução do *workflow*, depois de iniciada a execução, os dados inseridos são imutáveis. Como saída (*output*) do Core Node temos as funções objetivo "StressScore" e "NóScore", descritas anteriormente e cujo número é definido no nó "NSGA_II.InitialSolutionList".

É possível adicionar e remover as entradas e saídas dos nós personalizados, o que permite ao utilizador interagir com o nó e alterar o número de funções objetivo, se assim for necessário.

4.3.2 Fitness node

O Fitness Node (Figura 40) é o nó onde está inserido o Core Node. Nele a informação da população atual é combinada com os *scores* da estrutura retornado pelo Core Node com o intuito de criar uma nova função que inclua estes valores, de forma combinada, de maneira a fornecer a adequação da população atual. Essa nova função é o dado de saída (*output*) do Fitness Node.

As entradas do Fitness Node são semelhantes àquelas contidas no Core Node, com exceção dos itens que se pretende otimizar.

A Figura 41 mostra o *workflow* contido no interior do Fitness Node da Figura 41, usado para extrair o resultado da combinação entre a população e o resultado (*Score*). A explicação do gráfico pode ser encontrada no Quadro 10.

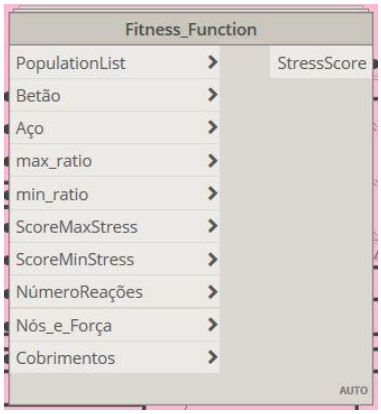


Figura 40: Fitness Node.

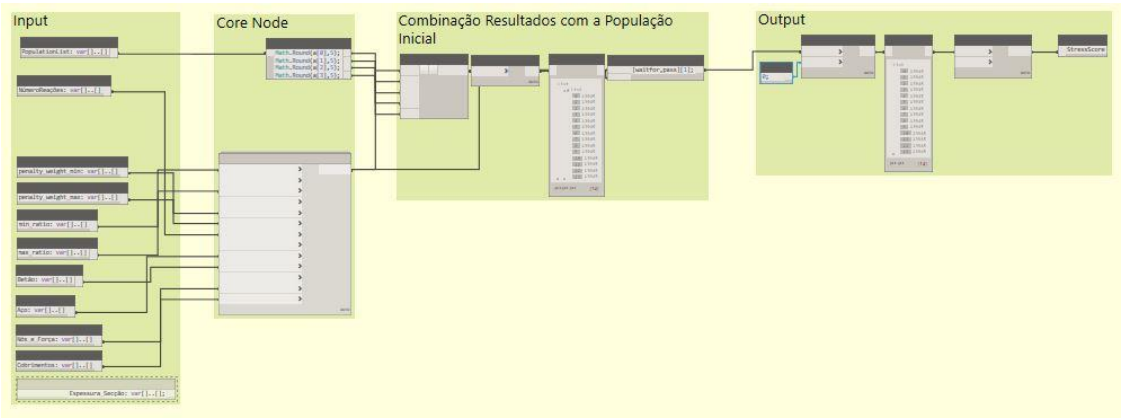


Figura 41: Interior do Fitness Node.

Quadro 10: Etapas do Fitness Node.

1. Input do nó
2. *Core Node* (Secção 4.3.1)
3. *Code Block* utilizado para arredondar os valores da população
4. Nó para combinar a população com os *scores* da estrutura
5. *Output* contendo a adequação da população estudada

4.4 Pré-dimensionamento da viga-parede

Assim como nos exemplos da Secção 4.1, optou-se por utilizar como base um exemplo disponível na literatura técnico-científica para validação do modelo desenvolvido, sendo assim, possível de comparação. Entretanto, desta vez, pretende-se validar todo o *workflow*

desenvolvido, partindo de uma viga-parede, obtendo o modelo STM e posteriormente otimizando as escoras e os tirantes que o compõem.

4.4.1 Caso de estudo

Para fins de comparação, adota-se o modelo apresentado por Singh et al. [66], desenvolvido pelos autores de forma analítica. As cargas, vãos e dimensões da viga-parede selecionada para análise são apresentados na Figura 43. Além disso, foram adotados os valores característicos de 24 MPa e 415 MPa para valores característicos da tensão de rotura do betão à compressão e da tensão de cedência à tracção do aço, respectivamente. Já para valores de cálculo, foi adotado f_{cd} = 16 MPa e f_{syd} = 360,87 MPa, respectivamente.

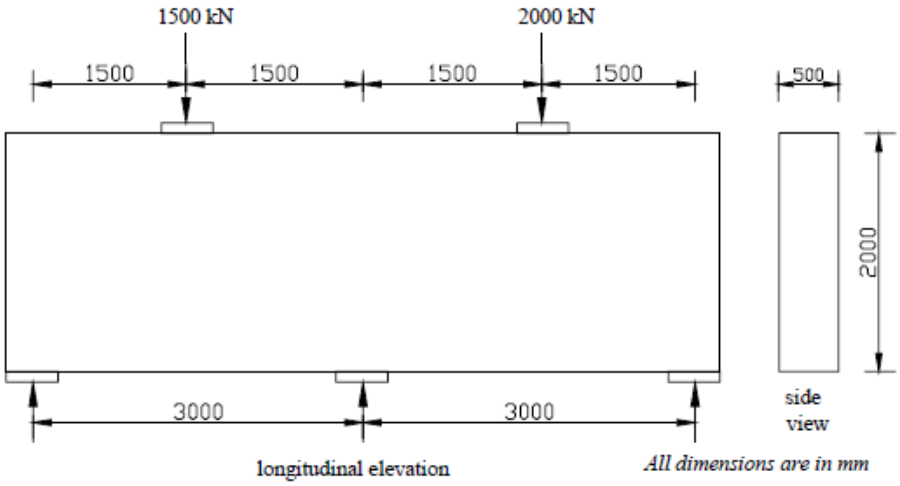


Figura 42: Caso de Estudo - Viga-parede utilizada com base para obtenção do modelo de escoras e tirantes [66].

Para explorar a capacidade total de carga da viga, é imperativo que os nós inferiores e superiores fiquem mais próximos possível das faces da viga [66]. Será assumido, conforme proposto por Singh et al. [66], que o centro de gravidade dos varões da armadura esteja situado a uma distância de 75 mm das faces superior e inferior da viga, respectivamente. A largura das escoras, a espessura da viga-parede e a área de armadura dos tirantes serão determinados por meio da utilização da optimização genética em função das forças aplicadas, da tensão admissível no betão e dos nós de ancoragem.

Há um detalhe importante que se deve ressaltar, se observamos tão somente o critério adotado pela ABNT NBR 6118:2014 e descrito no Capítulo 2 Secção (2.2.1), a viga-parede calculada por Singh et al. (2006) deveria comportar-se como uma viga esbelta, uma vez que para essa viga a relação ℓ/h é 3. No entanto, se considerarmos os critérios estabelecidos pelo Eurocódigo 2, a viga se comporta como viga-parede, uma vez que, a relação ℓ/h é 1,5 e portanto, inferior a 3. Ainda, de acordo com o ACI 318-14, também é possível concluir que a viga se comporta como uma região de descontinuidade generalizada, isto é, viga-parede. A esse respeito, Souza [67] declara:

"No entanto, a NBR 6118 (2003) e a maioria dos códigos acaba não capturando este efeito pois levam em consideração a relação l/h para classificar uma viga em comum ou parede. Dessa maneira, uma viga que deveria ser dimensionada como parede acaba sendo dimensionada como se fosse comum e um dimensionamento inseguro pode ser gerado".

Levando-se em consideração a segurança, as condições de contorno estabelecidas por Singh et al. (2006) são reproduzidas no *workflow* desenvolvido no Dynamo e a viga do exemplo será pré-dimensionada com base nos critérios da norma brasileira para vigas do tipo parede. Na Figura 43 está representada a malha inicial utilizada como base para obtenção do modelo de escoras e tirantes.

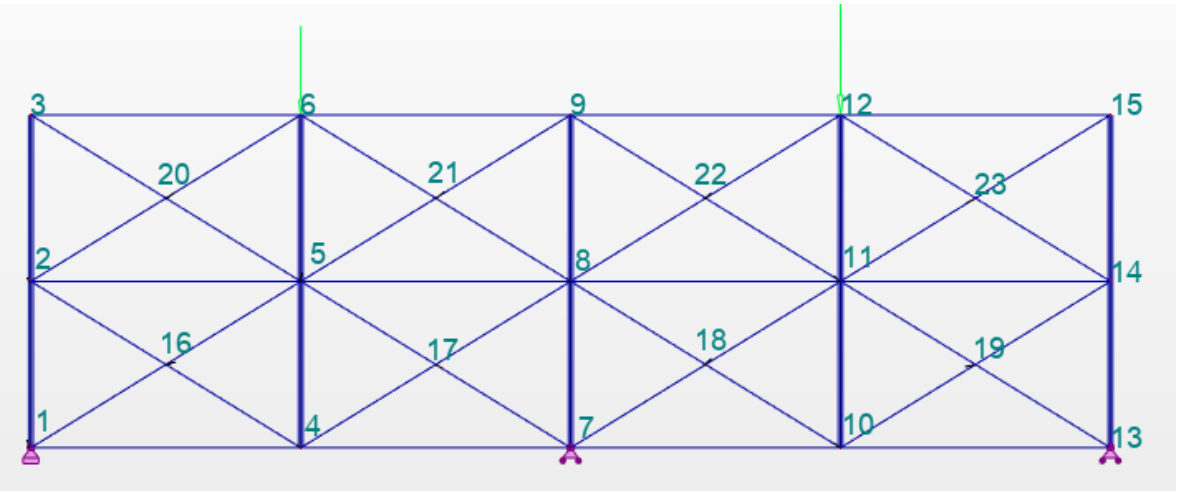


Figura 43: Malha inicial usada no Exemplo 4 - viga-parede, com a identificação dos nós.

4.4.2 Análise dos Resultados

Para obtenção dos resultados, partiu-se de uma população inicial (pais) composta por 15 elementos para cada variável que se deseja optimizar. Após a obtenção do "Score" para as 15 estruturas, a população inicial e os resultados foram combinados e enviados até o nó LoopWhile. Neste nó LoopWhile, com base nos resultados previamente obtidos, criou-se uma nova população (filhos) de 15 elementos para cada variável. Esses valores foram adotados devido às limitações descritas na Secção 3.2.5.

O Quadro 11 contém, de forma resumida, os resultados ótimos obtidos para cada uma das variáveis após finalizadas todas as iterações. Através dele é possível realizar uma comparação de forma clara entre os dois modelos.

Quadro 11: Comparação de resultados dos dois modelos – Caso de Estudo.

Parâmetro	Resultados de Singh et al. [66]	Resultados obtidos com o modelo desenvolvido
Espessura da viga-parede	0,50 m	0,47 m
Largura da escora com a maior solicitação de projeto	0,26 m	0,39 m
Área da secção transversal da biela com o maior esforço axial instalado	0,13 m^2	0,19 m^2
Largura da região onde estão aplicadas as forças e os apoios	0,60 m	0,51 m
Taxa de armadura	1,73 %	1,64 %
Maior área de armadura	22,5 cm^2	30,0 cm^2
Sugestão de armadura para o tirante com a maior solicitação de projeto*	12 x ø16 = 24,1 cm^2	15 x ø16 = 30,2 cm^2
Peso próprio da estrutura	-	82,6 kN
"Score"	-	9219,75

*A área de amadura sugerida para o tirante mais solicitado, foi obtida com o auxílio do Quadro B.1 disponível em [68].

Em relação ao dimensionamento realizado por Singh et al. (2006) e representada na Figura 44, percebe-se que os resultados obtidos nesta dissertação com auxílo dos algoritmos genéticos e representados na Figura 45 são muito similares.

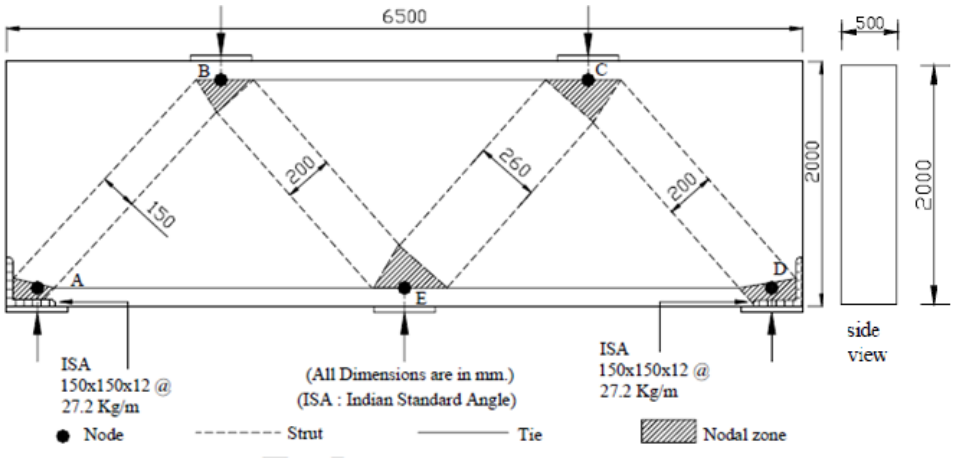


Figura 44: Estrutura dimensionada de acordo com Singh et al. [66].

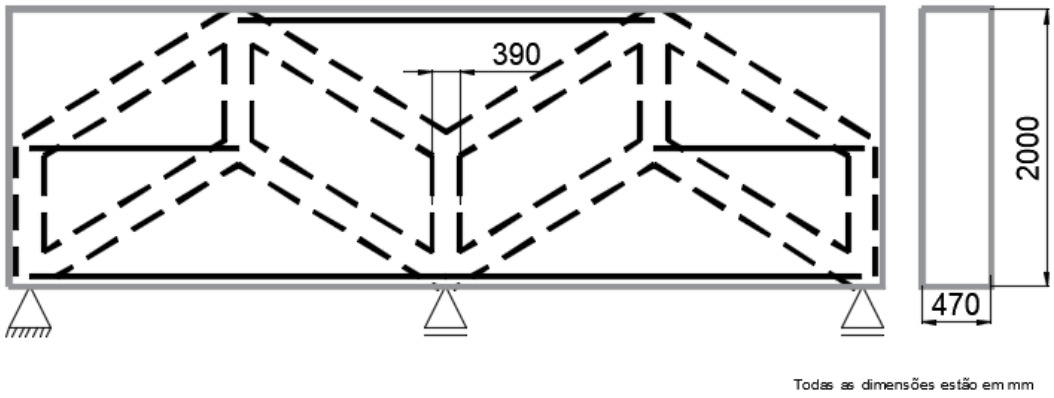


Figura 45: Estrutura obtida por meio da otimização com algoritmos genéticos.

Entretanto, apesar da semelhança de topologia, onde em ambos os casos as barras se concentram nos chamados “caminhos de carga” ou “caminhos de força”, que transferem os esforços dos pontos onde são aplicadas as cargas até aos apoios da estrutura, devem ser mencionadas a diferença no número de barras e a presença de um tirante nas regiões laterais da viga-parede a meia altura da secção. Tanto o número de barras, quanto a presença dos tirantes nessas regiões devem-se, aos critérios estabelecidos no Capítulo 3.4.2.

Uma vez que a topologia não é a mesma, é de se esperar que os esforços incidentes sobre os elementos das estruturas também sejam diferentes, conforme se pode observar na Figura 46.

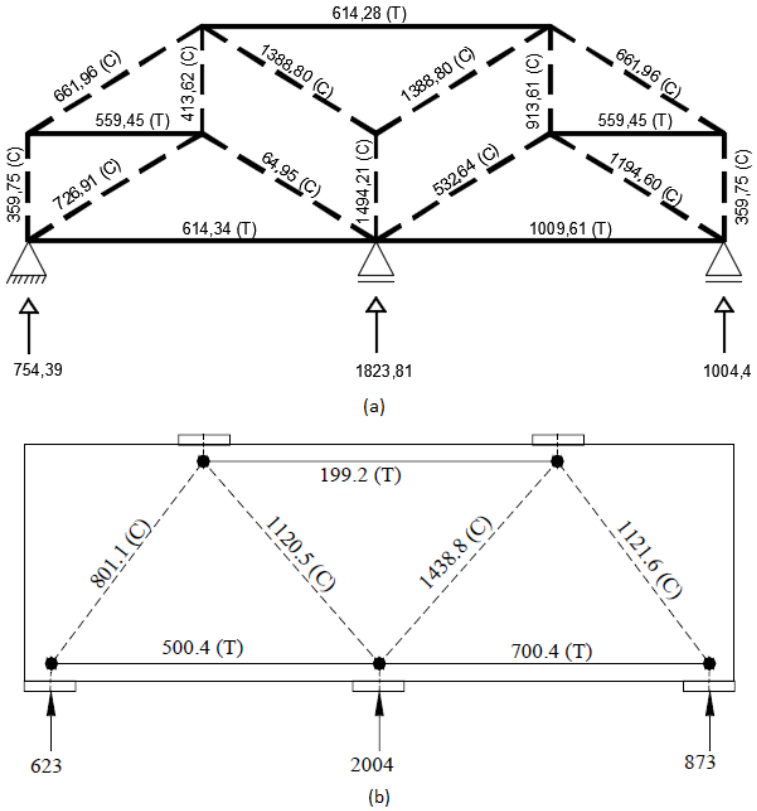


Figura 46: (a) Esforços obtida pelo modelo desenvolvido; (b) Esforços obtidos por [66].

De facto, ao se analizar a Figura 46, percebe-se que há uma diferença considerável. Entretanto, como o modelo desenvolvido neste trabalho considera para o pré-dimensionamento apenas a escora com maior esforço normal, sendo assim, os valores máximos de compressão em ambos os modelos são muito próximos, 1438,8 kN segundo Singh et al. [66] e 1494,21 kN obtidos pelo método desenvolvido. Esses valores justificam, em termos de dimensão máxima de secções, os resultados encontrados. Uma vez que em termos de área da maior secção, Singh et al. [66] obteve 0,13 m², já através do metódo desenvolvido neste trabalho obteve-se 0,19 m².

Nos esforços atuantes nas barras tracionadas há também uma grande diferença, principalmente no tirante superior posicionado entre as cargas nodais, onde o esforço de tração é aproximadamente 3 vezes maior. Nos tirantes inferiores há também um aumento significativo, cerca de 44% no tirante inferior direito e 23% no tirante inferior esquerdo. Para além das diferenças nos esforços de tração e compressão, as reações de apoio também foram afetadas. No resultado obtido com auxílio do Optimo, as reações dos apoios localizados nas extremidades sofreram um aumento significativo. Em contrapartida, para que haja o equilibrío das cargas, a reação localizada no apoio central da estrutura diminuiu. Ainda, em relação às reações de apoio,

para este trabalho, conforme visível na Figura 46, considerou-se um apoio rotulado na extremidade esquerda da viga-parede e outros dois apoios com liberdade de deslocamento em relação ao eixo X. Sendo assim, as reações de apoio se dão exclusivamente na vertical.

As diferenças encontradas, tanto nos esforços quanto nas reações, estão relacionadas à distribuição dos esforços entre um número maior de barras, principalmente há existência de escoras verticais (montantes) e à ausência destes elementos no modelo obtido por Singh et al. [66].

É importante citar que, o peso próprio da estrutura (82,6 kN) – que pode ser verificado através das reações de apoio presentes na Figura 47 – não apresenta grande influência nos resultados obtidos em comparação com o incremento no número de barras e consequente mudança de topologia. Entretanto, o peso próprio é imprescindível para o funcionamento da aplicação desenvolvida no trabalho, uma vez que, conforme descrito na Secção 4.3.1, algumas das funções de penalidade utilizadas pelo algoritmo de otimização baseiam-se no peso próprio da estrutura.

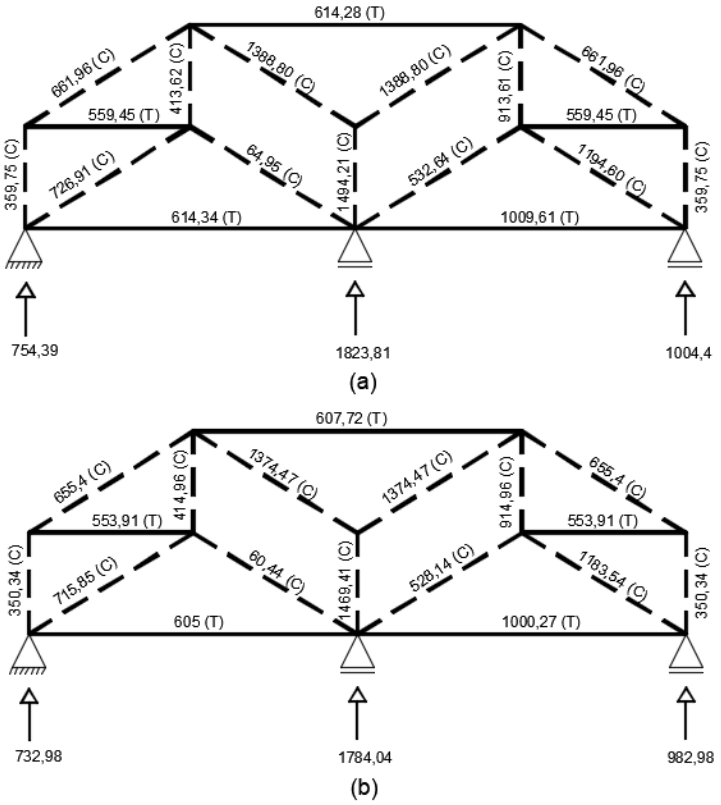


Figura 47: (a) Esforços obtidos com o auxílio do Ftool [65] a partir das forças externas e do peso próprio; (b) Esforços solicitantes obtidos com o auxílio do Ftool [65] a partir, única e exclusivamente, das forças externas.

Ainda relacionado aos esforços de tração, como citado anteriormente, devido à topologia do modelo de escoras e tirantes obtido, surgiram tirantes na zona a meia altura da viga-parede.

Para as armaduras em especial, percebe-se que a área de aço obtida através dos algoritmos genéticos é ligeiramente superior àquela obtida por Singh et al. [66], mas em correspondência com os resultados obtidos.

É importante ressaltar que, conforme descrito na Secção 4.1.4, todos os nós do modelo de escoras e tirantes foram verificados. Na Figura 48 está representado o nó do Dynamo criado com o intuito de verificar e comprovar que, após todo o processo de iteração, todos os nós do modelo de escoras e tirantes atendem aos critérios de verificação estabelecidos.

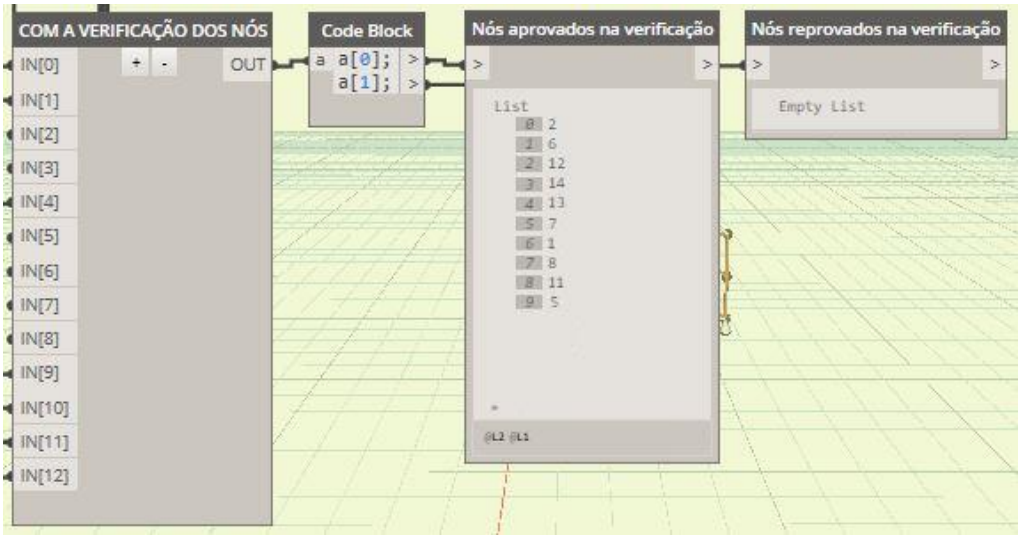


Figura 48: Verificação dos nós do modelo de escoras e tirantes.

No que diz respeito a utilização de algoritmo genético, mais precisamente o Optimo, a Figura 49 contém o gráfico com os resultados para cada uma das variáveis de projeto após o algoritmo concluir todas as iterações.

Na Figura 50 ilustram-se os resultados ótimos de cada uma das variáveis consideradas após concluídas as 30 iterações previstas no algoritmo. Estes resultados foram apresentados de forma mais detalhada no Quadro 11.

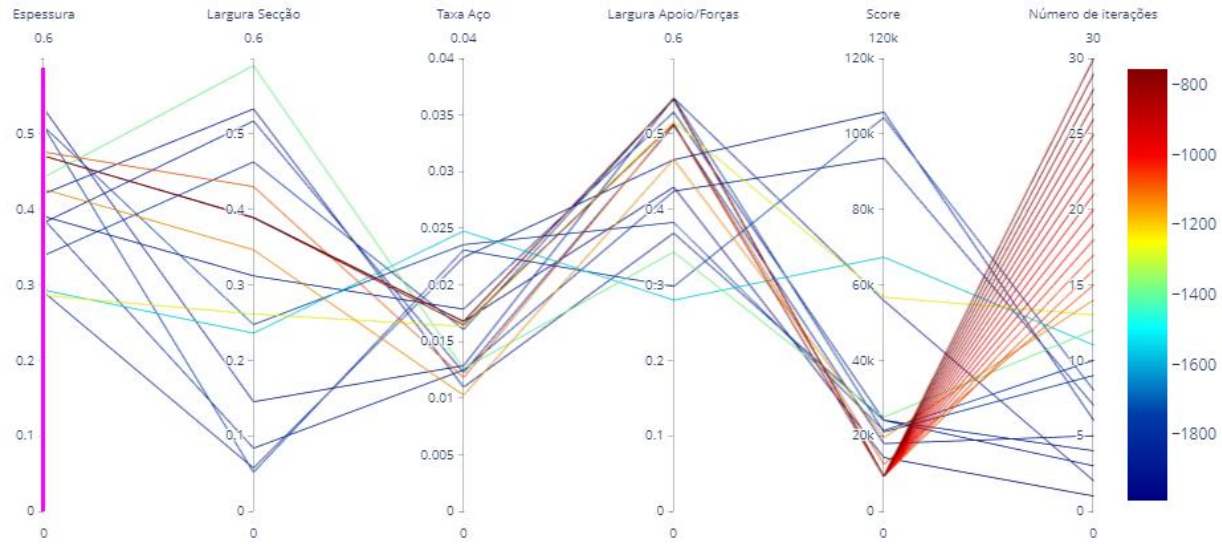


Figura 49: Evolução do valor das variáveis a otimizar.

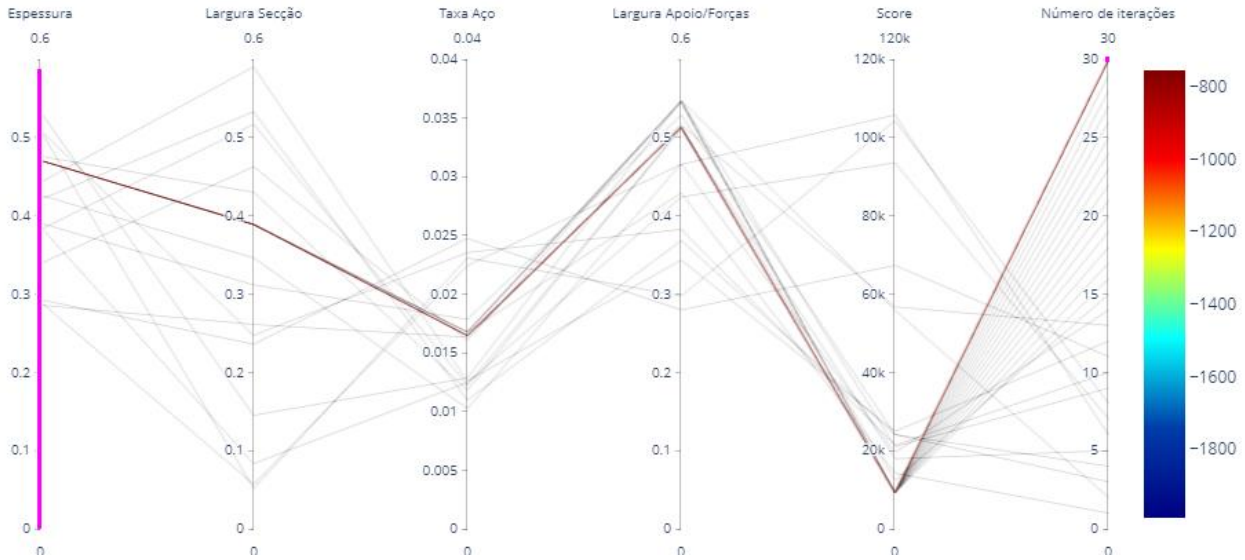


Figura 50: Solução final otimizada.

Quando se faz uma análise mais criteriosa dos resultados, mais precisamente da relação entre os valores atribuídos às variáveis e o número de iterações, é possível através da Figura 51 constatar que após a 15.ª iteração – quando ocorre a troca da população inicial – que o algoritmo de otimização já atribui valores muito próximos daqueles que viriam a ser os resultados ótimos para todas as variáveis. Portanto, concluí-se que, neste caso, uma população inicial (pais) maior contribuirá de forma mais efetiva para a obtenção da melhor solução.

Entretanto, apesar de contribuírem de forma menos expressiva na busca de uma solução ótima e de apresentarem valores muito próximos do resultado final, a partir da 15ª iteração – população de filhos – foi constatada uma pequena variação de valores apenas nas variáveis "Largura Apoios/Força" e "Taxa de armadura". Esses resultados são visíveis nos gráficos presentes nas Figura 52 à Figura 56.

Figura 51: Dimensões vs Número de iterações.

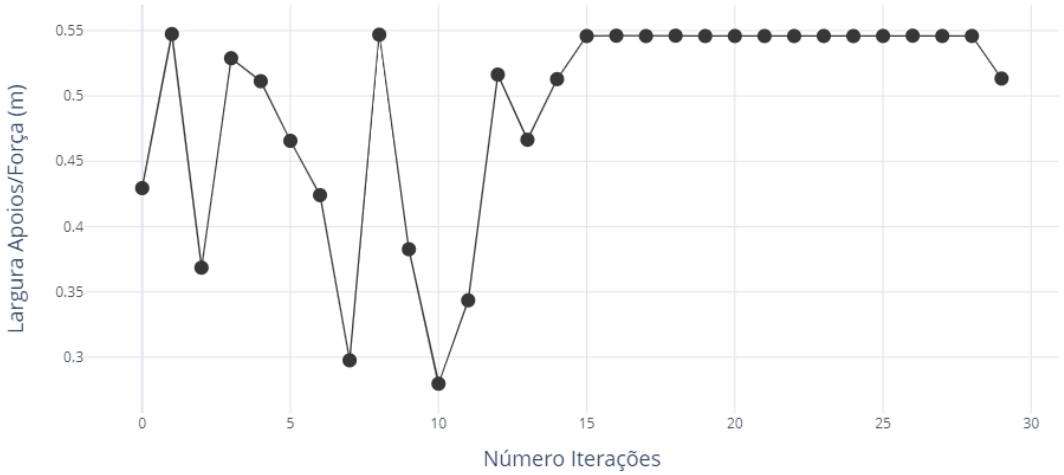


Figura 52: Largura Apoios/Força vs Número de iterações.

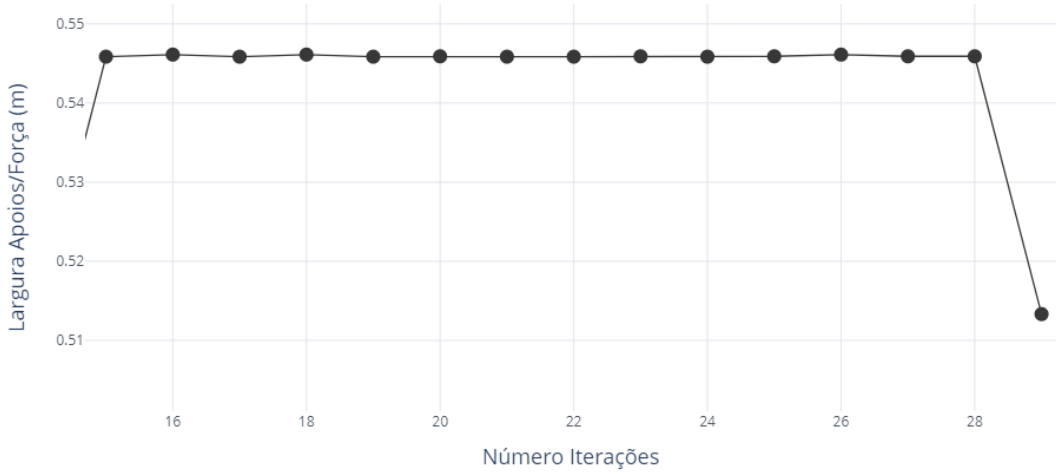


Figura 53: Largura Apoios/Força (após 15ª iteração) vs Número de iterações.

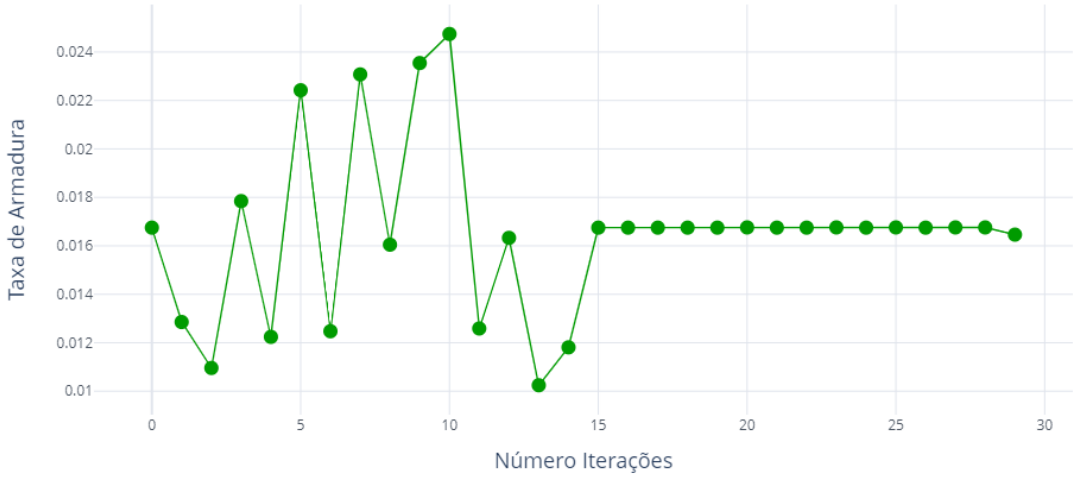


Figura 54: Taxa de armadura vs Número de iterações.

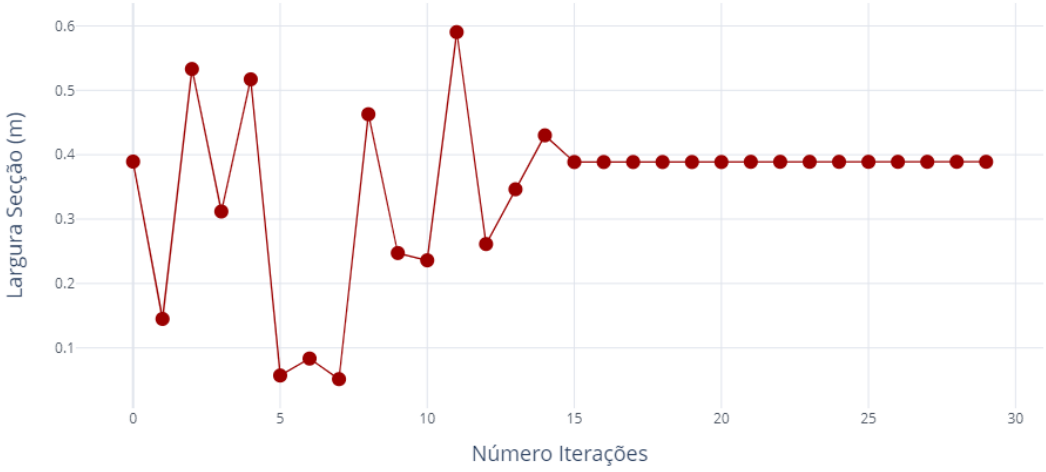


Figura 55: Largura da secção das escoras vs Número de iterações.

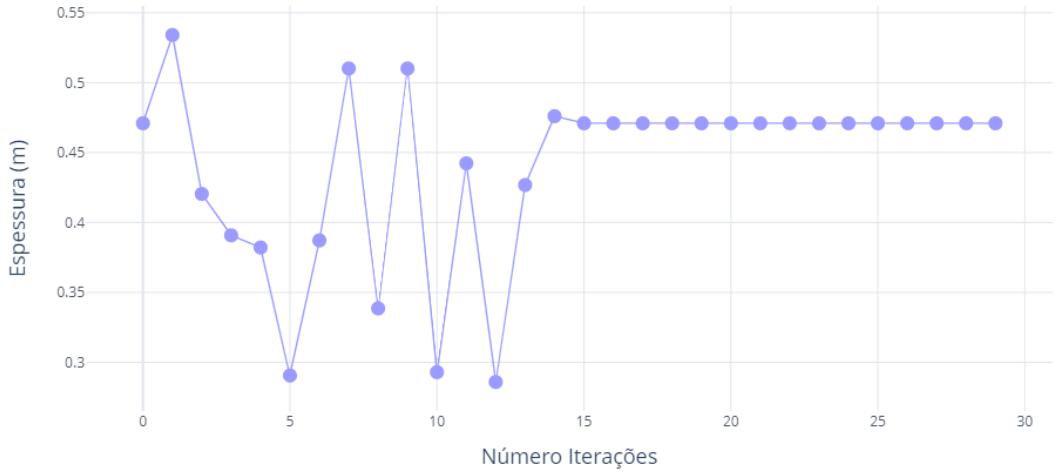


Figura 56: Espessura viga-parede vs Número de iterações.

Percebe-se de imediato que a variável que apresentou maior variação e, portanto, precisou de mais iterações para atingir o valor ótimo foi a “Largura Apoios/Força”. Um dos fatores que pode ter contribuído para esta variação, relaciona-se com o facto desta variável estar diretamente relacionada com a determinação das tensões nas regiões dos apoios e nas regiões onde se aplicam as forças, uma vez que, essas tensões são determinadas a partir da área – determinada em função da espessura da viga-parede e da variável “Largura Apoios/Força”. Ainda, como citado na Secção 4.3.2, para os nós que não atendendam os critérios estabelecidos pela norma ABNT NBR 61118:2014, é atribuída uma penalização de 2 (duas) vezes o “ScoreMaxStress”, contribuindo, portanto, de forma significativa na função objetivo. Além disso, essa variável não exerce influência direta no peso próprio da estrutura. As demais variáveis oscilaram de forma muito semelhante.

Em relação ao “Score” da estrutura para cada conjunto de variáveis, através de uma análise da Figura 57, conclui-se que, assim como o esperado, nas primeiras 15 iterações o “Score” variou de forma considerável. Isto está diretamente relacionado com a aleatoriedade da criação da população inicial.

Entretanto, na parte final (população composta pelos filhos) os resultados permanecem praticamente inalterados pois as variações nos valores das variáveis não são suficientemente grandes para afetar de maneira considerável no “Score” da estrutura.

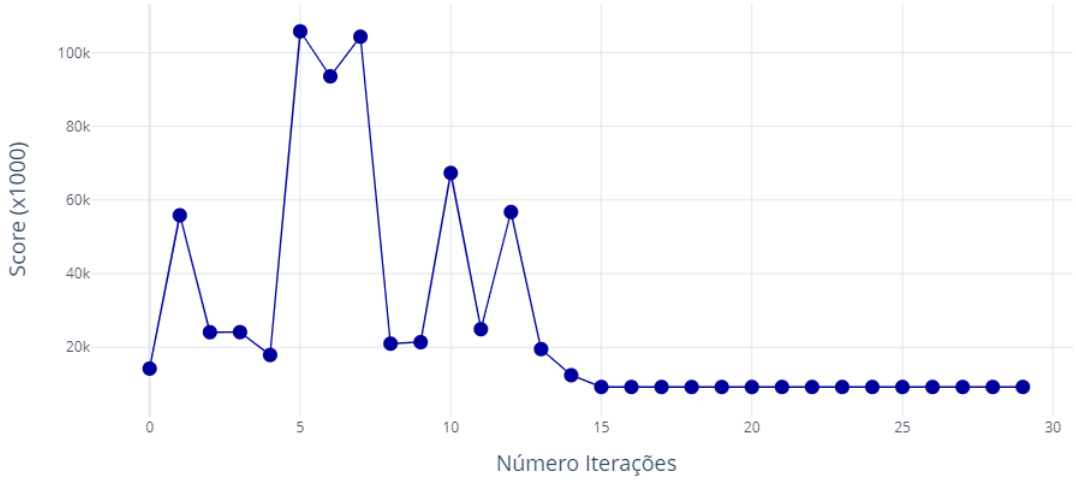


Figura 57: “Score" da estrutura vs Número de iterações.

Após o final de todas estas iterações, obteve-se o resultado ótimo para a estrutura. Como citado anteriormente, devido a limitações de processador e RAM e em função do número de itereações feitas, o processo de otimização durou aproximadamente 5 horas para ser concluído.

Ao analizarmos apenas o “Score” da estrutura, percebe-se que o valor ótimo obtido é ligeiramente maior que o peso próprio, exatamente 800 pontos. Esta diferença deve-se, única e exclusivamente, ao facto de 8 das 18 barras da estrutura apresentarem uma secção transversal maior do que a necessária para atender aos esforços atuantes e, portanto, nos critérios de avaliação encontram-se com um rácio menor do que o limite inferior estabelecido na Secção 4.3.1. Além disso, através da diferença entre o “Score” e o peso próprio da viga-parede conclui-se que todas as regiões nodais encontram-se dentro dos critérios normativos em relação a verificação de tensões e que nenhuma tirante se encontra subdimensionado, pois, de acordo com os critérios de penalização, quando isto ocorre é atribuído ao “Score” da estrutura uma penalidade de duas vezes o “ScoreMaxStress”, ou seja, 2000 pontos.

5. CONCLUSÕES E DESENVOLVIMENTOS FUTUROS

5.1 Considerações finais

Este trabalho pretendeu, por meio da programação em ferramentas utilizadas na metodologia BIM e da programação visual, desenvolver uma aplicação que permite ao utilizador criar de forma paramétrica, com base em condições de contorno pré-definidas, modelos de escoras e tirantes para serem utilizados no pré-dimensionamento de vigas-parede. Modelos estes que são obtidos através de um código Pyhton que se utiliza de critérios de exclusão baseados nos esforços atuantes e no critério de estabilidade global da estrutura. Posteriormente, com base na topologia obtida para o modelo de escoras e tirantes, buscou-se com o auxílio da otimização com algoritmos genéticos através do Optimo, disponível para o Dynamo, pré-dimensionar a estrutura de forma otimizada. Todo este processo só se fez possível graças à interoperabilidade, que permitiu a troca de informações entre o software de cálculo estrutural (Robot) e o software de programação visual Dynamo. Portanto, com base nos resultados obtidos, pode-se concluir que foi possível a definição de modelos de escoras e tirantes e proceder ao respetivo pré-dimensionamento e otimização através da aplicação desenvolvida. Assim, de forma geral, conclui-se que os objetivos inicialmente propostos foram alcançados.

Entretanto, o utilizador da aplicação deve entender que os resultados por ela obtidos baseiam-se em análises preliminares e devem ser tratados como tal. O utilizador deve considerar os resultados como propostas que orientam para áreas de intervenção promissoras. Tendo por base o estudo realizado, foi possível retirar três conclusões principais:

- Do ponto de vista da obtenção do modelo de escoras e tirantes observou-se que quanto menor o número de barras na malha inicial maior a probabilidade de se obter um modelo de escoras e tirantes satisfatório. Além disso, quanto maior o número de barras na malha inicial, mais demorado se torna o processo de obtenção do modelo de escoras e tirantes e, dependendo das condições de contorno, podem surgir algumas falhas

provenientes de instabilidades locais e/ou globais. Estas instabilidades têm a sua origem no elevado número de barras ligadas em cada nó, o que torna a verificação da estabilidade da estrutura mais difícil, uma vez que esta é feita com base no número de barras ligas ao nó. Refira-se que, no decorrer do workflow, a quantidade de barras conectadas a cada nó pode ser alterada;

- Relativamente ao processo de otimização, verificou-se que a ferramenta Optimo também apresentou problemas relacionados com o tempo de execução. Quanto maior a população inicial, maior o tempo necessário para a finalização do processo de otimização. Entretanto, a relação entre o tempo e a população inicial não se dá de forma linear, mas sim, exponencial. Assim, pequenos aumentos na população inicial acarretam um altíssimo tempo de execução.

- Apesar dos problemas relacionados com o tempo de execução do fluxo de trabalho e com o possível surgimento de instabilidades, tanto o código Python quanto a utilização dos algoritmos genéticos apresentaram resultados satisfatórios e compatíveis com aqueles obtidos pelos demais autores, mesmo quando estes se utilizaram de diferentes metodologias.

As conclusões específicas que se podem enumerar são:

- Os modelos de escoras e tirantes obtidos apresentaram um número maior de barras e de nós em comparação com os exemplos da literatura devido aos critérios de exclusão de barras utilizados. Entretanto, as barras "extra", na sua maioria, encontram-se ao longo dos caminhos de carga teóricos, que vão do local onde se aplicam as cargas até aos apoios;

- Com base nos critérios estabelecidos, ao comparar o resultado obtido no caso de estudo, após a utilização do algoritmo genético, verifica-se alguma semelhança com o modelo proposto. Tanto a escora quanto o tirante com maiores secções transversais e, consequentemente com as maiores solicitações de projeto, apresentaram uma área de betão e de aço muito próximas daquelas obtidas pelos autores [66]. Além disso, todos os nós atenderam aos critérios de verificação conforme recomendações normativas da ABNT NBR 6118:2014 [18].

- As variáveis condicionantes do processo de otimização estão relacionadas diretamete com a secção transversal das escoras. Isto ocorre devido à influência da área de betão nas 3 funções de otimização – peso próprio, tensões nas escoras e nós e na taxa de armadura.

5.2 Limitações do trabalho

Ao longo do desenvolvimento do fluxo deste trabalho, foram encontradas limitações de tempo e de memória RAM, designadamente, durante a interoperabilidade entre os softwares Dynamo e Robot. O aumento no tempo de iteração está possivelmente relacionado com o aumento de utilização da memória na RAM com o incremento das iterações. Com uma melhor interação entre o Dynamo e o Robot ou com melhores meios computacionais, será possível uma definição mais rápida dos modelos e, consequentemente, uma otimização e visualização mais eficaz.

Ao longo da dissertação, o Python foi usado para avaliar e analisar partes do fluxo de trabalho. Usar o Python foi de certa forma vantajoso, pois em comparação com outras linguagens de programação, o Python fornece um feedback instantâneo no Dynamo e é mais legível para utilizadores com conhecimentos básicos de linguagens de programação. Além disso, a sua utilização para acesso à API do Robot contribuiu grandemente para a diminuição no tempo de execução das tarefas. Por mais que a linguagem Python e a programação visual do Dynamo tenham faciliado o processo, para um não-programador, o desenvolvimento deste trabalho foi estimulante e desafiador.

Entretanto, a maior dificuldade encontrada foi desenvolver uma metodologia para a criação do modelo de escoras e tirantes a partir de uma malha inicial de barras e nós. Apesar da API do Robot ser acessível por meio de programação, não se tem acesso às funcionalidades desejadas, como por exemplo, às matrizes de rigidez geradas no processo de cálculo da estrutura. Aliado a isso, a grande maioria dos trabalhos científicos disponíveis na literatura, que não utilizam a metodologia FEM, fazem uso de programas de cálculo estrutural onde essas matrizes podem ser acedidas e até mesmo modificadas. Portanto, essa particularidade do software de certa forma acabou por dificultar e limitar o trabalho, onde critérios para a verificação da estabilidade da estrutura tiveram que ser incorporados.

No que diz respeito ao algoritmo de otimização genética, a dificuldade maior deu-se na compreensão do funcionamento da ferramenta Optimo para o Dynamo e no modo como esta se relaciona com o Robot. Por exemplo, uma das premissas deste trabalho era fazer uso somente do Optimo para obtenção do modelo de escoras e tirantes e realizar o pré-dimensionamento do mesmo, entretanto, devido às diversas dificuldades encontradas ao introduzir-se o Python script no Optimo, a solução encontrada foi obter o modelo de forma separada e utilizar o Optimo apenas para o pré-dimesionamento das escoras e dos tirantes. Para além disso, como citado na Secção 2.6, quando se trata de otimização estrutural de treliças, cada membro da estrutura é

considerado como tendo variáveis de projeto separadas, o que não foi possível de se implementar neste trabalho devido aos problemas não identificados na interação entre o Dynamo/Optimo e o Robot. Portanto, as escoras e tirantes foram otimizadas para atender à situação mais desfavorável em termos de esforços. A pouca documentação disponível na literatura acerca deste tema dificultou a realização desta etapa do trabalho, como foi inicialmente idealizada, mas por outro lado, foi possível desenvolver uma metodologia alternativa para alcançar os objetivos inicialmente porpostos.

Por último, é preciso ressaltar o facto de que este trabalho foi realizado num perído deveras complicado para a Humanidade, que enfrentou e ainda enfrenta uma pandemia causada pelo vírus Covid-19. Pandemia esta que indiretamente causou impactos no desenvolvimento da dissertação, pois limitou o acesso aos laboratórios das instituições de ensino envolvidas e, consequentemente, aos hardwares e demais equipamentos imprescindíveis para a sua realização.

5.3 Desenvolvimentos futuros

Este item contém algumas sugestões para a sequência do trabalho desenvolvido. As propostas que se apresentam não estão listadas em nenhuma ordem específica:

- Usar a aplicação desenvolvida na simulação de outros exemplos para validação do módulo de obtenção do modelo de escoras e tirantes. Estes exemplos devem ter diferentes carregamentos e possuir aberturas;
- Na simulação dos exemplos devem ser consideradas diferentes malhas inicias, através da variação do número de barras e de nós;
- Implementar diferentes critérios para a exclusão das barras, de forma a impedir ou minimizar o surgimento de mecanismos de instabilidades locais/globais;
- Investigar a possibilidade de criar um fluxo de trabalho semelhante interagindo com software de análise alternativo onde seja possível aceder, através da API/programação, às matrizes de rigidez utilizadas na obtenção dos esforços e deslocamentos atuantes na estrutura. Desta forma, tornar-se-ia possível, para além dos critérios de exclusão estabelecidos neste trabalho, a implementação de um critério com base nos deslocamentos nodais para obtenção do modelo STM. Além disso, comparações podem

ser feitas avaliando as vantagens e desvantagens de um software em relação ao outro, além de tornar o fluxo de trabalho desenvolvido aplicável para um número mais significativo de utilizadores;

- Alterar o código Python para que se obtenham as secções otimizadas para as escoras e para os tirantes de forma individual, evitando assim o surgimento de secções sobredimensionadas;
- Utilizar o Optimo para a obtenção do modelo de escoras e tirantes, para as verificações e para o dimensionamento de forma simultânea;
- Atualizar o fluxo de trabalho desenvolvido para versões futuras dos softwares. As atualizações futuras resultarão na necessidade de ajustes nos gráficos e scripts desenvolvidos, pois a interoperabilidade entre os vários softwares pode ser alterada;
- Por fim, sabe-se que para o dimensionamento através do modelo de escoras e tirantes não basta somente realizar as verificações das escoras, tirantes e nós. As próximas etapas do projeto das vigas-parede passam pelas verificações relacionadas ao detalhe da secção transversal e longitudinal e determinação das quantidades de armaduras. A ancoragem adequada das barras de aço é um aspeto relevante para o seu desempenho. Sugere-se assim, numa primeira etapa, partindo dos modelos de escoras e tirantes obtidos com a aplicação desenvolvida nesta dissertação, realizar a programação das etapas seguintes de pormenorização estrutural, incluindo as verificações de comprimento mínimo de ancoragem, de emendas de barras, espaçamentos máximos e mínimos das armaduras, permitindo-se a escolha da classe de betão, tipo de aço e do diâmetro dos varões, de forma que atendam à área de aço calculada e às verificações definidas para o betão na região das escoras. Também podem ser incluídas as verificações das armaduras na região dos apoios, bem como as armaduras de pele e de fendilhação, a fim de se executar um projeto completo destas vigas, para condições de vãos e carregamentos pré-definidos.

REFERÊNCIAS

[1] Olesen, J. R. *Intelligent optimization of steel structures in the early design phase through Visual Programming*. 2018. Master Thesis - Departament of Civil Engineering, Technical University of Denmark: Kongens Lyngby, Denmark.

[2] Guan, H. *Strut-and-tie model of deep beams with web openings - An optimization approach*. Structural Engineering and Mechanics, vol. 19, n. 4, p. 361-379, 2005.

[3] Eastman, C., Teicholz, P., Sacks, R. and Liston, K. *BIM Handbook A guide to Building information Modeling for Owner, Manager, Designers, Engineers, and Contractors*. 2011. ISBN 9780470541371.

[4] Azhar, S. *Building Information Modeling (BIM): Trends, Benefits, Risks, and Challenges for the AEC Industry*. 2011. Leadership and Management in Engineering, 11(3), 241-252.

[5] Lino, J. C., Azenha, M., Lourenço, P. B. *Integração da Metodologia BIM na Engenharia de Estruturas*. 2012. Encontro Nacional Betão Estrutural -BE2012, pp. 24–26.

[6] *BIM IS MORE*. N.º1. 2016. Porto: Editorial Director. ISSN 2183-7473

[7] Sharafutdinova A. *BIM in Practice*. 2012. Saimaa University of Applied Sciences: Lappeenranta, Finland.

[8] Architectural/Engineering Productivity Committee of the Construction Users Roundtable (CURT). *Collaboration, Integrated Information and the Project Lifecycle in Building Design, Construction and Operation (WP-1201)*. 2004. Cincinnati, Ohio, United States of America.

[9] *BIM no Projeto de Estruturas e no Planeamento da Construção*. 2016; Disponível em: http://www.csiportugal.com/news_view.php?content_id=72&acn=newlang&lang=en.

[10] Ferreira, P. A. P. *A metodologia BIM enquanto ferramenta no projeto de arquitectura*. 2015. Tese de Mestrado. Faculdade de Arquitetura da Universidade do Porto: Porto, Portugal.

[11] Alves, J. P. C. *Interoperabilidade BIM em projeto de estruturas*. 2018. Master thesis. Universidade de Coimbra: Coimbra, Portugal.

[12] Ferreira, B., Lima, J., Rio, J., Martins, J. P. *Integração da Tecnologia BIM no Projeto de Estruturas de Betão*. 2012. Encontro Nacional Betão Estrutural, pp. 24–26.

[13] Hunt, C. A. *The Benefits of Using Building Information Modeling in Structural Engineering*. 2013. Utah State University: Logan, Utah, United States of America.

[14] Revista AU, edição nº 208. São Paulo, Editora Pini. Julho 2011.

[15] BIM Interoperability - is the industry sailing under false colors?. 2016; Disponível em: http://blog.areo.io/BIM-interoperability/.

[16] Comité Europeu de Normalização. NP-EN_1992-1-1_2010: *Eurocódigo 2: Projeto de estruturas de betão – Parte 1-1: Regras gerais e regras para edifícios*. 2004. Bruxelas, Bélgica.

[17] Rosa, R. D. A. *Ferramenta computacional para o dimensionamento de estruturas laminares de betão armado*. 2010. Master Thesis - Departamento de Engenharia Civil, Faculdade de Engenharia da Universidade do Porto: Porto, Portugal.

[18] Associação Brasileira de Normas Técnicas. *NBR6118:*2014*: Projeto de estruturas de Concreto – Procedimento*, Rio de Janeiro, Brasil.

[19] Leonhardt, F.; Mönnig, E. Construções de concreto: casos especiais de dimensionamento de estruturas de concreto armado. 1. ed. Rio de Janeiro: Interciência, 1978.

[20] Schlaich, J., Schafer, K. *Desing and datailing of structural concrete using strut-and-tie models*. Struct. Eng., v.69, n.6, p.113-125, 1991

[21] American Concrete Institute (ACI) (2002), *Building Code Requirements for Structural Concrete (ACI 318-02) and Commentary - ACI318R-02*, Detroit, Michigan.

[22] Smith, K. N.; Vantsiotis, A. S. Shear strength of deep beams. ACI Structural Journal, Detroit, v. 79, n. 3, p. 201-213, May-June 1982.

[23] Manuel, R. F.; Slight, B. W.; Suter, G. T. Deep beams behavior affected by leghth and shear span variation. ACI Structural Journal, Detroit, v. 68, n. 12, p. 954-958, Dec. 1971.

[24] Oh, J. K., Shin, S. W.,*Shear Strength of Reinforced HighStrength Concrete Deep Beams*. 2001. ACI Structural Journal, V. 98, No. 2, pp. 164-173.

[25] Franco, I., E., F. Vigas-Parede: Comparação entre diferentes metodologias de cálculo. Trabalho de Conclusão de curso (Engenharia Civil) – Departamento de Engenharia Civil, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2015.

[26] Kuehn, A. Comparação entre métodos de análise estrutural para reservatórios retangulares de concreto armado. 2002. 201 f. Dissertação (Mestrado em Engenharia Civil) – Programa de

Pós-Graduação em Engenharia Civil, Universidade Federal de Santa Catarina, Florianópolis, 2002.

[27] Santos, G. G. M. *Análise sistemática de vigas-parede biapoiadas de concreto armado*. Dissertação (Mestrado em Ciências de Engenharia Civil – Estruturas) – Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 1999.

[28] Blaauwendraad, J.; Hoogenboom, P. C. J. Stringer panel model for structural concrete design. ACI Structural Journal, Farmington Hills, MI, USA, v. 93, n. 3, p. 295-305, May-June 1996.

[29] Azevedo, Domingos de. Análise Estrutural com ANSYS Workbench. Mogi das Cruzes: Domingos Flávio de Oliveira Azevedo, 2016. 180 p.

[30] Vanderplaats, G. N. *Numerical Optimization Techniques for Engineering Design with Applications*. EUA: McGraw-Hill, 1984.

[31] Dalfré, G. M. *Cruzetas de polímeros reciclados: caracterização dos materiais, análise numérica e ensaios de modelos reduzidos*. Dissertação de Mestrado. Escola de Engenharia de São Carlos, Universidade de São Paulo: São Carlos - SP, 2007.

[32] Maia, J. P. R. *Otimização estrutural: estudo e aplicações em problemas clássicos debvigas utilizando a ferramenta Solver*. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo: São Carlos - SP, 2009.

[33] Vargas , D.E.C., Lemonge, A.C.C., Barbosac, H.J.C., Bernardino, H.S. Um algoritmo baseado em evolução diferencial para problemas de otimização estrutural multiobjetivo com restrições. Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería, 2015.

[34] Deaton, J.D., Grandhi, R.V. A survy structural and mulidisciplinary continuum topology optimization: post 2000, Struct. Multidisc. Optim. 49, 2014.

[35] Sonmez, M. Discrete optimum design of truss structures using artificial bee colony lgorithm, Struct. Multidisc. Optim. 49, 2011.

[36] Luh, G. C., Lin, C. Y. Structural topology optimization using ant colony optimization algorithm, Appl. Soft Comput. 9, 2009.

[37] Faramarzi A., Afshar, F. M. Application of cellular automata to size and topology optimization of truss structures, Scientia Iranica, 2012.

[38] Li, L. J., Huang, Z. B., Liu, F. A heurisic particle swarm optimization method for truss structures with discrete variables, Comput Struct., 2009.

[39] Gan, B. S., Huang Z. B., Liu, F. Evolutionary ACO algorithms for truss optimization problems, Procedia Engineering 171, 2017.

[40] Holtz G. C. C. *Traçado automático de envoltórias de esforços em estruturas planas utilizando um algoritmo evolucionário*. 2005. Dissertação de Mestrado – Programa de Pós-graduação em Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro: Rio de Janeiro, Brasil.

[41] Kiusalaas, J. *Numerical Methods in Engineering whit MATLAB*. 2005. Cambridge University Press, New York, United States of America.

[42] Antunes, A. F. D. R. *BIM-based Parametric Optimisation of Structural Systems*. 2017. Master Thesis - Civil Engineering, Técnico Lisboa: Lisboa, Portugal.

[43] Darwin, C. On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.1859.

[44] Pavlov, P. *Automation of Information Flow From Revit To Bim Using Dynamo*. 2015. Master Thesis. Aalborg University: Aalborg, Denmark.

[45] Eiben, A E, Raué P -E., Ruttkay Z. *Genetic algorithms with multi-parent recombination*. 1994. Parallel Problem Solving from Nature - PPSN III, International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature, Jerusalem, Israel.

[46] Hayden, M. K. *An experimental investigation of visual enhancements for programming environments*. 1997. California State Polytechnic University: Pomona, California, United States of America.

[47] Vogt, T. M. *Current application of graphical programming in the design phase of a bim project: development opportunities and future scenarios with 'dynamo'*. 2016. Master Thesis - University of Northumbria: Newcastle, England.

[48] Terzidis, K. *Algorithmic architecture*. Routledge. 2006.

[49] Kensek, K. M.; Noble, D. *Building information modeling: BIM in current and future practice*. 2014. Hoboken, New Jersey: Wiley.

[50] Deutsch, R. *Data-driven design and construction: 25 strategies for capturing, analyzing and applying building data*. 2015. Hoboken: John Wiley & Sons Inc.

[51] Shu, N. C. *Visual programming*. 1988. New York: Van Nostrand Reinhold.

[52] Borges, L., E. Python para desenvolvedores. 1 ed. São Paulo: Novatec, outubro de 2014.

[53] Pires, H. J. de C. *Automatização da Modelação Bim de Armaduras no Projeto De Estruturas*. 2017. Master thesis – Engenharia Civil, Instituto Superior De Engenharia Do Porto: Porto, Portugal.

[54] Nezamaldin, D. *Parametric design with Visual Programming in Dynamo with Revit*. 2019. Master thesis - Skolan För Arkitektur Och Samhällsbyggnad, KTH Royal Institute of Technology: Stockholm, Sweden.

[55] Dynamopackages.com. Acessado em 17/11/2021.

[56] Rahmani, M., Stoupine, A., Zarrinmehr, S., Yan, W. *Optimo: A BIM-based Multi-Objective Optimization Tool Utilizing Visual Programming for High Performance Building Design*. 2015. eCAADe: Conference of Education and Research in Computer Aided Architectural Design in Europe 1, pp. 673–682.

[57] Borges, M. M. *Bim modelling automation on the reinforcement detailing of slabs.* 2018. Master thesis - Engenharia Civil, Instituto Superior de Engenharia do Porto: Porto, Portugal.

[58] CEN, "Norma Portuguesa EN 1992-1-1 -Eurocódigo 2 Projecto de estruturas de betão – Parte 1-1: Regras gerais e regras para edifícios", 2010.

[59] Petrovic N., Kostic, N., Marjanovic, N. Comparison of Approaches to 10 Bar Truss Structural Optimization With Included Buckling Constraints. Applied Engineering Letters Vol.2, No.3, 98-103, 2017.

[60] Faustino, A.M., Júdice, J.J., Ribeiro, I. M., Neves, A. S. An integer programming model for truss topology optimization. Investigação Operacional, p. 111-127, 2006.

[61] Standards Association of Australia (SAA) (2002), *AS3600-2002: Concrete Structures*, Sydney, Australia.

[62] Santos, G. G. M. *Análise sistemática de vigas-parede biapoiadas de concreto armado*. Dissertação (Mestrado em Ciências de Engenharia Civil – Estruturas) – Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 1999.

[63] Mello, A. F. A. de, Souza, R. A. de. *Analysis and Design of Reinforced Concrete Deep Beams by a Manual Approach of Stringer-Panel Method*. 2016. Latin American Journal of Solids and Structures.

[64] Tjhin T. N., Kuchma, D.A. *Computer-based tools for design by the strutand-tie method: advances and challenges*. 2002. ACI Struct Journal 99(5):586–594.

[65] FTOOL - Two-dimensional Frame Analysis Tool. Rio de Janeiro, Pontifícia Universidade Católica do Rio de Janeiro. Disponível em: http://www.tecgraf.puc-rio.br/ftool/. Acesso em: 28 jan. 2022

[66] Singh, B., Kaushik, S.K,. Naveen, K.F., Sharma, S. Design Of A Continuous Deep Beam Using The Strut And Tie Method. Asian Journal Of Civil Engineering (Building And Housing), Vol. 7, No. 5,p. 461-477, 2006.

[67] Souza, R. A. De. Concreto Estrutural: análise e dimensionamneto de elementos com descontinuidades. Tese de Doutorado – Escola politécnica de São Paulo, São Paulo, 2004.

[68] Associação Brasileira de Normas Técnicas. *NBR7480: Aço destinado a armaduras para estruturas de concreto armado*. 2007.

APÊNDICE 1

A. PYTHON *SCRIPTS*

1. VERIFICAÇÃO ANGULAR MODELO ESCORA TIRANTE

```
# Carregar as bibliotecas DesignScript e padrão do Python
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
# As entradas para este nó serão armazenadas como uma lista nas variáveis IN.
dataEnteringNode = IN
# Insira o código abaixo desta linha
	ab=IN[0]
	x=IN[1]
	y=IN[2]

	Resultado=0
	#if 1.0<=ab<=2.5:
	if 0.57<=ab<=2:
	    Resultado=x,y
	else:
	    Resultado="REVISAR","REVISAR"
	# Atribua a sua saída para a variável OUT.
	OUT = Resultado
```

2. VERIFICAR RELAÇÃO COMPRIMENTO/ALTURA VIGA-PAREDE

```
# Carregar as bibliotecas DesignScript e padrão do Python
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
# As entradas para este nó serão armazenadas como uma lista nas variáveis IN.
dataEnteringNode = IN
# Insira o código abaixo desta linha
l=IN[0]
h=IN[1]
Resultado=0
if l<(4*h):
```

```
13.      Resultado=l,h
14.  else:
15.      Resultado="REVISAR ALTURA E COMPRIMENTO POIS NÃO É UMA
DEEPBEAM","REVISAR ALTURA E COMPRIMENTO POIS NÃO É UMA DEEPBEAM"
16.  # Atribua a sua saída para a variável OUT.
17.  OUT = Resultado
```

3. CRIAR SECÇÃO NO ROBOT

```
1.   import clr
2.   import time
3.   #add Robot Structural Analysis API reference
4.   from System import Environment
5.   #get the current user folder i .e C:\Users\<you>\AppData\Roaming
6.   user =
Environment.GetFolderPath(Environmetnt.SpecialFolder.ApplicationData)
7.   # add the reference to the interop file shipped with the package
8.   clr.AddReferenceToFileAndPath(user +r"\Dynamo\Dynamo
Revit\2.3\packages\Structural Analysis for
Dynamo\bin\RSA\interop.RobotOM.dll")
9.   # add needed import to be able to use Robot Structural Analysis
objects
10.  from RobotOM import *
11.  from System import Object
12.  app = RobotApplicationClass()
13.  sup = app.Project.Structure.Nodes
14.
15.  #Define output
16.  output = ["Success"]
17.  #Define input
18.  SupportName = IN[0]
19.  SupportData = IN[1]
20.
21.  time.sleep(5)
22.  app.Project.ViewMngr.Refresh()
23.  #Connect to label server
24.  labels = app.Project.Structure.Labels
25.  #Set custom support from input data
26.  outs = labels.Create(IRobotLabelType.I_LT_NODE_SUPPORT, SupportName)
27.  outs.Data.UX = SupportData[0]
28.  outs.Data.UY = SupportData[1]
29.  outs.Data.UZ = SupportData[2]
30.  outs.Data.RX = SupportData[3]
31.  outs.Data.RY = SupportData[4]
32.  outs.Data.RZ = SupportData[5]
33.  outs.Data.Alpha = 0
34.  outs.Data.Beta = 0
35.  outs.Data.Gamma = 0
36.  #Store custom support in active RSA session
37.  labels.Store(outs)
38.
39.
40.  OUT = SupportName
```

4. CRIAR APOIO

```
import clr
import time
#add Robot Structural Analysis API reference
from System import Environment
#get the current user folder i .e C:\Users\<you>\AppData\Roaming
user =
Environment.GetFolderPath(Environmetnt.SpecialFolder.ApplicationData)
# add the reference to the interop file shipped with the package
clr.AddReferenceToFileAndPath(user +r"\Dynamo\Dynamo
Revit\2.3\packages\Structural Analysis for
Dynamo\bin\RSA\interop.RobotOM.dll")
# add needed import to be able to use Robot Structural Analysis
objects
from RobotOM import *
from System import Object
app = RobotApplicationClass()
sup = app.Project.Structure.Nodes

#Define output
output = ["Success"]
#Define input
SupportName = IN[0]
SupportData = IN[1]

time.sleep(5)
app.Project.ViewMngr.Refresh()
#Connect to label server
labels = app.Project.Structure.Labels
#Set custom support from input data
outs = labels.Create(IRobotLabelType.I_LT_NODE_SUPPORT, SupportName)
outs.Data.UX = SupportData[0]
outs.Data.UY = SupportData[1]
outs.Data.UZ = SupportData[2]
outs.Data.RX = SupportData[3]
outs.Data.RY = SupportData[4]
outs.Data.RZ = SupportData[5]
outs.Data.Alpha = 0
outs.Data.Beta = 0
outs.Data.Gamma = 0
#Store custom support in active RSA session
labels.Store(outs)


OUT = SupportName
```

5. OBTER MODELO DE ESCORAS E TIRANTES (PARCIAL)

```
# Carregar as bibliotecas DesignScript e padrão do Python
import sys
sys.path.append( r'C:\Program Files (x86)\IronPython 2.7\Lib')
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
# add Robot Structural Analysis API reference
```

```
from System import Environment
#get the current user folder i .e C:\Users\<you>\AppData\Roaming
user = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)
# add the reference to the interop file shipped with the package
clr.AddReferenceToFileAndPath(user +r"\Dynamo\Dynamo
Revit\2.3\packages\Structural Analysis for
Dynamo\bin\RSA\interop.RobotOM.dll")
# add needed import to be able to use Robot Structural Analysis objects
from RobotOM import *
from System import Object
app = RobotApplicationClass()
app.Project.ViewMngr.Refresh()
import csv
import time
import heapq
from time import gmtime, strftime
from itertools import chain
from collections import defaultdict


app = RobotApplicationClass()
app.Project.ViewMngr.Refresh()
bars = app.Project.Structure.Bars.GetAll()
nodes = app.Project.Structure.Nodes.GetAll()
cases = app.Project.Structure.Cases.GetAll()
labels = app.Project.Structure.Labels
results = app.Project.Structure.Results.Bars.Forces
stress = app.Project.Structure.Results.Bars.Stresses
reações = app.Project.Structure.Results.Nodes.Reactions
deslocamento = app.Project.Structure.Results.Nodes.Displacements

Tempo=IN[0]
NúmeroReações=IN[1]
Nócentrallist=IN[2]
NósemApoioeForças=IN[3]
NóHorizontalVerticallist=IN[4]
listanósforcas=IN[5]
MemberType=IN[6]
ReleaseName=IN[7]
AllowableStressCompressão=IN[8]
AllowableStressTração=IN[9]
altura=IN[10]
Espessura=IN[11]
Nóapoio=IN[12]
BetãoArmado=IN[13]
NumeroNósVertical=IN[14]
NumeroNósHorizontal=IN[15]
ListaNósSuperiores=IN[16]
Secçãobarrastracionadas=IN[17]
Secçãobarrascomprimidas=IN[18]
fcd=IN[19]
EspessuraApoio=IN[20]
LarguraApoio=IN[21]
LarguraForça=IN[22]
EspessuraForça=IN[23]
Nóseforçaas=IN[24]
#VERIFICAR O NÚMERO DE CARREGAMENTOS
numerodecarregamentos=3
#NúmeroReações=3
no1=[]
```

```
adjbars=[]
splitedlista=[]
fownflkdnv=[]
barrajuntar=[]
br1excluir=[]
br2excluir=[]
outputbarras=[]

time.sleep(1)
p=0

for i in range(1, bars.Count+1000):
    if app.Project.Structure.Bars.Exist(i) == -1:
        barras = app.Project.Structure.Bars.Get(i)
        barras.TrussBar=True

app.Project.CalcEngine.GenerateModel()
app.Project.CalcEngine.Calculate()
app.Project.CalcEngine.GenerateModel()
app.Project.ViewMngr.Refresh()
app.Project.CalcEngine.AnalysisParams.AutoVerification=True
app.Project.CalcEngine.AutoFreezeResults = False
app.Project.Structure.ResultsFreeze = False

if p==0:
    if p>=0:
#EXTRAIR RESULTADOS
#SEPARAR BARRAS
        listabarraverticali=[]
        listabarrahorizontali=[]
        listabarradiagonali=[]
        for i in range(1, bars.Count+800):
            if app.Project.Structure.Bars.Exist(i) == -1:
                barra = app.Project.Structure.Bars.Get(i)
                startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
                endnode = app.Project.Structure.Nodes.Get(barra.EndNode)
                Y0=startnode.Y
                Z0=startnode.Z
                Y1=endnode.Y
                Z1=endnode.Z
                if Y0==Y1:
                    listabarraverticali.append(barra.Number)
                if Z0==Z1:
                    listabarrahorizontali.append(barra.Number)
                if Y0!=Y1:
                    if Z0!=Z1:
                        listabarradiagonali.append(barra.Number)
#EXTRAIR RESULTADOS

        saida5=[]
        for i in range(1, bars.Count+800):
            if app.Project.Structure.Bars.Exist(i) == -1:
                barra = app.Project.Structure.Bars.Get(i).Number
                for j in range(1, cases.Count+1):
                    casos= app.Project.Structure.Cases.Get(j).Number
                    forças1 = stress.Value(barra, casos, 1.0)
                    fx1=forças1.FXSX
                    forças2 = stress.Value(barra, casos, 0.0)
                    fx2=forças2.FXSX
```

```
                    saida5.append(max(fx2,fx1))

        splitedlist5=[]
        comprimentoLista=len(saida5)
        sequencia=comprimentoLista/numerodecarregamentos
        for i in range(sequencia):
            start=(i*comprimentoLista/sequencia)
            end=((i+1)*comprimentoLista/sequencia)
            splitedlist5.append(round(sum(saida5[start:end]),2))
#BARRAS VERTICAIS RESULTADO
        saidavertical=[]
        for i,item in enumerate(listabarraverticali):
            if app.Project.Structure.Bars.Exist(item) == -1:
                barra = app.Project.Structure.Bars.Get(item).Number
                for j in range(1, cases.Count+1):
                    casos= app.Project.Structure.Cases.Get(j).Number
                    forças1 = stress.Value(barra, casos, 1.0)
                    fx1=forças1.FXSX
                    forças2 = stress.Value(barra, casos, 0.0)
                    fx2=forças2.FXSX
                    saidavertical.append(max(fx2,fx1))
                splitedlistvertical=[]
                comprimentoListav=len(saidavertical)
                sequenciav=comprimentoListav/numerodecarregamentos
                for i in range(sequenciav):
                    start=(i*comprimentoListav/sequenciav)
                    end=((i+1)*comprimentoListav/sequenciav)

splitedlistvertical.append(round(sum(saidavertical[start:end]),2))

#BARRAS HORIZONTAIS RESULTADOS
        saidahorizontal=[]
        splitedlisthorizontal=[]
        for i,item in enumerate(listabarrahorizontali):
            if app.Project.Structure.Bars.Exist(item) == -1:
                barra = app.Project.Structure.Bars.Get(item).Number
                for j in range(1, cases.Count+1):
                    casos= app.Project.Structure.Cases.Get(j).Number
                    forças1 = stress.Value(barra, casos, 1.0)
                    fx1=forças1.FXSX
                    forças2 = stress.Value(barra, casos, 0.0)
                    fx2=forças2.FXSX
                    saidahorizontal.append(max(fx2,fx1))

                comprimentoListah=len(saidahorizontal)
                sequenciah=comprimentoListah/numerodecarregamentos
                for i in range(sequenciah):
                    start=(i*comprimentoListah/sequenciah)
                    end=((i+1)*comprimentoListah/sequenciah)

splitedlisthorizontal.append(round(sum(saidahorizontal[start:end]),2))


#BARRAS DIAGONAL RESULTADOS
        saidadiagonal=[]
        splitedlistdiagonal=[]
        for i,item in enumerate(listabarradiagonali):
            if app.Project.Structure.Bars.Exist(item) == -1:
                barra = app.Project.Structure.Bars.Get(item).Number
                for j in range(1, cases.Count+1):
```

```
                    casos= app.Project.Structure.Cases.Get(j).Number
                    forças1 = stress.Value(barra, casos, 1.0)
                    fx1=forças1.FXSX
                    forças2 = stress.Value(barra, casos, 0.0)
                    fx2=forças2.FXSX
                    saidadiagonal.append(max(fx2,fx1))

                comprimentoListad=len(saidadiagonal)
                sequenciad=comprimentoListad/numerodecarregamentos
                for i in range(sequenciad):
                    start=(i*comprimentoListad/sequenciad)
                    end=((i+1)*comprimentoListad/sequenciad)

splitedlistdiagonal.append(round(sum(saidadiagonal[start:end]),2))


#FILTRAGEM VALORES
        Resultadoshorizontais=[]
        Indice=[]
        Resultadosverticais=[]
        Resultadosdiagonais=[]
        for i,item in enumerate(splitedlisthorizontal):
            if item >0:
                if abs(item) <= AllowableStressCompressão:
                    Resultadoshorizontais.append(item)
            if item <= 0:
                if abs(item) <= AllowableStressTração:
                    Resultadoshorizontais.append(item)

        for i,item in enumerate(splitedlist5):
            if Resultadoshorizontais.Contains(item):
                Indice.append(i+1)


        for i,item in enumerate(splitedlistvertical):
            if item >0:
                if abs(item) <= AllowableStressCompressão:
                    Resultadosverticais.append(item)
            if item <= 0:
                if abs(item) <= AllowableStressTração:
                    Resultadosverticais.append(item)

        for i,item in enumerate(splitedlist5):
            if Resultadosverticais.Contains(item):
                Indice.append(i+1)


        for i,item in enumerate(splitedlistdiagonal):
            if item >0:
                if abs(item) <= AllowableStressCompressão:
                    Resultadosdiagonais.append(item)
            if item <= 0:
                if abs(item) <= AllowableStressTração:
                    Resultadosdiagonais.append(item)

        for i, item in enumerate(splitedlist5):
            if Resultadosdiagonais.Contains(item):
                Indice.append(i+1)

        for i,item in enumerate(Indice):
```

```
            if item not in Indice:
                Indice.append(item)
```

6. AVALIAR A ESTRUTURA EM FUNÇÃO DAS TENSÕES NAS BARRAS

```
1. # Carregar as bibliotecas DesignScript e padrão do Python
2. import sys
3. sys.path.append( r'C:\Program Files (x86)\IronPython 2.7\Lib')
4. import clr
5. clr.AddReference('ProtoGeometry')
6. from Autodesk.DesignScript.Geometry import *
7. # add Robot Structural Analysis API reference
8. from System import Environment
9. #get the current user folder i .e C:\Users\<you>\AppData\Roaming
10. user =
Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)
11. # add the reference to the interop file shipped with the package
12. clr.AddReferenceToFileAndPath(user +r"\Dynamo\Dynamo
Revit\2.3\packages\Structural Analysis for
Dynamo\bin\RSA\interop.RobotOM.dll")
13. # add needed import to be able to use Robot Structural Analysis objects
14. from RobotOM import *
15. from System import Object
16. app = RobotApplicationClass()
17. app.Project.ViewMngr.Refresh()
18. import csv
19. import time
20. import heapq
21. from time import gmtime, strftime
22. from itertools import chain
23. from collections import defaultdict
24. import math
25.
26. app = RobotApplicationClass()
27. app.Project.ViewMngr.Refresh()
28. bars = app.Project.Structure.Bars.GetAll()
29. nodes = app.Project.Structure.Nodes.GetAll()
30. cases = app.Project.Structure.Cases.GetAll()
31. labels = app.Project.Structure.Labels
32. results = app.Project.Structure.Results.Bars.Forces
33. stress = app.Project.Structure.Results.Bars.Stresses
34. reações = app.Project.Structure.Results.Nodes.Reactions
35. deslocamento = app.Project.Structure.Results.Nodes.Displacements
36.
37. NúmeroReações=IN[0]
38. MemberType=IN[1]
39. AllowableStressCompressãoH=IN[2]
40. AllowableStressTraçãoH=IN[3]
41. AllowableStressCompressãoV=IN[4]
42. AllowableStressTraçãoV=IN[5]
43. AllowableStressCompressãoD=IN[6]
44. AllowableStressTraçãoD=IN[7]
45. penalty_weight_min=IN[8]
46. penalty_weight_max=IN[9]
47. min_ratio=IN[10]
```

```
max_ratio=IN[11]
altura=IN[12]
Espessura=IN[13]
Secçãobarrastracionadas=IN[14]
Secçãobarrascomprimidas=IN[15]
Taxa_aço=IN[16]
fcd=IN[17]
EspessuraApoio=IN[18]
LarguraApoio=IN[19]
LarguraForça=IN[20]
EspessuraForça=IN[21]
Nóseforçaas=IN[22]
ÁreaBarrasTracionadas=IN[23]
fyd=IN[24]
cobrimento=IN[25]
time.sleep(1)
#VERIFICAR O NÚMERO DE CARREGAMENTOS
numerodecarregamentos=3
p=0
for i in range(1, bars.Count+1):
    if app.Project.Structure.Bars.Exist(i) == -1:
        barras = app.Project.Structure.Bars.Get(i)
        barras.TrussBar=True

app.Project.ViewMngr.Refresh()
app.Project.CalcEngine.GenerateModel()
app.Project.CalcEngine.Calculate()
#EXTRAIR RESULTADOS
gefgrvsrfbg=[]
saida=[]
for i in range(1, bars.Count+800):
    if app.Project.Structure.Bars.Exist(i) == -1:
        barra = app.Project.Structure.Bars.Get(i).Number
        gefgrvsrfbg.append(barra)
        for j in range(1, cases.Count+1):
            casos= app.Project.Structure.Cases.Get(j).Number
            forças1 = stress.Value(barra, casos, 1.0)
            fx1=forças1.FXSX
            forças2 = stress.Value(barra, casos, 0.0)
            fx2=forças2.FXSX
            saida.append(max(fx2,fx1))

splitedlist=[]
comprimentoLista=len(saida)
sequencia=comprimentoLista/numerodecarregamentos
for i in range(sequencia):
    start=(i*comprimentoLista/sequencia)
    end=((i+1)*comprimentoLista/sequencia)
    splitedlist.append(round(sum(saida[start:end]),2))

kpvfreobfyhnfkv=[]
for i,item in enumerate(splitedlist):
      kpvfreobfyhnfkv.append(item)

 h=1
 for i,item in enumerate(gefgrvsrfbg):
      kpvfreobfyhnfkv.insert(h,item)
      h=h+2

 spkdmtyjntvsk=[]
```

```
comprimentoLista=len(kpvfreobfyhnfkv)
sequencia=comprimentoLista/2
for i in range(sequencia):
    start=(i*comprimentoLista/sequencia)
    end=((i+1)*comprimentoLista/sequencia)
    spkdmtyjntvsk.append([kpvfreobfyhnfkv[start:end]])

#SEPARAR BARRAS TRACIONADAS DE BARRAS COMPRIMIDAS
barrascomprimidas=[]
barrastracionadas=[]
for i,item in enumerate(spkdmtyjntvsk):
    if spkdmtyjntvsk[i][0][0] >= 0:
        barrascomprimidas.append(spkdmtyjntvsk[i][0][1])
    if spkdmtyjntvsk[i][0][0] <0:
        barrastracionadas.append(spkdmtyjntvsk[i][0][1])

#ATRIBUIR SECÇÃO DE AÇO PARA BARRAS TRACIONADAS
app.Project.ViewMngr.Refresh()
app.Project.CalcEngine.GenerateModel()
if barrastracionadas.Count >=1:
    for i,item in enumerate(barrastracionadas):
        if app.Project.Structure.Bars.Exist(item) == -1:
            barraas = app.Project.Structure.Bars.Get(item)
            barraas.SetLabel(IRobotLabelType.I_LT_BAR_SECTION,
Secçãobarrastracionadas)
            barraas.SetLabel(IRobotLabelType.I_LT_MEMBER_TYPE,
MemberType)
            barraas.SetLabel(IRobotLabelType.I_LT_BAR_RELEASE, "N/A")
            barraas.SetLabel(IRobotLabelType.I_LT_BAR_MATERIAL,
"STEEL")
            barraas.TrussBar=True

app.Project.ViewMngr.Refresh()
app.Project.CalcEngine.GenerateModel()
veovrfnvelrfv=[]
if barrascomprimidas.Count >=1:
    for j,itemj in enumerate(barrascomprimidas):
        if app.Project.Structure.Bars.Exist(itemj) == -1:
            if j>=0:
                barraas = app.Project.Structure.Bars.Get(itemj)
                veovrfnvelrfv.append(barraas)
                barraas.SetLabel(IRobotLabelType.I_LT_BAR_SECTION,
Secçãobarrascomprimidas)
                barraas.SetLabel(IRobotLabelType.I_LT_MEMBER_TYPE,
MemberType)
                barraas.SetLabel(IRobotLabelType.I_LT_BAR_RELEASE,
"N/A")
                barraas.SetLabel(IRobotLabelType.I_LT_BAR_MATERIAL,
"CONCR")
                barraas.TrussBar=True

#FILTRAR OS NÓS DE ACORDO COM O NÚMERO DE BARRAS CONECTADAS
maoi=[]
maoi2=[]
maoi3=[]
maoi4=[]
maoi5=[]
maoi6=[]
for i in range(1, nodes.Count+200):
    if app.Project.Structure.Nodes.Exist(i) == -1:
```

```
161.          nó = app.Project.Structure.Nodes.Get(i)
162.          adjoint =
app.Project.Structure.Nodes.GetConnectedBars(nó.Number)
163.          maoi.append(adjoint)
164.          if adjoint.Count ==2:
165.              maoi2.append(nó.Number)
166.          if adjoint.Count ==3:
167.              maoi3.append(nó.Number)
168.          if adjoint.Count ==4:
169.              maoi4.append(nó.Number)
170.          if adjoint.Count ==5:
171.              maoi5.append(nó.Number)
172.          if adjoint.Count ==6:
173.              maoi6.append(nó.Number)
174.  dois=[]
175.  dois2=[]
176.  tres=[]
177.  tres3=[]
178.  quatro=[]
179.  quatro4=[]
180.  cinco=[]
181.  cinco5=[]
182.  apoio2=[]
183.  apoio22=[]
184.  apoio222=[]
185.  apoiofinal222=[]
186.  for i,item in enumerate(maoi2):
187.      if app.Project.Structure.Nodes.Exist(item) == -1:
188.          nó = app.Project.Structure.Nodes.Get(item)
189.          if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
190.              if nó.Z>0:
191.                  for i in range(1,bars.Count+1000):
192.                      if app.Project.Structure.Bars.Exist(i) == -1:
193.                          barra = app.Project.Structure.Bars.Get(i)
194.                          startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
195.                          endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
196.                          if startnode.Number == nó.Number:
197.                              dois.append(barra.Number)
198.                          if endnode.Number == nó.Number:
199.                              dois.append(barra.Number)
200.          if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
201.              if nó.Z==0:
202.                  for i in range(1,bars.Count+1000):
203.                      if app.Project.Structure.Bars.Exist(i) == -1:
204.                          barra = app.Project.Structure.Bars.Get(i)
205.                          startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
206.                          endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
207.                          if startnode.Number == nó.Number:
208.                              apoio2.append(barra.Number)
209.                          if endnode.Number == nó.Number:
210.                              apoio2.append(barra.Number)
211.
212.      comprimentoLista=len(dois)
213.      sequencia=comprimentoLista/2
214.      for i in range(sequencia):
215.          start=(i*comprimentoLista/sequencia)
```

```
216.            end=((i+1)*comprimentoLista/sequencia)
217.            dois2.append(dois[start:end])
218.
219.
220.        comprimentoLista=len(apoio2)
221.        sequencia=comprimentoLista/2
222.        for i in range(sequencia):
223.            start=(i*comprimentoLista/sequencia)
224.            end=((i+1)*comprimentoLista/sequencia)
225.            apoio22.append(apoio2[start:end])
226.
227.
228.        for sublist in apoio22:
229.            for item in sublist:
230.                if item not in apoio222:
231.                    apoio222.append(item)
232.
233.        comprimentoLista=len(apoio222)
234.        sequencia=comprimentoLista/2
235.        for i in range(sequencia):
236.            start=(i*comprimentoLista/sequencia)
237.            end=((i+1)*comprimentoLista/sequencia)
238.            apoiofinal222.append(apoio222[start:end])
239.
240.
241.    apoio33=[]
242.    apoio3=[]
243.    apoiofinal333=[]
244.    apoio333=[]
245.    for i,item in enumerate(maoi3):
246.        if app.Project.Structure.Nodes.Exist(item) == -1:
247.            nó = app.Project.Structure.Nodes.Get(item)
248.            if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
249.                if nó.Z>0:
250.                    for i in range(1,bars.Count+1000):
251.                        if app.Project.Structure.Bars.Exist(i) == -1:
252.                            barra = app.Project.Structure.Bars.Get(i)
253.                            startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
254.                            endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
255.                            if startnode.Number == nó.Number:
256.                                tres.append(barra.Number)
257.                            if endnode.Number == nó.Number:
258.                                tres.append(barra.Number)
259.            if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
260.                if nó.Z==0:
261.                    for i in range(1,bars.Count+1000):
262.                        if app.Project.Structure.Bars.Exist(i) == -1:
263.                            barra = app.Project.Structure.Bars.Get(i)
264.                            startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
265.                            endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
266.                            if startnode.Number == nó.Number:
267.                                apoio3.append(barra.Number)
268.                            if endnode.Number == nó.Number:
269.                                apoio3.append(barra.Number)
270.        comprimentoLista=len(tres)
271.        sequencia=comprimentoLista/3
```

```
272.      for i in range(sequencia):
273.          start=(i*comprimentoLista/sequencia)
274.          end=((i+1)*comprimentoLista/sequencia)
275.          tres3.append(tres[start:end])
276.
277.
278.      comprimentoLista=len(apoio3)
279.      sequencia=comprimentoLista/3
280.      for i in range(sequencia):
281.          start=(i*comprimentoLista/sequencia)
282.          end=((i+1)*comprimentoLista/sequencia)
283.          apoio33.append(apoio3[start:end])
284.
285.
286.      for sublist in apoio33:
287.          for item in sublist:
288.              if item not in apoio333:
289.                  apoio333.append(item)
290.
291.      comprimentoLista=len(apoio333)
292.      sequencia=comprimentoLista/3
293.      for i in range(sequencia):
294.          start=(i*comprimentoLista/sequencia)
295.          end=((i+1)*comprimentoLista/sequencia)
296.          apoiofinal333.append(apoio333[start:end])
297.
298.  apoio44=[]
299.  apoio4=[]
300.  apoiofinal444=[]
301.  apoio444=[]
302.  for i,item in enumerate(maoi4):
303.      if app.Project.Structure.Nodes.Exist(item) == -1:
304.          nó = app.Project.Structure.Nodes.Get(item)
305.          if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
306.              if nó.Z>0:
307.                  for i in range(1,bars.Count+1000):
308.                      if app.Project.Structure.Bars.Exist(i) == -1:
309.                          barra = app.Project.Structure.Bars.Get(i)
310.                          startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
311.                          endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
312.                          if startnode.Number == nó.Number:
313.                              quatro.append(barra.Number)
314.                          if endnode.Number == nó.Number:
315.                              quatro.append(barra.Number)
316.          if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
317.              if nó.Z==0:
318.                  for i in range(1,bars.Count+1000):
319.                      if app.Project.Structure.Bars.Exist(i) == -1:
320.                          barra = app.Project.Structure.Bars.Get(i)
321.                          startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
322.                          endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
323.                          if startnode.Number == nó.Number:
324.                              apoio4.append(barra.Number)
325.                          if endnode.Number == nó.Number:
326.                              apoio4.append(barra.Number)
327.      comprimentoLista=len(quatro)
```

```
328.        sequencia=comprimentoLista/4
329.        for i in range(sequencia):
330.            start=(i*comprimentoLista/sequencia)
331.            end=((i+1)*comprimentoLista/sequencia)
332.            quatro4.append(quatro[start:end])
333.
334.
335.        comprimentoLista=len(apoio4)
336.        sequencia=comprimentoLista/4
337.        for i in range(sequencia):
338.            start=(i*comprimentoLista/sequencia)
339.            end=((i+1)*comprimentoLista/sequencia)
340.            apoio44.append(apoio4[start:end])
341.
342.
343.        for sublist in apoio44:
344.            for item in sublist:
345.                if item not in apoio444:
346.                    apoio444.append(item)
347.
348.        comprimentoLista=len(apoio444)
349.        sequencia=comprimentoLista/4
350.        for i in range(sequencia):
351.            start=(i*comprimentoLista/sequencia)
352.            end=((i+1)*comprimentoLista/sequencia)
353.            apoiofinal444.append(apoio444[start:end])
354.
355.    apoio55=[]
356.    apoio5=[]
357.    apoiofinal555=[]
358.    apoio555=[]
359.
360.    app.Project.ViewMngr.Refresh()
361.    app.Project.CalcEngine.GenerateModel()
362.    for i,item in enumerate(maoi5):
363.        if app.Project.Structure.Nodes.Exist(item) == -1:
364.            nó = app.Project.Structure.Nodes.Get(item)
365.            #if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) == 0:
366.            if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
367.                if nó.Z>0:
368.                    for i in range(1,bars.Count+1000):
369.                        if app.Project.Structure.Bars.Exist(i) == -1:
370.                            barra = app.Project.Structure.Bars.Get(i)
371.                            startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
372.                            endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
373.                            if startnode.Number == nó.Number:
374.                                cinco.append(barra.Number)
375.                            if endnode.Number == nó.Number:
376.                                cinco.append(barra.Number)
377.            if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
378.                if nó.Z==0:
379.                    for i in range(1,bars.Count+1000):
380.                        if app.Project.Structure.Bars.Exist(i) == -1:
381.                            barra = app.Project.Structure.Bars.Get(i)
382.                            startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
383.                            endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
```

```
                            if startnode.Number == nó.Number:
                                apoio5.append(barra.Number)
                            if endnode.Number == nó.Number:
                                apoio5.append(barra.Number)
        comprimentoLista=len(cinco)
        sequencia=comprimentoLista/5
        for i in range(sequencia):
            start=(i*comprimentoLista/sequencia)
            end=((i+1)*comprimentoLista/sequencia)
            cinco5.append(cinco[start:end])


        comprimentoLista=len(apoio5)
        sequencia=comprimentoLista/5
        for i in range(sequencia):
            start=(i*comprimentoLista/sequencia)
            end=((i+1)*comprimentoLista/sequencia)
            apoio55.append(apoio5[start:end])


        for sublist in apoio55:
            for item in sublist:
                if item not in apoio555:
                    apoio555.append(item)

        comprimentoLista=len(apoio555)
        sequencia=comprimentoLista/5
        for i in range(sequencia):
            start=(i*comprimentoLista/sequencia)
            end=((i+1)*comprimentoLista/sequencia)
            if apoio555[start:end] not in apoiofinal555:
                apoiofinal555.append(apoio555[start:end])
    seis=[]
    seis6=[]
    apoio66=[]
    apoio6=[]
    apoiofinal666=[]
    apoio666=[]
    for i,item in enumerate(maoi6):
        if app.Project.Structure.Nodes.Exist(item) == -1:
            nó = app.Project.Structure.Nodes.Get(item)
            if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
                if nó.Z>0:
                    for i in range(1,bars.Count+1000):
                        if app.Project.Structure.Bars.Exist(i) == -1:
                            barra = app.Project.Structure.Bars.Get(i)
                            startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
                            endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
                            if startnode.Number == nó.Number:
                                seis.append(barra.Number)
                            if endnode.Number == nó.Number:
                                seis.append(barra.Number)
            if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
                if nó.Z==0:
                    for i in range(1,bars.Count+1000):
                        if app.Project.Structure.Bars.Exist(i) == -1:
                            barra = app.Project.Structure.Bars.Get(i)
```

```
                            startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
                            endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
                            if startnode.Number == nó.Number:
                                apoio6.append(barra.Number)
                            if endnode.Number == nó.Number:
                                apoio6.append(barra.Number)

        comprimentoLista=len(seis)
        sequencia=comprimentoLista/6
        for i in range(sequencia):
            start=(i*comprimentoLista/sequencia)
            end=((i+1)*comprimentoLista/sequencia)
            seis6.append(seis[start:end])


        comprimentoLista=len(apoio6)
        sequencia=comprimentoLista/6
        for i in range(sequencia):
            start=(i*comprimentoLista/sequencia)
            end=((i+1)*comprimentoLista/sequencia)
            apoio66.append(apoio6[start:end])


        for sublist in apoio66:
            for item in sublist:
                if item not in apoio666:
                    apoio666.append(item)

        comprimentoLista=len(apoio666)
        sequencia=comprimentoLista/6
        for i in range(sequencia):
            start=(i*comprimentoLista/sequencia)
            end=((i+1)*comprimentoLista/sequencia)
            if apoio666[start:end] not in apoiofinal666:
                apoiofinal666.append(apoio666[start:end])

    app.Project.CalcEngine.GenerateModel()
    app.Project.CalcEngine.Calculate()
    app.Project.ViewMngr.Refresh()
    numerodecarregamentos=3
    p=0
    if p<=0:
        if p>=0:
    #EXTRAIR RESULTADOS
            saida123=[]
            kpvfreokv=[]
            for i in range(1, bars.Count+1000):
                if app.Project.Structure.Bars.Exist(i) == -1:
                    barra = app.Project.Structure.Bars.Get(i).Number
                    kpvfreokv.append(barra)
                    for j in range(1, cases.Count+1):
                        casos= app.Project.Structure.Cases.Get(j).Number
                        forças1 = stress.Value(barra, casos, 1.0)
                        fx1=forças1.FXSX
                        forças2 = stress.Value(barra, casos, 0.0)
                        fx2=forças2.FXSX
                        saida123.append(max(fx2,fx1))

```

```
499.          splitedlist123=[]
500.          comprimentoLista=len(saida123)
501.          sequencia=comprimentoLista/numerodecarregamentos
502.          for i in range(sequencia):
503.              start=(i*comprimentoLista/sequencia)
504.              end=((i+1)*comprimentoLista/sequencia)
505.              splitedlist123.append(round(sum(saida123[start:end]),2))
506.
507.
508.          h=1
509.          for i,item in enumerate(splitedlist123):
510.              kpvfreokv.insert(h,item)
511.              h=h+2
512.          spkdmvsk=[]
513.          comprimentoLista=len(kpvfreokv)
514.          sequencia=comprimentoLista/2
515.          for i in range(sequencia):
516.              start=(i*comprimentoLista/sequencia)
517.              end=((i+1)*comprimentoLista/sequencia)
518.              spkdmvsk.append([kpvfreokv[start:end]])
519.
520.  #SEPARAR BARRAS TRACIONADAS DE BARRAS COMPRIMIDAS
521.          barrascomprimidasavaliaçãonós=[]
522.          barrastracionadasavaliaçãonós=[]
523.          for i,item in enumerate(spkdmvsk):
524.              if spkdmvsk[i][0][1] >= 0:
525.
barrascomprimidasavaliaçãonós.append(spkdmvsk[i][0][0])
526.              if spkdmvsk[i][0][1] <0:
527.
barrastracionadasavaliaçãonós.append(spkdmvsk[i][0][0])
528.
529.
530.
531.  doiscomp=[]
532.  doistr=[]
533.  CCC2=[]
534.  CCT2=[]
535.  CTTeTTT2=[]
536.  for af,item in enumerate(dois2):
537.      if dois2[af][0] in barrascomprimidasavaliaçãonós:
538.          doiscomp.append(item[0])
539.      if dois2[af][0] in barrastracionadasavaliaçãonós:
540.          doistr.append(item[0])
541.      if dois2[af][1] in barrascomprimidasavaliaçãonós:
542.          doiscomp.append(item[1])
543.      if dois2[af][1] in barrastracionadasavaliaçãonós:
544.          doistr.append(item[1])
545.
546.
547.      if doiscomp.Count>=doistr.Count:
548.          if doistr.Count==1:
549.              for i,item in enumerate(doiscomp):
550.                  if app.Project.Structure.Bars.Exist(item) == -1:
551.                      barra = app.Project.Structure.Bars.Get(item)
552.                      startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
553.                      endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
554.                      for j,itemj in enumerate(doistr):
```

```
555.                            if app.Project.Structure.Bars.Exist(itemj) ==
-1:
556.                                barraj =
app.Project.Structure.Bars.Get(itemj)
557.                                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
558.                                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
559.                                if startnode.Number == startnodej.Number:
560.                                    if startnode.Number not in CCC2:
561.                                        CCT2.append(startnode.Number)
562.                                if startnode.Number == endnodej.Number:
563.                                    if startnode.Number not in CCC2:
564.                                        CCT2.append(startnode.Number)
565.                                if endnode.Number == startnodej.Number:
566.                                    if endnode.Number not in CCC2:
567.                                        CCT2.append(startnode.Number)
568.                                if endnode.Number == endnodej.Number:
569.                                    if endnode.Number not in CCC2:
570.                                        CCT2.append(startnode.Number)
571.                doiscomp.Clear()
572.                doistr.Clear()
573.            if doistr.Count>1:
574.                for i,item in enumerate(doiscomp):
575.                    if app.Project.Structure.Bars.Exist(item) == -1:
576.                        barra = app.Project.Structure.Bars.Get(item)
577.                        startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
578.                        endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
579.                        for j,itemj in enumerate(doistr):
580.                            if app.Project.Structure.Bars.Exist(itemj) ==
-1:
581.                                barraj =
app.Project.Structure.Bars.Get(itemj)
582.                                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
583.                                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
584.                                if startnode.Number == startnodej.Number:
585.                                    if startnode.Number not in CTTeTTT2:
586.                                        CTTeTTT2.append(startnode.Number)
587.                                if startnode.Number == endnodej.Number:
588.                                    if startnode.Number not in CTTeTTT2:
589.                                        CTTeTTT2.append(startnode.Number)
590.                                if endnode.Number == startnodej.Number:
591.                                    if endnode.Number not in CTTeTTT2:
592.                                        CTTeTTT2.append(startnode.Number)
593.                                if endnode.Number == endnodej.Number:
594.                                    if endnode.Number not in CTTeTTT2:
595.                                        CTTeTTT2.append(startnode.Number)
596.                doiscomp.Clear()
597.                doistr.Clear()
598.            m=0
599.            if doistr.Count==0:
600.                for i,item in enumerate(doiscomp):
601.                    if app.Project.Structure.Bars.Exist(doiscomp[0]) == -
1:
602.                        barra =
app.Project.Structure.Bars.Get(doiscomp[0])
```

```
603.                        startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
604.                        endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
605.                        if m<=0:
606.                            if
app.Project.Structure.Bars.Exist(doiscomp[1]) == -1:
607.                                barraj =
app.Project.Structure.Bars.Get(doiscomp[1])
608.                                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
609.                                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
610.                                if startnode.Number == startnodej.Number:
611.                                    if startnode.Number not in CCC2:
612.                                        CCC2.append(startnode.Number)
613.                                if startnode.Number == endnodej.Number:
614.                                    if startnode.Number not in CCC2:
615.                                        CCC2.append(startnode.Number)
616.                                if endnode.Number == startnodej.Number:
617.                                    if endnode.Number not in CCC2:
618.                                        CCC2.append(endnode.Number)
619.                                if endnode.Number == endnodej.Number:
620.                                    if endnode.Number not in CCC2:
621.                                        CCC2.append(endnode.Number)
622.                                m=m+1
623.                doiscomp.Clear()
624.                doistr.Clear()
625.
626.
627.
628.        if doiscomp.Count<doistr.Count:
629.            if doiscomp.Count==0:
630.                for i,item in enumerate(doiscomp):
631.                    if app.Project.Structure.Bars.Exist(item) == -1:
632.                        barra = app.Project.Structure.Bars.Get(item)
633.                        startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
634.                        endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
635.                        for j,itemj in enumerate(doistr):
636.                            if app.Project.Structure.Bars.Exist(itemj) ==
-1:
637.                                barraj =
app.Project.Structure.Bars.Get(itemj)
638.                                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
639.                                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
640.                                if startnode.Number == startnodej.Number:
641.                                    if startnode.Number not in CTTeTTT2:
642.                                        CTTeTTT2.append(startnode.Number)
643.                                if startnode.Number == endnodej.Number:
644.                                    if startnode.Number not in CTTeTTT2:
645.                                        CTTeTTT2.append(startnode.Number)
646.                                if endnode.Number == startnodej.Number:
647.                                    if endnode.Number not in CTTeTTT2:
648.                                        CTTeTTT2.append(startnode.Number)
649.                                if endnode.Number == endnodej.Number:
650.                                    if endnode.Number not in CTTeTTT2:
```

```
651.                                        CTTeTTT2.append(startnode.Number)
652.
653.
654.
655.  trescomp=[]
656.  trestr=[]
657.  trescomp1=[]
658.  trestr1=[]
659.  CCC3=[]
660.  CCT3=[]
661.  CTTeTTT3=[]
662.  for af,item in enumerate(tres3):
663.      if tres3[af][0] in barrascomprimidasavaliaçãonós:
664.          trescomp.append(item[0])
665.          trescomp1.append(item[0])
666.      if tres3[af][0] in barrastracionadasavaliaçãonós:
667.          trestr.append(item[0])
668.          trestr1.append(item[0])
669.      if tres3[af][1] in barrascomprimidasavaliaçãonós:
670.          trescomp.append(item[1])
671.          trescomp1.append(item[1])
672.      if tres3[af][1] in barrastracionadasavaliaçãonós:
673.          trestr.append(item[1])
674.          trestr1.append(item[1])
675.      if tres3[af][2] in barrascomprimidasavaliaçãonós:
676.          trescomp.append(item[2])
677.          trescomp1.append(item[2])
678.      if tres3[af][2] in barrastracionadasavaliaçãonós:
679.          trestr.append(item[2])
680.          trestr1.append(item[2])
681.      if trescomp.Count>=trestr.Count:
682.          if trestr.Count==1:
683.              for i,item in enumerate(trescomp):
684.                  if app.Project.Structure.Bars.Exist(item) == -1:
685.                      barra = app.Project.Structure.Bars.Get(item)
686.                      startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
687.                      endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
688.                      for j,itemj in enumerate(trestr):
689.                          if app.Project.Structure.Bars.Exist(itemj) ==
-1:
690.                              barraj =
app.Project.Structure.Bars.Get(itemj)
691.                              startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
692.                              endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
693.                              if startnode.Number == startnodej.Number:
694.                                  if startnode.Number not in CCT3:
695.                                      CCT3.append(startnode.Number)
696.                              if startnode.Number == endnodej.Number:
697.                                  if startnode.Number not in CCT3:
698.                                      CCT3.append(startnode.Number)
699.                              if endnode.Number == startnodej.Number:
700.                                  if endnode.Number not in CCT3:
701.                                      CCT3.append(endnode.Number)
702.                              if endnode.Number == endnodej.Number:
703.                                  if endnode.Number not in CCT3:
704.                                      CCT3.append(endnode.Number)
```

```
705.                trescomp.Clear()
706.                trestr.Clear()
707.            if trestr.Count>1:
708.                for i,item in enumerate(trescomp):
709.                    if app.Project.Structure.Bars.Exist(item) == -1:
710.                        barra = app.Project.Structure.Bars.Get(item)
711.                        startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
712.                        endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
713.                        for j,itemj in enumerate(trestr):
714.                            if app.Project.Structure.Bars.Exist(itemj) ==
-1:
715.                                barraj =
app.Project.Structure.Bars.Get(itemj)
716.                                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
717.                                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
718.                                if startnode.Number == startnodej.Number:
719.                                    if startnode.Number not in CTTeTTT3:
720.                                        CTTeTTT3.append(startnode.Number)
721.                                if startnode.Number == endnodej.Number:
722.                                    if startnode.Number not in CTTeTTT3:
723.                                        CTTeTTT3.append(startnode.Number)
724.                                if endnode.Number == startnodej.Number:
725.                                    if endnode.Number not in CTTeTTT3:
726.                                        CTTeTTT3.append(endnode.Number)
727.                                if endnode.Number == endnodej.Number:
728.                                    if endnode.Number not in CTTeTTT3:
729.                                        CTTeTTT3.append(endnode.Number)
730.                trescomp.Clear()
731.                trestr.Clear()
732.            m=0
733.            if trestr.Count==0:
734.                for i,item in enumerate(trescomp):
735.                    if app.Project.Structure.Bars.Exist(trescomp[0]) == -
1:
736.                        barra =
app.Project.Structure.Bars.Get(trescomp[0])
737.                        startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
738.                        endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
739.                        if m<=0:
740.                            if
app.Project.Structure.Bars.Exist(trescomp[1]) == -1:
741.                                barraj =
app.Project.Structure.Bars.Get(trescomp[1])
742.                                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
743.                                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
744.                                if startnode.Number == startnodej.Number:
745.                                    if startnode.Number not in CCC3:
746.                                        CCC3.append(startnode.Number)
747.                                if startnode.Number == endnodej.Number:
748.                                    if startnode.Number not in CCC3:
749.                                        CCC3.append(startnode.Number)
750.                                if endnode.Number == startnodej.Number:
```

```
751.                                    if endnode.Number not in CCC3:
752.                                        CCC3.append(endnode.Number)
753.                                if endnode.Number == endnodej.Number:
754.                                    if endnode.Number not in CCC3:
755.                                        CCC3.append(endnode.Number)
756.                                m=m+1
757.          trescomp.Clear()
758.          trestr.Clear()
759.      if trescomp.Count<trestr.Count:
760.          for i,item in enumerate(trescomp):
761.              if app.Project.Structure.Bars.Exist(item) == -1:
762.                  barra = app.Project.Structure.Bars.Get(item)
763.                  startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
764.                  endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
765.                  for j,itemj in enumerate(trestr):
766.                      if app.Project.Structure.Bars.Exist(itemj) == -1:
767.                          barraj =
app.Project.Structure.Bars.Get(itemj)
768.                          startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
769.                          endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
770.                          if startnode.Number == startnodej.Number:
771.                              if startnode.Number not in CTTeTTT3:
772.                                  CTTeTTT3.append(startnode.Number)
773.                          if startnode.Number == endnodej.Number:
774.                              if startnode.Number not in CTTeTTT3:
775.                                  CTTeTTT3.append(startnode.Number)
776.                          if endnode.Number == startnodej.Number:
777.                              if endnode.Number not in CTTeTTT3:
778.                                  CTTeTTT3.append(endnode.Number)
779.                          if endnode.Number == endnodej.Number:
780.                              if endnode.Number not in CTTeTTT3:
781.                                  CTTeTTT3.append(endnode.Number)
782.          trescomp.Clear()
783.          trestr.Clear()
784.
785.  quatrocomp=[]
786.  quatrotr=[]
787.  quatrocomp1=[]
788.  quatrotr1=[]
789.  CCC4=[]
790.  CCT4=[]
791.  CTTeTTT4=[]
792.  for af,item in enumerate(quatro4):
793.      if quatro4[af][0] in barrascomprimidasavaliaçãonós:
794.          quatrocomp.append(item[0])
795.          quatrocomp1.append(item[0])
796.      if quatro4[af][0] in barrastracionadasavaliaçãonós:
797.          quatrotr.append(item[0])
798.          quatrotr1.append(item[0])
799.      if quatro4[af][1] in barrascomprimidasavaliaçãonós:
800.          quatrocomp.append(item[1])
801.          quatrocomp1.append(item[1])
802.      if quatro4[af][1] in barrastracionadasavaliaçãonós:
803.          quatrotr.append(item[1])
804.          quatrotr1.append(item[1])
805.      if quatro4[af][2] in barrascomprimidasavaliaçãonós:
```

```
806.            quatrocomp.append(item[2])
807.            quatrocomp1.append(item[2])
808.        if quatro4[af][2] in barrastracionadasavaliaçãonós:
809.            quatrotr.append(item[2])
810.            quatrotr1.append(item[2])
811.        if quatro4[af][3] in barrascomprimidasavaliaçãonós:
812.            quatrocomp.append(item[3])
813.            quatrocomp1.append(item[3])
814.        if quatro4[af][3] in barrastracionadasavaliaçãonós:
815.            quatrotr.append(item[3])
816.            quatrotr1.append(item[3])
817.
818.
819.        if quatrocomp.Count>=quatrotr.Count:
820.            if quatrotr.Count==1:
821.                for i,item in enumerate(quatrocomp):
822.                    if app.Project.Structure.Bars.Exist(item) == -1:
823.                        barra = app.Project.Structure.Bars.Get(item)
824.                        startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
825.                        endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
826.                        for j,itemj in enumerate(quatrotr):
827.                            if app.Project.Structure.Bars.Exist(itemj) ==
-1:
828.                                barraj =
app.Project.Structure.Bars.Get(itemj)
829.                                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
830.                                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
831.                                if startnode.Number == startnodej.Number:
832.                                    if startnode.Number not in CCT4:
833.                                        CCT4.append(startnode.Number)
834.                                if startnode.Number == endnodej.Number:
835.                                    if startnode.Number not in CCT4:
836.                                        CCT4.append(startnode.Number)
837.                                if endnode.Number == startnodej.Number:
838.                                    if endnode.Number not in CCT4:
839.                                        CCT4.append(endnode.Number)
840.                                if endnode.Number == endnodej.Number:
841.                                    if endnode.Number not in CCT4:
842.                                        CCT4.append(endnode.Number)
843.                quatrocomp.Clear()
844.                quatrotr.Clear()
845.            if quatrotr.Count>1:
846.                for i,item in enumerate(quatrocomp):
847.                    if app.Project.Structure.Bars.Exist(item) == -1:
848.                        barra = app.Project.Structure.Bars.Get(item)
849.                        startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
850.                        endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
851.                        for j,itemj in enumerate(quatrotr):
852.                            if app.Project.Structure.Bars.Exist(itemj) ==
-1:
853.                                barraj =
app.Project.Structure.Bars.Get(itemj)
854.                                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
```

```
855.                                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
856.                                if startnode.Number == startnodej.Number:
857.                                    if startnode.Number not in CTTeTTT4:
858.                                        CTTeTTT4.append(startnode.Number)
859.                                if startnode.Number == endnodej.Number:
860.                                    if startnode.Number not in CTTeTTT4:
861.                                        CTTeTTT4.append(startnode.Number)
862.                                if endnode.Number == startnodej.Number:
863.                                    if endnode.Number not in CTTeTTT4:
864.                                        CTTeTTT4.append(endnode.Number)
865.                                if endnode.Number == endnodej.Number:
866.                                    if endnode.Number not in CTTeTTT4:
867.                                        CTTeTTT4.append(endnode.Number)
868.                quatrocomp.Clear()
869.                quatrotr.Clear()
870.            m=0
871.            if quatrotr.Count==0:
872.                for i,item in enumerate(quatrocomp):
873.                    if app.Project.Structure.Bars.Exist(quatrocomp[0]) ==
-1:
874.                        barra =
app.Project.Structure.Bars.Get(quatrocomp[0])
875.                        startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
876.                        endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
877.                        if m<=0:
878.                            if
app.Project.Structure.Bars.Exist(quatrocomp[1]) == -1:
879.                                barraj =
app.Project.Structure.Bars.Get(quatrocomp[1])
880.                                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
881.                                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
882.                                if startnode.Number == startnodej.Number:
883.                                    if startnode.Number not in CCC4:
884.                                        CCC4.append(startnode.Number)
885.                                if startnode.Number == endnodej.Number:
886.                                    if startnode.Number not in CCC4:
887.                                        CCC4.append(startnode.Number)
888.                                if endnode.Number == startnodej.Number:
889.                                    if endnode.Number not in CCC4:
890.                                        CCC4.append(endnode.Number)
891.                                if endnode.Number == endnodej.Number:
892.                                    if endnode.Number not in CCC4:
893.                                        CCC4.append(endnode.Number)
894.                                m=m+1
895.                quatrocomp.Clear()
896.                quatrotr.Clear()
897.        if quatrocomp.Count<quatrotr.Count:
898.            for i,item in enumerate(quatrotr):
899.                if app.Project.Structure.Bars.Exist(quatrotr[0]) == -1:
900.                    barra = app.Project.Structure.Bars.Get(quatrotr[0])
901.                    startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
902.                    endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
```

```
903.                 if app.Project.Structure.Bars.Exist(quatrotr[1]) == -
1:
904.                     barraj =
app.Project.Structure.Bars.Get(quatrotr[1])
905.                     startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
906.                     endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
907.                     if startnode.Number == startnodej.Number:
908.                         if startnode.Number not in CTTeTTT4:
909.                             CTTeTTT4.append(startnode.Number)
910.                     if startnode.Number == endnodej.Number:
911.                         if startnodej.Number not in CTTeTTT4:
912.                             CTTeTTT4.append(startnode.Number)
913.                     if endnode.Number == startnodej.Number:
914.                         if endnode.Number not in CTTeTTT4:
915.                             CTTeTTT4.append(endnode.Number)
916.                     if endnode.Number == endnodej.Number:
917.                         if endnode.Number not in CTTeTTT4:
918.                             CTTeTTT4.append(endnode.Number)
919.         quatrocomp.Clear()
920.         quatrotr.Clear()
921.
922.  cincocomp=[]
923.  cincotr=[]
924.  cincocomp1=[]
925.  cincotr1=[]
926.  CCC5=[]
927.  CCT5=[]
928.  CTTeTTT5=[]
929.  for af,item in enumerate(cinco5):
930.      if cinco5[af][0] in barrascomprimidasavaliaçãonós:
931.          cincocomp.append(item[0])
932.          cincocomp1.append(item[0])
933.      if cinco5[af][0] in barrastracionadasavaliaçãonós:
934.          cincotr.append(item[0])
935.          cincotr1.append(item[0])
936.      if cinco5[af][1] in barrascomprimidasavaliaçãonós:
937.          cincocomp.append(item[1])
938.          cincocomp1.append(item[1])
939.      if cinco5[af][1] in barrastracionadasavaliaçãonós:
940.          cincotr.append(item[1])
941.          cincotr1.append(item[1])
942.      if cinco5[af][2] in barrascomprimidasavaliaçãonós:
943.          cincocomp.append(item[2])
944.          cincocomp1.append(item[2])
945.      if cinco5[af][2] in barrastracionadasavaliaçãonós:
946.          cincotr.append(item[2])
947.          cincotr1.append(item[2])
948.      if cinco5[af][3] in barrascomprimidasavaliaçãonós:
949.          cincocomp.append(item[3])
950.          cincocomp1.append(item[3])
951.      if cinco5[af][3] in barrastracionadasavaliaçãonós:
952.          cincotr.append(item[3])
953.          cincotr1.append(item[3])
954.      if cinco5[af][4] in barrascomprimidasavaliaçãonós:
955.          cincocomp.append(item[4])
956.          cincocomp1.append(item[4])
957.      if cinco5[af][4] in barrastracionadasavaliaçãonós:
958.          cincotr.append(item[4])
```

```
959.           cincotr1.append(item[4])
960.
961.       if cincocomp.Count>=cincotr.Count:
962.           if cincotr.Count==1:
963.               for i,item in enumerate(cincocomp):
964.                   if app.Project.Structure.Bars.Exist(item) == -1:
965.                       barra = app.Project.Structure.Bars.Get(item)
966.                       startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
967.                       endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
968.                       for j,itemj in enumerate(cincotr):
969.                           if app.Project.Structure.Bars.Exist(itemj) ==
-1:
970.                               barraj =
app.Project.Structure.Bars.Get(itemj)
971.                               startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
972.                               endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
973.                               if startnode.Number == startnodej.Number:
974.                                   if startnode.Number not in CCT5:
975.                                       CCT5.append(startnode.Number)
976.                               if startnode.Number == endnodej.Number:
977.                                   if startnode.Number not in CCT5:
978.                                       CCT5.append(startnode.Number)
979.                               if endnode.Number == startnodej.Number:
980.                                   if endnode.Number not in CCT5:
981.                                       CCT5.append(endnode.Number)
982.                               if endnode.Number == endnodej.Number:
983.                                   if endnode.Number not in CCT5:
984.                                       CCT5.append(endnode.Number)
985.               cincocomp.Clear()
986.               cincotr.Clear()
987.           if cincotr.Count>1:
988.               for i,item in enumerate(cincocomp):
989.                   if app.Project.Structure.Bars.Exist(item) == -1:
990.                       barra = app.Project.Structure.Bars.Get(item)
991.                       startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
992.                       endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
993.                       for j,itemj in enumerate(cincotr):
994.                           if app.Project.Structure.Bars.Exist(itemj) ==
-1:
995.                               barraj =
app.Project.Structure.Bars.Get(itemj)
996.                               startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
997.                               endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
998.                               if startnode.Number == startnodej.Number:
999.                                   if startnode.Number not in CTTeTTT5:
1000.                                      CTTeTTT5.append(startnode.Number)
1001.                              if startnode.Number == endnodej.Number:
1002.                                  if startnode.Number not in CTTeTTT5:
1003.                                      CTTeTTT5.append(startnode.Number)
1004.                              if endnode.Number == startnodej.Number:
1005.                                  if endnode.Number not in CTTeTTT5:
1006.                                      CTTeTTT5.append(endnode.Number)
```

```
1007.                               if endnode.Number == endnodej.Number:
1008.                                   if endnode.Number not in CTTeTTT5:
1009.                                       CTTeTTT5.append(endnode.Number)
1010.             cincocomp.Clear()
1011.             cincotr.Clear()
1012.         m=0
1013.         if cincotr.Count==0:
1014.             for i,item in enumerate(cincocomp):
1015.                 if app.Project.Structure.Bars.Exist(cincocomp[0]) ==
-1:
1016.                     barra =
app.Project.Structure.Bars.Get(cincocomp[0])
1017.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1018.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1019.                     if m <= 0:
1020.                         if
app.Project.Structure.Bars.Exist(cincocomp[1]) == -1:
1021.                             barraj =
app.Project.Structure.Bars.Get(cincocomp[1])
1022.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1023.                             endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1024.                             if startnode.Number == startnodej.Number:
1025.                                 if startnode.Number not in CCC5:
1026.                                     CCC5.append(startnode.Number)
1027.                             if startnode.Number == endnodej.Number:
1028.                                 if startnode.Number not in CCC5:
1029.                                     CCC5.append(startnode.Number)
1030.                             if endnode.Number == startnodej.Number:
1031.                                 if endnode.Number not in CCC5:
1032.                                     CCC5.append(endnode.Number)
1033.                             if endnode.Number == endnodej.Number:
1034.                                 if endnode.Number not in CCC5:
1035.                                     CCC5.append(endnode.Number)
1036.                             m=m+1
1037.         cincocomp.Clear()
1038.         cincotr.Clear()
1039.     if cincocomp.Count<cincotr.Count:
1040.         for i,item in enumerate(cincocomp):
1041.             if app.Project.Structure.Bars.Exist(item) == -1:
1042.                 barra = app.Project.Structure.Bars.Get(item)
1043.                 startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1044.                 endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1045.                 for j,itemj in enumerate(cincotr):
1046.                     if app.Project.Structure.Bars.Exist(itemj) == -1:
1047.                         barraj =
app.Project.Structure.Bars.Get(itemj)
1048.                         startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1049.                         endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1050.                         if startnode.Number == startnodej.Number:
1051.                             if startnode.Number not in CTTeTTT5:
1052.                                 CTTeTTT5.append(startnode.Number)
1053.                         if startnode.Number == endnodej.Number:
```

```
                                if startnodej.Number not in CTTeTTT5:
                                    CTTeTTT5.append(startnode.Number)
                            if endnode.Number == startnodej.Number:
                                if endnode.Number not in CTTeTTT5:
                                    CTTeTTT5.append(endnode.Number)
                            if endnode.Number == endnodej.Number:
                                if endnode.Number not in CTTeTTT5:
                                    CTTeTTT5.append(endnode.Number)
        cincocomp.Clear()
        cincotr.Clear()

seiscomp=[]
seistr=[]
seiscomp1=[]
seistr1=[]
CCC6=[]
CCT6=[]
CTTeTTT6=[]
for af,item in enumerate(seis6):
    if seis6[af][0] in barrascomprimidasavaliaçãonós:
        seiscomp.append(item[0])
        seiscomp1.append(item[0])
    if seis6[af][0] in barrastracionadasavaliaçãonós:
        seistr.append(item[0])
        seistr1.append(item[0])
    if seis6[af][1] in barrascomprimidasavaliaçãonós:
        seiscomp.append(item[1])
        seiscomp1.append(item[1])
    if seis6[af][1] in barrastracionadasavaliaçãonós:
        seistr.append(item[1])
        seistr1.append(item[1])
    if seis6[af][2] in barrascomprimidasavaliaçãonós:
        seiscomp.append(item[2])
        seiscomp1.append(item[2])
    if seis6[af][2] in barrastracionadasavaliaçãonós:
        seistr.append(item[2])
        seistr1.append(item[2])
    if seis6[af][3] in barrascomprimidasavaliaçãonós:
        seiscomp.append(item[3])
        seiscomp1.append(item[3])
    if seis6[af][3] in barrastracionadasavaliaçãonós:
        seistr.append(item[3])
        seistr1.append(item[3])
    if seis6[af][4] in barrascomprimidasavaliaçãonós:
        seiscomp.append(item[4])
        seiscomp1.append(item[4])
    if seis6[af][4] in barrastracionadasavaliaçãonós:
        seistr.append(item[4])
        seistr1.append(item[4])
    if seis6[af][5] in barrascomprimidasavaliaçãonós:
        seiscomp.append(item[5])
        seiscomp1.append(item[5])
    if seis6[af][5] in barrastracionadasavaliaçãonós:
        seistr.append(item[5])
        seistr1.append(item[5])

    if seiscomp.Count>=seistr.Count:
        if seistr.Count==1:
            for i,item in enumerate(seiscomp):
                if app.Project.Structure.Bars.Exist(item) == -1:
```

```
1114.                     barra = app.Project.Structure.Bars.Get(item)
1115.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1116.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1117.                     for j,itemj in enumerate(seistr):
1118.                         if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1119.                             barraj =
app.Project.Structure.Bars.Get(itemj)
1120.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1121.                             endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1122.                             if startnode.Number == startnodej.Number:
1123.                                 if startnode.Number not in CCT6:
1124.                                     CCT6.append(startnode.Number)
1125.                             if startnode.Number == endnodej.Number:
1126.                                 if startnode.Number not in CCT6:
1127.                                     CCT6.append(startnode.Number)
1128.                             if endnode.Number == startnodej.Number:
1129.                                 if endnode.Number not in CCT6:
1130.                                     CCT6.append(endnode.Number)
1131.                             if endnode.Number == endnodej.Number:
1132.                                 if endnode.Number not in CCT6:
1133.                                     CCT6.append(endnode.Number)
1134.             seiscomp.Clear()
1135.             seistr.Clear()
1136.         if seistr.Count>1:
1137.             for i,item in enumerate(seiscomp):
1138.                 if app.Project.Structure.Bars.Exist(item) == -1:
1139.                     barra = app.Project.Structure.Bars.Get(item)
1140.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1141.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1142.                     for j,itemj in enumerate(seistr):
1143.                         if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1144.                             barraj =
app.Project.Structure.Bars.Get(itemj)
1145.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1146.                             endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1147.                             if startnode.Number == startnodej.Number:
1148.                                 if startnode.Number not in CTTeTTT6:
1149.                                     CTTeTTT6.append(startnode.Number)
1150.                             if startnode.Number == endnodej.Number:
1151.                                 if startnode.Number not in CTTeTTT6:
1152.                                     CTTeTTT6.append(startnode.Number)
1153.                             if endnode.Number == startnodej.Number:
1154.                                 if endnode.Number not in CTTeTTT6:
1155.                                     CTTeTTT6.append(endnode.Number)
1156.                             if endnode.Number == endnodej.Number:
1157.                                 if endnode.Number not in CTTeTTT6:
1158.                                     CTTeTTT6.append(endnode.Number)
1159.             seiscomp.Clear()
1160.             seistr.Clear()
1161.         m=0
```

```
1162.         if seistr.Count==0:
1163.             for i,item in enumerate(seiscomp):
1164.                 if app.Project.Structure.Bars.Exist(seiscomp[0]) == -
1:
1165.                     barra =
app.Project.Structure.Bars.Get(seiscomp[0])
1166.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1167.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1168.                     if m<=0:
1169.                         if
app.Project.Structure.Bars.Exist(seiscomp[1]) == -1:
1170.                             barraj =
app.Project.Structure.Bars.Get(seiscomp[1])
1171.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1172.                             endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1173.                             if startnode.Number == startnodej.Number:
1174.                                 if startnode.Number not in CCC6:
1175.                                     CCC6.append(startnode.Number)
1176.                             if startnode.Number == endnodej.Number:
1177.                                 if startnode.Number not in CCC6:
1178.                                     CCC6.append(startnode.Number)
1179.                             if endnode.Number == startnodej.Number:
1180.                                 if endnode.Number not in CCC6:
1181.                                     CCC6.append(endnode.Number)
1182.                             if endnode.Number == endnodej.Number:
1183.                                 if endnode.Number not in CCC6:
1184.                                     CCC6.append(endnode.Number)
1185.                             m=m+1
1186.         seiscomp.Clear()
1187.         seistr.Clear()
1188.     if seiscomp.Count<seistr.Count:
1189.         for i,item in enumerate(seiscomp):
1190.             if app.Project.Structure.Bars.Exist(item) == -1:
1191.                 barra = app.Project.Structure.Bars.Get(item)
1192.                 startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1193.                 endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1194.                 for j,itemj in enumerate(seistr):
1195.                     if app.Project.Structure.Bars.Exist(itemj) == -1:
1196.                         barraj =
app.Project.Structure.Bars.Get(itemj)
1197.                         startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1198.                         endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1199.                         if startnode.Number == startnodej.Number:
1200.                             if startnode.Number not in CTTeTTT6:
1201.                                 CTTeTTT6.append(startnode.Number)
1202.                         if startnode.Number == endnodej.Number:
1203.                             if startnodej.Number not in CTTeTTT6:
1204.                                 CTTeTTT6.append(startnode.Number)
1205.                         if endnode.Number == startnodej.Number:
1206.                             if endnode.Number not in CTTeTTT6:
1207.                                 CTTeTTT6.append(endnode.Number)
1208.                         if endnode.Number == endnodej.Number:
```

```
1209.                               if endnode.Number not in CTTeTTT6:
1210.                                   CTTeTTT6.append(endnode.Number)
1211.         seiscomp.Clear()
1212.         seistr.Clear()
1213.
1214.
1215. #    FAZER PARA OS APOIOS
1216. apoio333comp=[]
1217. apoio333tr=[]
1218. apoio333comp1=[]
1219. apoio333tr1=[]
1220. CCTapoio2=[]
1221. CCCapoio2=[]
1222. CTTeTTTapoio2=[]
1223. for af,item in enumerate(apoiofinal222):
1224.     if apoiofinal222[af][0] in barrascomprimidasavaliaçãonós:
1225.         apoio333comp.append(item[0])
1226.         apoio333comp1.append(item[0])
1227.     if apoiofinal222[af][0] in barrastracionadasavaliaçãonós:
1228.         apoio333tr.append(item[0])
1229.         apoio333tr1.append(item[0])
1230.     if apoiofinal222[af][1] in barrascomprimidasavaliaçãonós:
1231.         apoio333comp.append(item[1])
1232.         apoio333comp1.append(item[1])
1233.     if apoiofinal222[af][1] in barrastracionadasavaliaçãonós:
1234.         apoio333tr.append(item[1])
1235.         apoio333tr1.append(item[1])
1236.
1237.     if apoio333comp.Count>apoio333tr.Count:
1238.         if apoio333tr.Count==1:
1239.             for i,item in enumerate(apoio333comp):
1240.                 if app.Project.Structure.Bars.Exist(item) == -1:
1241.                     barra = app.Project.Structure.Bars.Get(item)
1242.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1243.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1244.                     for j,itemj in enumerate(apoio333tr):
1245.                         if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1246.                             barraj =
app.Project.Structure.Bars.Get(itemj)
1247.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1248.                             endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1249.                             if startnode.Number == startnodej.Number:
1250.                                 if startnode.Number not in CCTapoio2:
1251.
CCTapoio2.append(startnode.Number)
1252.                             if startnode.Number == endnodej.Number:
1253.                                 if startnode.Number not in CCTapoio2:
1254.
CCTapoio2.append(startnode.Number)
1255.                             if endnode.Number == startnodej.Number:
1256.                                 if endnode.Number not in CCTapoio2:
1257.                                     CCTapoio2.append(endnode.Number)
1258.                             if endnode.Number == endnodej.Number:
1259.                                 if endnode.Number not in CCTapoio2:
1260.                                     CCTapoio2.append(endnode.Number)
```

```
1261.             apoio333comp.Clear()
1262.             apoio333tr.Clear()
1263.         if apoio333tr.Count>1:
1264.             for i,item in enumerate(apoio333comp):
1265.                 if app.Project.Structure.Bars.Exist(item) == -1:
1266.                     barra = app.Project.Structure.Bars.Get(item)
1267.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1268.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1269.                     for j,itemj in enumerate(apoio333tr):
1270.                         if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1271.                             barraj =
app.Project.Structure.Bars.Get(itemj)
1272.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1273.                             endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1274.                             if startnode.Number == startnodej.Number:
1275.                                 if startnode.Number not in
CTTeTTTapoio2:
1276.
CTTeTTTapoio2.append(startnode.Number)
1277.                             if startnode.Number == endnodej.Number:
1278.                                 if startnode.Number not in
CTTeTTTapoio2:
1279.
CTTeTTTapoio2.append(startnode.Number)
1280.                             if endnode.Number == startnodej.Number:
1281.                                 if endnode.Number not in
CTTeTTTapoio2:
1282.
CTTeTTTapoio2.append(endnode.Number)
1283.                             if endnode.Number == endnodej.Number:
1284.                                 if endnode.Number not in
CTTeTTTapoio2:
1285.
CTTeTTTapoio2.append(endnode.Number)
1286.             apoio333comp.Clear()
1287.             apoio333tr.Clear()
1288.         m=0
1289.         if apoio333tr.Count==0:
1290.             for i,item in enumerate(apoio333comp):
1291.                 if app.Project.Structure.Bars.Exist(apoio333comp[0])
== -1:
1292.                     barra =
app.Project.Structure.Bars.Get(apoio333comp[0])
1293.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1294.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1295.                     if m<=0:
1296.                         if
app.Project.Structure.Bars.Exist(apoio333comp[1]) == -1:
1297.                             barraj =
app.Project.Structure.Bars.Get(apoio333comp[1])
1298.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
```

```
1299.                                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1300.                                if startnode.Number == startnodej.Number:
1301.                                    if startnode.Number not in CCCapoio2:
1302.
CCCapoio2.append(startnode.Number)
1303.                                if startnode.Number == endnodej.Number:
1304.                                    if startnode.Number not in CCCapoio2:
1305.
CCCapoio2.append(startnode.Number)
1306.                                if endnode.Number == startnodej.Number:
1307.                                    if endnode.Number not in CCCapoio2:
1308.                                        CCCapoio2.append(endnode.Number)
1309.                                if endnode.Number == endnodej.Number:
1310.                                    if endnode.Number not in CCCapoio2:
1311.                                        CCCapoio2.append(endnode.Number)
1312.                                m=m+1
1313.          apoio333comp.Clear()
1314.          apoio333tr.Clear()
1315.      if apoio333comp.Count<=apoio333tr.Count:
1316.          for i,item in enumerate(apoio333comp):
1317.              if app.Project.Structure.Bars.Exist(item) == -1:
1318.                  barra = app.Project.Structure.Bars.Get(item)
1319.                  startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1320.                  endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1321.                  for j,itemj in enumerate(apoio333tr):
1322.                      if app.Project.Structure.Bars.Exist(itemj) == -1:
1323.                          barraj =
app.Project.Structure.Bars.Get(itemj)
1324.                          startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1325.                          endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1326.                          if startnode.Number == startnodej.Number:
1327.                              if startnode.Number not in CTTeTTTapoio2:
1328.
CTTeTTTapoio2.append(startnode.Number)
1329.                          if startnode.Number == endnodej.Number:
1330.                              if startnodej.Number not in
CTTeTTTapoio2:
1331.
CTTeTTTapoio2.append(startnode.Number)
1332.                          if endnode.Number == startnodej.Number:
1333.                              if endnode.Number not in CTTeTTTapoio2:
1334.                                  CTTeTTTapoio2.append(endnode.Number)
1335.                          if endnode.Number == endnodej.Number:
1336.                              if endnode.Number not in CTTeTTTapoio2:
1337.                                  CTTeTTTapoio2.append(endnode.Number)
1338.          apoio333comp.Clear()
1339.          apoio333tr.Clear()
1340.
1341. apoio444comp=[]
1342. apoio444tr=[]
1343. apoio444comp1=[]
1344. apoio444tr1=[]
1345. CCTapoio3=[]
1346. CCCapoio3=[]
1347. CTTeTTTapoio3=[]
```

```
1348. for af,item in enumerate(apoiofinal333):
1349.     if apoiofinal333[af][0] in barrascomprimidasavaliaçãonós:
1350.         apoio444comp.append(item[0])
1351.         apoio444comp1.append(item[0])
1352.     if apoiofinal333[af][0] in barrastracionadasavaliaçãonós:
1353.         apoio444tr.append(item[0])
1354.         apoio444tr1.append(item[0])
1355.     if apoiofinal333[af][1] in barrascomprimidasavaliaçãonós:
1356.         apoio444comp.append(item[1])
1357.         apoio444comp1.append(item[1])
1358.     if apoiofinal333[af][1] in barrastracionadasavaliaçãonós:
1359.         apoio444tr.append(item[1])
1360.         apoio444tr1.append(item[1])
1361.     if apoiofinal333[af][2] in barrascomprimidasavaliaçãonós:
1362.         apoio444comp.append(item[2])
1363.         apoio444comp1.append(item[2])
1364.     if apoiofinal333[af][2] in barrastracionadasavaliaçãonós:
1365.         apoio444tr.append(item[2])
1366.         apoio444tr1.append(item[2])
1367.
1368.     if apoio444comp.Count>apoio444tr.Count:
1369.         if apoio444tr.Count==1:
1370.             for i,item in enumerate(apoio444comp):
1371.                 if app.Project.Structure.Bars.Exist(item) == -1:
1372.                     barra = app.Project.Structure.Bars.Get(item)
1373.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1374.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1375.                     for j,itemj in enumerate(apoio444tr):
1376.                         if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1377.                             barraj =
app.Project.Structure.Bars.Get(itemj)
1378.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1379.                             endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1380.                             if startnode.Number == startnodej.Number:
1381.                                 if startnode.Number not in CCTapoio3:
1382.
CCTapoio3.append(startnode.Number)
1383.                             if startnode.Number == endnodej.Number:
1384.                                 if startnode.Number not in CCTapoio3:
1385.
CCTapoio3.append(startnode.Number)
1386.                             if endnode.Number == startnodej.Number:
1387.                                 if endnode.Number not in CCTapoio3:
1388.                                     CCTapoio3.append(endnode.Number)
1389.                             if endnode.Number == endnodej.Number:
1390.                                 if endnode.Number not in CCTapoio3:
1391.                                     CCTapoio3.append(endnode.Number)
1392.             apoio444comp.Clear()
1393.             apoio444tr.Clear()
1394.         if apoio444tr.Count>1:
1395.             for i,item in enumerate(apoio444comp):
1396.                 if app.Project.Structure.Bars.Exist(item) == -1:
1397.                     barra = app.Project.Structure.Bars.Get(item)
1398.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
```

```
1399.                       endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1400.                       for j,itemj in enumerate(apoio444tr):
1401.                           if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1402.                               barraj =
app.Project.Structure.Bars.Get(itemj)
1403.                               startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1404.                               endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1405.                               if startnode.Number == startnodej.Number:
1406.                                   if startnode.Number not in
CTTeTTTapoio3:
1407.
CTTeTTTapoio3.append(startnode.Number)
1408.                               if startnode.Number == endnodej.Number:
1409.                                   if startnode.Number not in
CTTeTTTapoio3:
1410.
CTTeTTTapoio3.append(startnode.Number)
1411.                               if endnode.Number == startnodej.Number:
1412.                                   if endnode.Number not in
CTTeTTTapoio3:
1413.
CTTeTTTapoio3.append(endnode.Number)
1414.                               if endnode.Number == endnodej.Number:
1415.                                   if endnode.Number not in
CTTeTTTapoio3:
1416.
CTTeTTTapoio3.append(endnode.Number)
1417.               apoio444comp.Clear()
1418.               apoio444tr.Clear()
1419.           m=0
1420.           if apoio444tr.Count==0:
1421.               for i,item in enumerate(apoio444comp):
1422.                   if app.Project.Structure.Bars.Exist(apoio444comp[0])
== -1:
1423.                       barra =
app.Project.Structure.Bars.Get(apoio444comp[0])
1424.                       startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1425.                       endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1426.                       if m<=0:
1427.                           if
app.Project.Structure.Bars.Exist(apoio444comp[1]) == -1:
1428.                               barraj =
app.Project.Structure.Bars.Get(apoio444comp[1])
1429.                               startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1430.                               endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1431.                               if startnode.Number == startnodej.Number:
1432.                                   if startnode.Number not in CCCapoio3:
1433.
CCCapoio3.append(startnode.Number)
1434.                               if startnode.Number == endnodej.Number:
1435.                                   if startnode.Number not in CCCapoio3:
```

```
1436.
CCCapoio3.append(startnode.Number)
1437.                                 if endnode.Number == startnodej.Number:
1438.                                     if endnode.Number not in CCCapoio3:
1439.                                         CCCapoio3.append(endnode.Number)
1440.                                 if endnode.Number == endnodej.Number:
1441.                                     if endnode.Number not in CCCapoio3:
1442.                                         CCCapoio3.append(endnode.Number)
1443.                                 m=m+1
1444.         apoio444comp.Clear()
1445.         apoio444tr.Clear()
1446.     if apoio444comp.Count<=apoio444tr.Count:
1447.         for i,item in enumerate(apoio444comp):
1448.             if app.Project.Structure.Bars.Exist(item) == -1:
1449.                 barra = app.Project.Structure.Bars.Get(item)
1450.                 startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1451.                 endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1452.                 for j,itemj in enumerate(apoio444tr):
1453.                     if app.Project.Structure.Bars.Exist(itemj) == -1:
1454.                         barraj =
app.Project.Structure.Bars.Get(itemj)
1455.                         startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1456.                         endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1457.                         if startnode.Number == startnodej.Number:
1458.                             if startnode.Number not in CTTeTTTapoio3:
1459.
CTTeTTTapoio3.append(startnode.Number)
1460.                         if startnode.Number == endnodej.Number:
1461.                             if startnodej.Number not in
CTTeTTTapoio3:
1462.
CTTeTTTapoio3.append(startnode.Number)
1463.                         if endnode.Number == startnodej.Number:
1464.                             if endnode.Number not in CTTeTTTapoio3:
1465.                                 CTTeTTTapoio3.append(endnode.Number)
1466.                         if endnode.Number == endnodej.Number:
1467.                             if endnode.Number not in CTTeTTTapoio3:
1468.                                 CTTeTTTapoio3.append(endnode.Number)
1469.         apoio444comp.Clear()
1470.         apoio444tr.Clear()
1471.
1472. apoio555comp=[]
1473. apoio555tr=[]
1474. apoio555comp1=[]
1475. apoio555tr1=[]
1476. CCTapoio4=[]
1477. CCCapoio4=[]
1478. CTTeTTTapoio4=[]
1479. for af,item in enumerate(apoiofinal444):
1480.     if apoiofinal444[af][0] in barrascomprimidasavaliaçãonós:
1481.         apoio555comp.append(item[0])
1482.         apoio555comp1.append(item[0])
1483.     if apoiofinal444[af][0] in barrastracionadasavaliaçãonós:
1484.         apoio555tr.append(item[0])
1485.         apoio555tr1.append(item[0])
1486.     if apoiofinal444[af][1] in barrascomprimidasavaliaçãonós:
```

```
1487.          apoio555comp.append(item[1])
1488.          apoio555comp1.append(item[1])
1489.      if apoiofinal444[af][1] in barrastracionadasavaliaçãonós:
1490.          apoio555tr.append(item[1])
1491.          apoio555tr1.append(item[1])
1492.      if apoiofinal444[af][2] in barrascomprimidasavaliaçãonós:
1493.          apoio555comp.append(item[2])
1494.          apoio555comp1.append(item[2])
1495.      if apoiofinal444[af][2] in barrastracionadasavaliaçãonós:
1496.          apoio555tr.append(item[2])
1497.          apoio555tr1.append(item[2])
1498.      if apoiofinal444[af][3] in barrascomprimidasavaliaçãonós:
1499.          apoio555comp.append(item[3])
1500.          apoio555comp1.append(item[3])
1501.      if apoiofinal444[af][3] in barrastracionadasavaliaçãonós:
1502.          apoio555tr.append(item[3])
1503.          apoio555tr1.append(item[3])
1504.
1505.      if apoio555comp.Count>apoio555tr.Count:
1506.          if apoio555tr.Count==1:
1507.              for i,item in enumerate(apoio555comp):
1508.                  if app.Project.Structure.Bars.Exist(item) == -1:
1509.                      barra = app.Project.Structure.Bars.Get(item)
1510.                      startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1511.                      endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1512.                      for j,itemj in enumerate(apoio555tr):
1513.                          if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1514.                              barraj =
app.Project.Structure.Bars.Get(itemj)
1515.                              startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1516.                              endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1517.                              if startnode.Number == startnodej.Number:
1518.                                  if startnode.Number not in CCTapoio4:
1519.
CCTapoio4.append(startnode.Number)
1520.                              if startnode.Number == endnodej.Number:
1521.                                  if startnode.Number not in CCTapoio4:
1522.
CCTapoio4.append(startnode.Number)
1523.                              if endnode.Number == startnodej.Number:
1524.                                  if endnode.Number not in CCTapoio4:
1525.                                      CCTapoio4.append(endnode.Number)
1526.                              if endnode.Number == endnodej.Number:
1527.                                  if endnode.Number not in CCTapoio4:
1528.                                      CCTapoio4.append(endnode.Number)
1529.              apoio555comp.Clear()
1530.              apoio555tr.Clear()
1531.          if apoio555tr.Count>1:
1532.              for i,item in enumerate(apoio555comp):
1533.                  if app.Project.Structure.Bars.Exist(item) == -1:
1534.                      barra = app.Project.Structure.Bars.Get(item)
1535.                      startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1536.                      endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
```

```
1537.                     for j,itemj in enumerate(apoio555tr):
1538.                         if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1539.                             barraj =
app.Project.Structure.Bars.Get(itemj)
1540.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1541.                             endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1542.                             if startnode.Number == startnodej.Number:
1543.                                 if startnode.Number not in
CTTeTTTapoio4:
1544.
CTTeTTTapoio4.append(startnode.Number)
1545.                             if startnode.Number == endnodej.Number:
1546.                                 if startnode.Number not in
CTTeTTTapoio4:
1547.
CTTeTTTapoio4.append(startnode.Number)
1548.                             if endnode.Number == startnodej.Number:
1549.                                 if endnode.Number not in
CTTeTTTapoio4:
1550.
CTTeTTTapoio4.append(endnode.Number)
1551.                             if endnode.Number == endnodej.Number:
1552.                                 if endnode.Number not in
CTTeTTTapoio4:
1553.
CTTeTTTapoio4.append(endnode.Number)
1554.             apoio555comp.Clear()
1555.             apoio555tr.Clear()
1556.         m=0
1557.         if apoio555tr.Count==0:
1558.             for i,item in enumerate(apoio555comp):
1559.                 if app.Project.Structure.Bars.Exist(apoio555comp[0])
== -1:
1560.                     barra =
app.Project.Structure.Bars.Get(apoio555comp[0])
1561.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1562.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1563.                     if m<=0:
1564.                         if
app.Project.Structure.Bars.Exist(apoio555comp[1]) == -1:
1565.                             barraj =
app.Project.Structure.Bars.Get(apoio555comp[1])
1566.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1567.                             endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1568.                             if startnode.Number == startnodej.Number:
1569.                                 if startnode.Number not in CCCapoio4:
1570.
CCCapoio4.append(startnode.Number)
1571.                             if startnode.Number == endnodej.Number:
1572.                                 if startnode.Number not in CCCapoio4:
1573.
CCCapoio4.append(startnode.Number)
1574.                             if endnode.Number == startnodej.Number:
```

```
1575.                                   if endnode.Number not in CCCapoio4:
1576.                                       CCCapoio4.append(endnode.Number)
1577.                               if endnode.Number == endnodej.Number:
1578.                                   if endnode.Number not in CCCapoio4:
1579.                                       CCCapoio4.append(endnode.Number)
1580.                               m=m+1
1581.         apoio555comp.Clear()
1582.         apoio555tr.Clear()
1583.     if apoio555comp.Count<=apoio555tr.Count:
1584.         for i,item in enumerate(apoio555comp):
1585.             if app.Project.Structure.Bars.Exist(item) == -1:
1586.                 barra = app.Project.Structure.Bars.Get(item)
1587.                 startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1588.                 endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1589.                 for j,itemj in enumerate(apoio555tr):
1590.                     if app.Project.Structure.Bars.Exist(itemj) == -1:
1591.                         barraj =
app.Project.Structure.Bars.Get(itemj)
1592.                         startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1593.                         endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1594.                         if startnode.Number == startnodej.Number:
1595.                             if startnode.Number not in CTTeTTTapoio4:
1596.
CTTeTTTapoio4.append(startnode.Number)
1597.                         if startnode.Number == endnodej.Number:
1598.                             if startnodej.Number not in
CTTeTTTapoio4:
1599.
CTTeTTTapoio4.append(startnode.Number)
1600.                         if endnode.Number == startnodej.Number:
1601.                             if endnode.Number not in CTTeTTTapoio4:
1602.                                 CTTeTTTapoio4.append(endnode.Number)
1603.                         if endnode.Number == endnodej.Number:
1604.                             if endnode.Number not in CTTeTTTapoio4:
1605.                                 CTTeTTTapoio4.append(endnode.Number)
1606.         apoio555comp.Clear()
1607.         apoio555tr.Clear()
1608.
1609. apoio666comp=[]
1610. apoio666tr=[]
1611. apoio666comp1=[]
1612. apoio666tr1=[]
1613. CCTapoio5=[]
1614. CCCapoio5=[]
1615. CTTeTTTapoio5=[]
1616. for af,item in enumerate(apoiofinal555):
1617.     if apoiofinal555[af][0] in barrascomprimidasavaliaçãonós:
1618.         apoio666comp.append(item[0])
1619.         apoio666comp1.append(item[0])
1620.     if apoiofinal555[af][0] in barrastracionadasavaliaçãonós:
1621.         apoio666tr.append(item[0])
1622.         apoio666tr1.append(item[0])
1623.     if apoiofinal555[af][1] in barrascomprimidasavaliaçãonós:
1624.         apoio666comp.append(item[1])
1625.         apoio666comp1.append(item[1])
1626.     if apoiofinal555[af][1] in barrastracionadasavaliaçãonós:
```

```
1627.          apoio666tr.append(item[1])
1628.          apoio666tr1.append(item[1])
1629.      if apoiofinal555[af][2] in barrascomprimidasavaliaçãonós:
1630.          apoio666comp.append(item[2])
1631.          apoio666comp1.append(item[2])
1632.      if apoiofinal555[af][2] in barrastracionadasavaliaçãonós:
1633.          apoio666tr.append(item[2])
1634.          apoio666tr1.append(item[2])
1635.      if apoiofinal555[af][3] in barrascomprimidasavaliaçãonós:
1636.          apoio666comp.append(item[3])
1637.          apoio666comp1.append(item[3])
1638.      if apoiofinal555[af][3] in barrastracionadasavaliaçãonós:
1639.          apoio666tr.append(item[3])
1640.          apoio666tr1.append(item[3])
1641.      if apoiofinal555[af][4] in barrascomprimidasavaliaçãonós:
1642.          apoio666comp.append(item[4])
1643.          apoio666comp1.append(item[4])
1644.      if apoiofinal555[af][4] in barrastracionadasavaliaçãonós:
1645.          apoio666tr.append(item[4])
1646.          apoio666tr1.append(item[4])
1647.
1648.      if apoio666comp.Count>apoio666tr.Count:
1649.          if apoio666tr.Count==1:
1650.              for i,item in enumerate(apoio666comp):
1651.                  if app.Project.Structure.Bars.Exist(item) == -1:
1652.                      barra = app.Project.Structure.Bars.Get(item)
1653.                      startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1654.                      endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1655.                      for j,itemj in enumerate(apoio666tr):
1656.                          if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1657.                              barraj =
app.Project.Structure.Bars.Get(itemj)
1658.                              startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1659.                              endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1660.                              if startnode.Number == startnodej.Number:
1661.                                  if startnode.Number not in CCTapoio5:
1662.
CCTapoio5.append(startnode.Number)
1663.                              if startnode.Number == endnodej.Number:
1664.                                  if startnode.Number not in CCTapoio5:
1665.
CCTapoio5.append(startnode.Number)
1666.                              if endnode.Number == startnodej.Number:
1667.                                  if endnode.Number not in CCTapoio5:
1668.                                      CCTapoio5.append(endnode.Number)
1669.                              if endnode.Number == endnodej.Number:
1670.                                  if endnode.Number not in CCTapoio5:
1671.                                      CCTapoio5.append(endnode.Number)
1672.              apoio666comp.Clear()
1673.              apoio666tr.Clear()
1674.          if apoio666tr.Count>1:
1675.              for i,item in enumerate(apoio666comp):
1676.                  if app.Project.Structure.Bars.Exist(item) == -1:
1677.                      barra = app.Project.Structure.Bars.Get(item)
```

```
1678.                       startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1679.                       endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1680.                       for j,itemj in enumerate(apoio666tr):
1681.                           if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1682.                               barraj =
app.Project.Structure.Bars.Get(itemj)
1683.                               startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1684.                               endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1685.                               if startnode.Number == startnodej.Number:
1686.                                   if startnode.Number not in
CTTeTTTapoio5:
1687.
CTTeTTTapoio5.append(startnode.Number)
1688.                               if startnode.Number == endnodej.Number:
1689.                                   if startnode.Number not in
CTTeTTTapoio5:
1690.
CTTeTTTapoio5.append(startnode.Number)
1691.                               if endnode.Number == startnodej.Number:
1692.                                   if endnode.Number not in
CTTeTTTapoio5:
1693.
CTTeTTTapoio5.append(endnode.Number)
1694.                               if endnode.Number == endnodej.Number:
1695.                                   if endnode.Number not in
CTTeTTTapoio5:
1696.
CTTeTTTapoio5.append(endnode.Number)
1697.             apoio666comp.Clear()
1698.             apoio666tr.Clear()
1699.
1700.         m=0
1701.         if apoio666tr.Count==0:
1702.             for i,item in enumerate(apoio666comp):
1703.                 if app.Project.Structure.Bars.Exist(apoio666comp[0])
== -1:
1704.                     barra =
app.Project.Structure.Bars.Get(apoio666comp[0])
1705.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1706.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1707.                     if m<=0:
1708.                         if
app.Project.Structure.Bars.Exist(apoio666comp[1]) == -1:
1709.                             barraj =
app.Project.Structure.Bars.Get(apoio666comp[1])
1710.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1711.                             endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1712.                             if startnode.Number == startnodej.Number:
1713.                                 if startnode.Number not in CCCapoio5:
1714.
CCCapoio5.append(startnode.Number)
```

```
1715.                                 if startnode.Number == endnodej.Number:
1716.                                     if startnode.Number not in CCCapoio5:
1717.
CCCapoio5.append(startnode.Number)
1718.                                 if endnode.Number == startnodej.Number:
1719.                                     if endnode.Number not in CCCapoio5:
1720.                                         CCCapoio5.append(endnode.Number)
1721.                                 if endnode.Number == endnodej.Number:
1722.                                     if endnode.Number not in CCCapoio5:
1723.                                         CCCapoio5.append(endnode.Number)
1724.                                 m=m+1
1725.         apoio666comp.Clear()
1726.         apoio666tr.Clear()
1727.     if apoio666comp.Count<=apoio666tr.Count:
1728.         for i,item in enumerate(apoio666comp):
1729.             if app.Project.Structure.Bars.Exist(item) == -1:
1730.                 barra = app.Project.Structure.Bars.Get(item)
1731.                 startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1732.                 endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1733.                 for j,itemj in enumerate(apoio666tr):
1734.                     if app.Project.Structure.Bars.Exist(itemj) == -1:
1735.                         barraj =
app.Project.Structure.Bars.Get(itemj)
1736.                         startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1737.                         endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1738.                         if startnode.Number == startnodej.Number:
1739.                             if startnode.Number not in CTTeTTTapoio5:
1740.
CTTeTTTapoio5.append(startnode.Number)
1741.                         if startnode.Number == endnodej.Number:
1742.                             if startnodej.Number not in
CTTeTTTapoio5:
1743.
CTTeTTTapoio5.append(startnode.Number)
1744.                         if endnode.Number == startnodej.Number:
1745.                             if endnode.Number not in CTTeTTTapoio5:
1746.                                 CTTeTTTapoio5.append(endnode.Number)
1747.                         if endnode.Number == endnodej.Number:
1748.                             if endnode.Number not in CTTeTTTapoio5:
1749.                                 CTTeTTTapoio5.append(endnode.Number)
1750.         apoio666comp.Clear()
1751.         apoio666tr.Clear()
1752.
1753. apoio6666comp=[]
1754. apoio6666tr=[]
1755. apoio6666comp1=[]
1756. apoio6666tr1=[]
1757. CCTapoio6=[]
1758. CCCapoio6=[]
1759. CTTeTTTapoio6=[]
1760. for af,item in enumerate(apoiofinal666):
1761.     if apoiofinal666[af][0] in barrascomprimidasavaliaçãonós:
1762.         apoio6666comp.append(item[0])
1763.         apoio6666comp1.append(item[0])
1764.     if apoiofinal666[af][0] in barrastracionadasavaliaçãonós:
1765.         apoio6666tr.append(item[0])
```

```
1766.          apoio6666tr1.append(item[0])
1767.      if apoiofinal666[af][1] in barrascomprimidasavaliaçãonós:
1768.          apoio6666comp.append(item[1])
1769.          apoio6666comp1.append(item[1])
1770.      if apoiofinal666[af][1] in barrastracionadasavaliaçãonós:
1771.          apoio6666tr.append(item[1])
1772.          apoio6666tr1.append(item[1])
1773.      if apoiofinal666[af][2] in barrascomprimidasavaliaçãonós:
1774.          apoio6666comp.append(item[2])
1775.          apoio6666comp1.append(item[2])
1776.      if apoiofinal666[af][2] in barrastracionadasavaliaçãonós:
1777.          apoio6666tr.append(item[2])
1778.          apoio6666tr1.append(item[2])
1779.      if apoiofinal666[af][3] in barrascomprimidasavaliaçãonós:
1780.          apoio6666comp.append(item[3])
1781.          apoio6666comp1.append(item[3])
1782.      if apoiofinal666[af][3] in barrastracionadasavaliaçãonós:
1783.          apoio6666tr.append(item[3])
1784.          apoio6666tr1.append(item[3])
1785.      if apoiofinal666[af][4] in barrascomprimidasavaliaçãonós:
1786.          apoio6666comp.append(item[4])
1787.          apoio6666comp1.append(item[4])
1788.      if apoiofinal666[af][4] in barrastracionadasavaliaçãonós:
1789.          apoio6666tr.append(item[4])
1790.          apoio6666tr1.append(item[4])
1791.      if apoiofinal666[af][5] in barrascomprimidasavaliaçãonós:
1792.          apoio6666comp.append(item[5])
1793.          apoio6666comp1.append(item[5])
1794.      if apoiofinal666[af][5] in barrastracionadasavaliaçãonós:
1795.          apoio6666tr.append(item[5])
1796.          apoio6666tr1.append(item[5])
1797.
1798.      if apoio6666comp.Count>apoio6666tr.Count:
1799.          if apoio6666tr.Count==1:
1800.              for i,item in enumerate(apoio6666comp):
1801.                  if app.Project.Structure.Bars.Exist(item) == -1:
1802.                      barra = app.Project.Structure.Bars.Get(item)
1803.                      startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1804.                      endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1805.                      for j,itemj in enumerate(apoio6666tr):
1806.                          if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1807.                              barraj =
app.Project.Structure.Bars.Get(itemj)
1808.                              startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1809.                              endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1810.                              if startnode.Number == startnodej.Number:
1811.                                  if startnode.Number not in CCTapoio6:
1812.
CCTapoio6.append(startnode.Number)
1813.                              if startnode.Number == endnodej.Number:
1814.                                  if startnode.Number not in CCTapoio6:
1815.
CCTapoio6.append(startnode.Number)
1816.                              if endnode.Number == startnodej.Number:
1817.                                  if endnode.Number not in CCTapoio6:
```

```
1818.                                       CCTapoio6.append(endnode.Number)
1819.                                   if endnode.Number == endnodej.Number:
1820.                                       if endnode.Number not in CCTapoio6:
1821.                                       CCTapoio6.append(endnode.Number)
1822.             apoio6666comp.Clear()
1823.             apoio6666tr.Clear()
1824.         if apoio6666tr.Count>1:
1825.             for i,item in enumerate(apoio6666comp):
1826.                 if app.Project.Structure.Bars.Exist(item) == -1:
1827.                     barra = app.Project.Structure.Bars.Get(item)
1828.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1829.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1830.                     for j,itemj in enumerate(apoio6666tr):
1831.                         if app.Project.Structure.Bars.Exist(itemj) ==
-1:
1832.                             barraj =
app.Project.Structure.Bars.Get(itemj)
1833.                             startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1834.                             endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1835.                             if startnode.Number == startnodej.Number:
1836.                                 if startnode.Number not in
CTTeTTTapoio6:
1837.
CTTeTTTapoio6.append(startnode.Number)
1838.                             if startnode.Number == endnodej.Number:
1839.                                 if startnode.Number not in
CTTeTTTapoio6:
1840.
CTTeTTTapoio6.append(startnode.Number)
1841.                             if endnode.Number == startnodej.Number:
1842.                                 if endnode.Number not in
CTTeTTTapoio6:
1843.
CTTeTTTapoio6.append(endnode.Number)
1844.                             if endnode.Number == endnodej.Number:
1845.                                 if endnode.Number not in
CTTeTTTapoio6:
1846.
CTTeTTTapoio6.append(endnode.Number)
1847.             apoio6666comp.Clear()
1848.             apoio6666tr.Clear()
1849.         m=0
1850.         if apoio6666tr.Count==0:
1851.             for i,item in enumerate(apoio6666comp):
1852.                 if app.Project.Structure.Bars.Exist(apoio6666comp[0])
== -1:
1853.                     barra =
app.Project.Structure.Bars.Get(apoio6666comp[0])
1854.                     startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1855.                     endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1856.                     if m<=0:
1857.                         if
app.Project.Structure.Bars.Exist(apoio6666comp[1]) == -1:
```

```
1858.                                barraj =
app.Project.Structure.Bars.Get(apoio6666comp[1])
1859.                                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1860.                                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1861.                                if startnode.Number == startnodej.Number:
1862.                                    if startnode.Number not in CCCapoio6:
1863.
CCCapoio6.append(startnode.Number)
1864.                                if startnode.Number == endnodej.Number:
1865.                                    if startnode.Number not in CCCapoio6:
1866.
CCCapoio6.append(startnode.Number)
1867.                                if endnode.Number == startnodej.Number:
1868.                                    if endnode.Number not in CCCapoio6:
1869.                                        CCCapoio6.append(endnode.Number)
1870.                                if endnode.Number == endnodej.Number:
1871.                                    if endnode.Number not in CCCapoio6:
1872.                                        CCCapoio6.append(endnode.Number)
1873.                                m=m+1
1874.          apoio6666comp.Clear()
1875.          apoio6666tr.Clear()
1876.      if apoio6666comp.Count<=apoio6666tr.Count:
1877.          for i,item in enumerate(apoio6666comp):
1878.              if app.Project.Structure.Bars.Exist(item) == -1:
1879.                  barra = app.Project.Structure.Bars.Get(item)
1880.                  startnode =
app.Project.Structure.Nodes.Get(barra.StartNode)
1881.                  endnode =
app.Project.Structure.Nodes.Get(barra.EndNode)
1882.                  for j,itemj in enumerate(apoio6666tr):
1883.                      if app.Project.Structure.Bars.Exist(itemj) == -1:
1884.                          barraj =
app.Project.Structure.Bars.Get(itemj)
1885.                          startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
1886.                          endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
1887.                          if startnode.Number == startnodej.Number:
1888.                              if startnode.Number not in CTTeTTTapoio6:
1889.
CTTeTTTapoio6.append(startnode.Number)
1890.                          if startnode.Number == endnodej.Number:
1891.                              if startnodej.Number not in
CTTeTTTapoio6:
1892.
CTTeTTTapoio6.append(startnode.Number)
1893.                          if endnode.Number == startnodej.Number:
1894.                              if endnode.Number not in CTTeTTTapoio6:
1895.                                  CTTeTTTapoio6.append(endnode.Number)
1896.                          if endnode.Number == endnodej.Number:
1897.                              if endnode.Number not in CTTeTTTapoio6:
1898.                                  CTTeTTTapoio6.append(endnode.Number)
1899.          apoio6666comp.Clear()
1900.          apoio6666tr.Clear()
1901.
1902. nósCCC2=[]
1903. for i,item in enumerate(CCC2):
1904.     if item not in nósCCC2:
```

```
1905.             nósCCC2.append(item)
1906.     nósCCT2=[]
1907.     for i,item in enumerate(CCT2):
1908.         if item not in nósCCT2:
1909.             nósCCT2.append(item)
1910.     nósCTTeTTT2=[]
1911.     for i,item in enumerate(CTTeTTT2):
1912.         if item not in nósCTTeTTT2:
1913.             nósCTTeTTT2.append(item)
1914.     nósCCC3=[]
1915.     for i,item in enumerate(CCC3):
1916.         if item not in nósCCC3:
1917.             nósCCC3.append(item)
1918.     nósCCT3=[]
1919.     for i,item in enumerate(CCT3):
1920.         if item not in nósCCT3:
1921.             nósCCT3.append(item)
1922.     nósCTTeTTT3=[]
1923.     for i,item in enumerate(CTTeTTT3):
1924.         if item not in nósCTTeTTT3:
1925.             nósCTTeTTT3.append(item)
1926.     nósCCC4=[]
1927.     for i,item in enumerate(CCC4):
1928.         if item not in nósCCC4:
1929.             nósCCC4.append(item)
1930.     nósCCT4=[]
1931.     for i,item in enumerate(CCT4):
1932.         if item not in nósCCT4:
1933.             nósCCT4.append(item)
1934.     nósCTTeTTT4=[]
1935.     for i,item in enumerate(CTTeTTT4):
1936.         if item not in nósCTTeTTT4:
1937.             nósCTTeTTT4.append(item)
1938.     nósCCC5=[]
1939.     for i,item in enumerate(CCC5):
1940.         if item not in nósCCC5:
1941.             nósCCC5.append(item)
1942.     nósCCT5=[]
1943.     for i,item in enumerate(CCT5):
1944.         if item not in nósCCT5:
1945.             nósCCT5.append(item)
1946.     nósCTTeTTT5=[]
1947.     for i,item in enumerate(CTTeTTT5):
1948.         if item not in nósCTTeTTT5:
1949.             nósCTTeTTT5.append(item)
1950.     nósCCC6=[]
1951.     for i,item in enumerate(CCC6):
1952.         if item not in nósCCC6:
1953.             nósCCC6.append(item)
1954.     nósCCT6=[]
1955.     for i,item in enumerate(CCT6):
1956.         if item not in nósCCT6:
1957.             nósCCT6.append(item)
1958.     nósCTTeTTT6=[]
1959.     for i,item in enumerate(CTTeTTT6):
1960.         if item not in nósCTTeTTT6:
1961.             nósCTTeTTT6.append(item)
1962.     nósapoioCCC2=[]
1963.     for i,item in enumerate(CCCapoio2):
1964.         if item not in nósapoioCCC2:
```

```
1965.         nósapoioCCC2.append(item)
1966. nósapoioCCT2=[]
1967. for i,item in enumerate(CCTapoio2):
1968.     if item not in nósapoioCCT2:
1969.         nósapoioCCT2.append(item)
1970. nósapoioCTTeTTT2=[]
1971. for i,item in enumerate(CTTeTTTapoio2):
1972.     if item not in nósapoioCTTeTTT2:
1973.         nósapoioCTTeTTT2.append(item)
1974. nósapoioCCC3=[]
1975. for i,item in enumerate(CCCapoio3):
1976.     if item not in nósapoioCCC3:
1977.         nósapoioCCC3.append(item)
1978. nósapoioCCT3=[]
1979. for i,item in enumerate(CCTapoio3):
1980.     if item not in nósapoioCCT3:
1981.         nósapoioCCT3.append(item)
1982. nósapoioCTTeTTT3=[]
1983. for i,item in enumerate(CTTeTTTapoio3):
1984.     if item not in nósapoioCTTeTTT3:
1985.         nósapoioCTTeTTT3.append(item)
1986. nósapoioCCC4=[]
1987. for i,item in enumerate(CCCapoio4):
1988.     if item not in nósapoioCCC4:
1989.         nósapoioCCC4.append(item)
1990. nósapoioCCT4=[]
1991. for i,item in enumerate(CCTapoio4):
1992.     if item not in nósapoioCCT4:
1993.         nósapoioCCT4.append(item)
1994. nósapoioCTTeTTT4=[]
1995. for i,item in enumerate(CTTeTTTapoio4):
1996.     if item not in nósapoioCTTeTTT4:
1997.         nósapoioCTTeTTT4.append(item)
1998. nósapoioCCC5=[]
1999. for i,item in enumerate(CCCapoio5):
2000.     if item not in nósapoioCCC5:
2001.         nósapoioCCC5.append(item)
2002. nósapoioCCT5=[]
2003. for i,item in enumerate(CCTapoio5):
2004.     if item not in nósapoioCCT5:
2005.         nósapoioCCT5.append(item)
2006. nósapoioCTTeTTT5=[]
2007. for i,item in enumerate(CTTeTTTapoio5):
2008.     if item not in nósapoioCTTeTTT5:
2009.         nósapoioCTTeTTT5.append(item)
2010. nósapoioCCC6=[]
2011. for i,item in enumerate(CCCapoio6):
2012.     if item not in nósapoioCCC6:
2013.         nósapoioCCC6.append(item)
2014. nósapoioCCT6=[]
2015. for i,item in enumerate(CCTapoio6):
2016.     if item not in nósapoioCCT6:
2017.         nósapoioCCT6.append(item)
2018. nósapoioCTTeTTT6=[]
2019. for i,item in enumerate(CTTeTTTapoio6):
2020.     if item not in nósapoioCTTeTTT6:
2021.         nósapoioCTTeTTT6.append(item)
2022.
2023. #NÓS COMPRESSÃO
2024. sfrfs=[]
```

```
ohiun=[]
brtgbhrtg=[]
tensãoCCC=[]
dbdtbdt=[]
for i,item in enumerate(nósCCC2):
    if app.Project.Structure.Nodes.Exist(item) == -1:
        nó = app.Project.Structure.Nodes.Get(item)
        for j in range(1,bars.Count+1000):
            if app.Project.Structure.Bars.Exist(j) == -1:
                barraj = app.Project.Structure.Bars.Get(j)
                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
                if startnodej.Number == nó.Number:
                    if barraj.Number not in ohiun:
                        ohiun.append([nó.Number,barraj.Number])
                        brtgbhrtg.append(barraj.Number)
                        if nó.Number not in sfrfs:
                            sfrfs.append(nó.Number)
                if nó.Number == endnodej.Number:
                    if barraj.Number not in ohiun:
                        ohiun.append([nó.Number,barraj.Number])
                        brtgbhrtg.append(barraj.Number)
                        if nó.Number not in sfrfs:
                            sfrfs.append(nó.Number)

for i,item in enumerate(brtgbhrtg):
    for sublist in spkdmvsk:
        for itemj in sublist:
            if item ==itemj[0]:
                dbdtbdt.append(itemj[1])

comprimentoLista=len(dbdtbdt)
sequencia=comprimentoLista/2
for i in range(sequencia):
    start=(i*comprimentoLista/sequencia)
    end=((i+1)*comprimentoLista/sequencia)
    tensãoCCC.append([dbdtbdt[start:end]])
for i,item in enumerate(sfrfs):
    tensãoCCC[i].Insert(0,[item])

sfrfs1=[]
ohiun1=[]
brtgbhrtg1=[]
dbdtbdt1=[]
for i,item in enumerate(nósCCC3):
    if app.Project.Structure.Nodes.Exist(item) == -1:
        nó = app.Project.Structure.Nodes.Get(item)
        for j in range(1,bars.Count+1000):
            if app.Project.Structure.Bars.Exist(j) == -1:
                barraj = app.Project.Structure.Bars.Get(j)
                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
                if startnodej.Number == nó.Number:
                    if barraj.Number not in ohiun1:
                        ohiun1.append([nó.Number,barraj.Number])
                        brtgbhrtg1.append(barraj.Number)
```

```
2081.                         if nó.Number not in sfrfs1:
2082.                             sfrfs1.append(nó.Number)
2083.                 if nó.Number == endnodej.Number:
2084.                     if barraj.Number not in ohiun1:
2085.                         ohiun1.append([nó.Number,barraj.Number])
2086.                         brtgbhrtg1.append(barraj.Number)
2087.                         if nó.Number not in sfrfs1:
2088.                             sfrfs1.append(nó.Number)
2089.
2090. for i,item in enumerate(brtgbhrtg1):
2091.     for sublist in spkdmvsk:
2092.         for itemj in sublist:
2093.             if item ==itemj[0]:
2094.                 dbdtbdt1.append(itemj[1])
2095.
2096. comprimentoLista=len(dbdtbdt1)
2097. sequencia=comprimentoLista/3
2098. for i in range(sequencia):
2099.     start=(i*comprimentoLista/sequencia)
2100.     end=((i+1)*comprimentoLista/sequencia)
2101.     tensãoCCC.append([dbdtbdt1[start:end]])
2102. for i,item in enumerate(sfrfs1):
2103.     tensãoCCC[i+sfrfs.Count].Insert(0,[item])
2104.
2105. sfrfs2=[]
2106. ohiun2=[]
2107. brtgbhrtg2=[]
2108. dbdtbdt2=[]
2109. for i,item in enumerate(nósCCC4):
2110.     if app.Project.Structure.Nodes.Exist(item) == -1:
2111.         nó = app.Project.Structure.Nodes.Get(item)
2112.         for j in range(1,bars.Count+1000):
2113.             if app.Project.Structure.Bars.Exist(j) == -1:
2114.                 barraj = app.Project.Structure.Bars.Get(j)
2115.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2116.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2117.                 if startnodej.Number == nó.Number:
2118.                     if barraj.Number not in ohiun2:
2119.                         ohiun2.append([nó.Number,barraj.Number])
2120.                         brtgbhrtg2.append(barraj.Number)
2121.                         if nó.Number not in sfrfs2:
2122.                             sfrfs2.append(nó.Number)
2123.                 if nó.Number == endnodej.Number:
2124.                     if barraj.Number not in ohiun2:
2125.                         ohiun2.append([nó.Number,barraj.Number])
2126.                         brtgbhrtg2.append(barraj.Number)
2127.                         if nó.Number not in sfrfs2:
2128.                             sfrfs2.append(nó.Number)
2129.
2130. for i,item in enumerate(brtgbhrtg2):
2131.     for sublist in spkdmvsk:
2132.         for itemj in sublist:
2133.             if item ==itemj[0]:
2134.                 dbdtbdt2.append(itemj[1])
2135.
2136. comprimentoLista=len(dbdtbdt2)
2137. sequencia=comprimentoLista/4
2138. for i in range(sequencia):
```

```
2139.     start=(i*comprimentoLista/sequencia)
2140.     end=((i+1)*comprimentoLista/sequencia)
2141.     tensãoCCC.append([dbdtbdt2[start:end]])
2142. for i,item in enumerate(sfrfs2):
2143.     tensãoCCC[i+sfrfs.Count+sfrfs1.Count].Insert(0,[item])
2144.
2145. sfrfs3=[]
2146. ohiun3=[]
2147. brtgbhrtg3=[]
2148. dbdtbdt3=[]
2149. for i,item in enumerate(nósCCC5):
2150.     if app.Project.Structure.Nodes.Exist(item) == -1:
2151.         nó = app.Project.Structure.Nodes.Get(item)
2152.         for j in range(1,bars.Count+1000):
2153.             if app.Project.Structure.Bars.Exist(j) == -1:
2154.                 barraj = app.Project.Structure.Bars.Get(j)
2155.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2156.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2157.                 if startnodej.Number == nó.Number:
2158.                     if barraj.Number not in ohiun3:
2159.                         ohiun3.append([nó.Number,barraj.Number])
2160.                         brtgbhrtg3.append(barraj.Number)
2161.                         if nó.Number not in sfrfs3:
2162.                             sfrfs3.append(nó.Number)
2163.                 if nó.Number == endnodej.Number:
2164.                     if barraj.Number not in ohiun3:
2165.                         ohiun3.append([nó.Number,barraj.Number])
2166.                         brtgbhrtg3.append(barraj.Number)
2167.                         if nó.Number not in sfrfs3:
2168.                             sfrfs3.append(nó.Number)
2169.
2170. for i,item in enumerate(brtgbhrtg3):
2171.     for sublist in spkdmvsk:
2172.         for itemj in sublist:
2173.             if item ==itemj[0]:
2174.                 dbdtbdt3.append(itemj[1])
2175.
2176. comprimentoLista=len(dbdtbdt3)
2177. sequencia=comprimentoLista/5
2178. for i in range(sequencia):
2179.     start=(i*comprimentoLista/sequencia)
2180.     end=((i+1)*comprimentoLista/sequencia)
2181.     tensãoCCC.append([dbdtbdt3[start:end]])
2182. for i,item in enumerate(sfrfs3):
2183.
tensãoCCC[i+sfrfs.Count+sfrfs1.Count+sfrfs2.Count].Insert(0,[item])
2184.
2185. sfrfs4=[]
2186. ohiun4=[]
2187. brtgbhrtg4=[]
2188. dbdtbdt4=[]
2189. for i,item in enumerate(nósapoioCCC2):
2190.     if app.Project.Structure.Nodes.Exist(item) == -1:
2191.         nó = app.Project.Structure.Nodes.Get(item)
2192.         for j in range(1,bars.Count+1000):
2193.             if app.Project.Structure.Bars.Exist(j) == -1:
2194.                 barraj = app.Project.Structure.Bars.Get(j)
```

```
2195.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2196.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2197.                 if startnodej.Number == nó.Number:
2198.                     if barraj.Number not in ohiun4:
2199.                         ohiun4.append([nó.Number,barraj.Number])
2200.                         brtgbhrtg4.append(barraj.Number)
2201.                         if nó.Number not in sfrfs4:
2202.                             sfrfs4.append(nó.Number)
2203.                 if nó.Number == endnodej.Number:
2204.                     if barraj.Number not in ohiun4:
2205.                         ohiun4.append([nó.Number,barraj.Number])
2206.                         brtgbhrtg4.append(barraj.Number)
2207.                         if nó.Number not in sfrfs4:
2208.                             sfrfs4.append(nó.Number)
2209.
2210. for i,item in enumerate(brtgbhrtg4):
2211.     for sublist in spkdmvsk:
2212.         for itemj in sublist:
2213.             if item ==itemj[0]:
2214.                 dbdtbdt4.append(itemj[1])
2215.
2216. comprimentoLista=len(dbdtbdt4)
2217. sequencia=comprimentoLista/2
2218. for i in range(sequencia):
2219.     start=(i*comprimentoLista/sequencia)
2220.     end=((i+1)*comprimentoLista/sequencia)
2221.     tensãoCCC.append([dbdtbdt4[start:end]])
2222. for i,item in enumerate(sfrfs4):
2223.
tensãoCCC[i+sfrfs.Count+sfrfs1.Count+sfrfs2.Count+sfrfs3.Count].Insert(0,[i
tem])
2224.
2225. sfrfs5=[]
2226. ohiun5=[]
2227. brtgbhrtg5=[]
2228. dbdtbdt5=[]
2229. for i,item in enumerate(nósapoioCCC3):
2230.     if app.Project.Structure.Nodes.Exist(item) == -1:
2231.         nó = app.Project.Structure.Nodes.Get(item)
2232.         for j in range(1,bars.Count+1000):
2233.             if app.Project.Structure.Bars.Exist(j) == -1:
2234.                 barraj = app.Project.Structure.Bars.Get(j)
2235.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2236.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2237.                 if startnodej.Number == nó.Number:
2238.                     if barraj.Number not in ohiun5:
2239.                         ohiun5.append([nó.Number,barraj.Number])
2240.                         brtgbhrtg5.append(barraj.Number)
2241.                         if nó.Number not in sfrfs5:
2242.                             sfrfs5.append(nó.Number)
2243.                 if nó.Number == endnodej.Number:
2244.                     if barraj.Number not in ohiun5:
2245.                         ohiun5.append([nó.Number,barraj.Number])
2246.                         brtgbhrtg5.append(barraj.Number)
2247.                         if nó.Number not in sfrfs5:
2248.                             sfrfs5.append(nó.Number)
```

```
2249.
2250. for i,item in enumerate(brtgbhrtg5):
2251.     for sublist in spkdmvsk:
2252.         for itemj in sublist:
2253.             if item ==itemj[0]:
2254.                 dbdtbdt5.append(itemj[1])
2255.
2256. comprimentoLista=len(dbdtbdt5)
2257. sequencia=comprimentoLista/3
2258. for i in range(sequencia):
2259.     start=(i*comprimentoLista/sequencia)
2260.     end=((i+1)*comprimentoLista/sequencia)
2261.     tensãoCCC.append([dbdtbdt5[start:end]])
2262. for i,item in enumerate(sfrfs5):
2263.
tensãoCCC[i+sfrfs.Count+sfrfs1.Count+sfrfs2.Count+sfrfs3.Count+sfrfs4.Count
].Insert(0,[item])
2264.
2265. sfrfs6=[]
2266. ohiun6=[]
2267. brtgbhrtg6=[]
2268. dbdtbdt6=[]
2269. for i,item in enumerate(nósapoioCCC4):
2270.     if app.Project.Structure.Nodes.Exist(item) == -1:
2271.         nó = app.Project.Structure.Nodes.Get(item)
2272.         for j in range(1,bars.Count+1000):
2273.             if app.Project.Structure.Bars.Exist(j) == -1:
2274.                 barraj = app.Project.Structure.Bars.Get(j)
2275.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2276.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2277.                 if startnodej.Number == nó.Number:
2278.                     if barraj.Number not in ohiun6:
2279.                         brtgbhrtg6.append(barraj.Number)
2280.                         ohiun6.append([nó.Number,barraj.Number])
2281.                         if nó.Number not in sfrfs6:
2282.                             sfrfs16.append(nó.Number)
2283.                 if nó.Number == endnodej.Number:
2284.                     if barraj.Number not in ohiun6:
2285.                         brtgbhrtg6.append(barraj.Number)
2286.                         ohiun6.append([nó.Number,barraj.Number])
2287.                         if nó.Number not in sfrfs6:
2288.                             sfrfs6.append(nó.Number)
2289.
2290. for i,item in enumerate(brtgbhrtg6):
2291.     for sublist in spkdmvsk:
2292.         for itemj in sublist:
2293.             if item ==itemj[0]:
2294.                 dbdtbdt6.append(itemj[1])
2295.
2296. comprimentoLista=len(dbdtbdt6)
2297. sequencia=comprimentoLista/4
2298. for i in range(sequencia):
2299.     start=(i*comprimentoLista/sequencia)
2300.     end=((i+1)*comprimentoLista/sequencia)
2301.     tensãoCCC.append([dbdtbdt6[start:end]])
2302. for i,item in enumerate(sfrfs6):
```

```
2303.
tensãoCCC[i+sfrfs.Count+sfrfs1.Count+sfrfs2.Count+sfrfs3.Count+sfrfs4.Count
+sfrfs5.Count].Insert(0,[item])
2304.
2305. sfrfs7=[]
2306. ohiun7=[]
2307. nfyjjjthhh=[]
2308. dbdtbdt7=[]
2309. for i,item in enumerate(nósapoioCCC5):
2310.     if app.Project.Structure.Nodes.Exist(item) == -1:
2311.         nó = app.Project.Structure.Nodes.Get(item)
2312.         for j in range(1,bars.Count+1000):
2313.             if app.Project.Structure.Bars.Exist(j) == -1:
2314.                 barraj = app.Project.Structure.Bars.Get(j)
2315.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2316.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2317.                 if startnodej.Number == nó.Number:
2318.                     if barraj.Number not in ohiun7:
2319.                         ohiun7.append([nó.Number,barraj.Number])
2320.                         nfyjjjthhh.append(barraj.Number)
2321.                         if nó.Number not in sfrfs7:
2322.                             sfrfs7.append(nó.Number)
2323.                 if nó.Number == endnodej.Number:
2324.                     if barraj.Number not in ohiun7:
2325.                         ohiun7.append([nó.Number,barraj.Number])
2326.                         nfyjjjthhh.append(barraj.Number)
2327.                         if nó.Number not in sfrfs11111111:
2328.                             sfrfs7.append(nó.Number)
2329.
2330. for i,item in enumerate(nfyjjjthhh):
2331.     for sublist in spkdmvsk:
2332.         for itemj in sublist:
2333.             if item ==itemj[0]:
2334.                 dbdtbdt7.append(itemj[1])
2335.
2336. comprimentoLista=len(dbdtbdt7)
2337. sequencia=comprimentoLista/5
2338. for i in range(sequencia):
2339.     start=(i*comprimentoLista/sequencia)
2340.     end=((i+1)*comprimentoLista/sequencia)
2341.     tensãoCCC.append([dbdtbdt7[start:end]])
2342. for i,item in enumerate(sfrfs7):
2343.
tensãoCCC[i+sfrfs.Count+sfrfs1.Count+sfrfs2.Count+sfrfs3.Count+sfrfs4.Count
+sfrfs5.Count+sfrfs6.Count].Insert(0,[item])
2344.
2345.
2346.
2347. sfrfs8=[]
2348. ohiun8=[]
2349. brtgbhrtg8=[]
2350. dbdtbdt8=[]
2351. for i,item in enumerate(nósCCC6):
2352.     if app.Project.Structure.Nodes.Exist(item) == -1:
2353.         nó = app.Project.Structure.Nodes.Get(item)
2354.         for j in range(1,bars.Count+1000):
2355.             if app.Project.Structure.Bars.Exist(j) == -1:
2356.                 barraj = app.Project.Structure.Bars.Get(j)
```

```
2357.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2358.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2359.                 if startnodej.Number == nó.Number:
2360.                     if barraj.Number not in ohiun8:
2361.                         ohiun8.append([nó.Number,barraj.Number])
2362.                         brtgbhrtg8.append(barraj.Number)
2363.                         if nó.Number not in sfrfs8:
2364.                             sfrfs8.append(nó.Number)
2365.                 if nó.Number == endnodej.Number:
2366.                     if barraj.Number not in ohiun8:
2367.                         ohiun8.append([nó.Number,barraj.Number])
2368.                         brtgbhrtg8.append(barraj.Number)
2369.                         if nó.Number not in sfrfs8:
2370.                             sfrfs8.append(nó.Number)
2371.
2372. for i,item in enumerate(brtgbhrtg8):
2373.     for sublist in spkdmvsk:
2374.         for itemj in sublist:
2375.             if item ==itemj[0]:
2376.                 dbdtbdt8.append(itemj[1])
2377.
2378. comprimentoLista=len(dbdtbdt8)
2379. sequencia=comprimentoLista/6
2380. for i in range(sequencia):
2381.     start=(i*comprimentoLista/sequencia)
2382.     end=((i+1)*comprimentoLista/sequencia)
2383.     tensãoCCC.append([dbdtbdt8[start:end]])
2384. for i,item in enumerate(sfrfs8):
2385.
tensãoCCC[i+sfrfs.Count+sfrfs1.Count+sfrfs2.Count+sfrfs3.Count+sfrfs4.Count
+sfrfs5.Count+sfrfs6.Count+sfrfs7.Count].Insert(0,[item])
2386.
2387. sfrfs9=[]
2388. ohiun9=[]
2389. nfyjjjthhh9=[]
2390. dbdtbdt9=[]
2391. for i,item in enumerate(nósapoioCCC6):
2392.     if app.Project.Structure.Nodes.Exist(item) == -1:
2393.         nó = app.Project.Structure.Nodes.Get(item)
2394.         for j in range(1,bars.Count+1000):
2395.             if app.Project.Structure.Bars.Exist(j) == -1:
2396.                 barraj = app.Project.Structure.Bars.Get(j)
2397.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2398.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2399.                 if startnodej.Number == nó.Number:
2400.                     if barraj.Number not in ohiun9:
2401.                         ohiun9.append([nó.Number,barraj.Number])
2402.                         nfyjjjthhh9.append(barraj.Number)
2403.                         if nó.Number not in sfrfs9:
2404.                             sfrfs9.append(nó.Number)
2405.                 if nó.Number == endnodej.Number:
2406.                     if barraj.Number not in ohiun9:
2407.                         ohiun9.append([nó.Number,barraj.Number])
2408.                         nfyjjjthhh9.append(barraj.Number)
2409.                         if nó.Number not in sfrfs9:
2410.                             sfrfs9.append(nó.Number)
```

```
2411.
2412. for i,item in enumerate(nfyjjjthhh9):
2413.     for sublist in spkdmvsk:
2414.         for itemj in sublist:
2415.             if item ==itemj[0]:
2416.                 dbdtbdt9.append(itemj[1])
2417.
2418. comprimentoLista=len(dbdtbdt9)
2419. sequencia=comprimentoLista/6
2420. for i in range(sequencia):
2421.     start=(i*comprimentoLista/sequencia)
2422.     end=((i+1)*comprimentoLista/sequencia)
2423.     tensãoCCC.append([dbdtbdt9[start:end]])
2424. for i,item in enumerate(sfrfs9):
2425.
tensãoCCC[i+sfrfs.Count+sfrfs1.Count+sfrfs2.Count+sfrfs3.Count+sfrfs4.Count
+sfrfs5.Count+sfrfs6.Count+sfrfs7.Count+sfrfs8.Count].Insert(0,[item])
2426.
2427. #NÓS CCT
2428. ohiun0=[]
2429. ohiunc2=[]
2430. edgnrdhny0=[]
2431. vsdrgfvs0=[]
2432. tensãoCCT=[]
2433. for i,item in enumerate(nósCCT2):
2434.     if app.Project.Structure.Nodes.Exist(item) == -1:
2435.         nó = app.Project.Structure.Nodes.Get(item)
2436.         for j in range(1,bars.Count+1000):
2437.             if app.Project.Structure.Bars.Exist(j) == -1:
2438.                 barraj = app.Project.Structure.Bars.Get(j)
2439.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2440.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2441.                 if startnodej.Number == nó.Number:
2442.                     if barraj.Number not in ohiun0:
2443.                         ohiun0.append([nó.Number,barraj.Number])
2444.                         edgnrdhny0.append(barraj.Number)
2445.                         if nó.Number not in ohiunc2:
2446.                             ohiunc2.append(nó.Number)
2447.                 if nó.Number == endnodej.Number:
2448.                     if barraj.Number not in ohiun0:
2449.                         ohiun0.append([nó.Number,barraj.Number])
2450.                         edgnrdhny0.append(barraj.Number)
2451.                         if nó.Number not in ohiunc2:
2452.                             ohiunc2.append(nó.Number)
2453.
2454. for i,item in enumerate(edgnrdhny0):
2455.     for sublist in spkdmvsk:
2456.         for itemj in sublist:
2457.             if item ==itemj[0]:
2458.                 vsdrgfvs0.append(itemj[1])
2459.
2460. comprimentoLista=len(vsdrgfvs0)
2461. sequencia=comprimentoLista/2
2462. for i in range(sequencia):
2463.     start=(i*comprimentoLista/sequencia)
2464.     end=((i+1)*comprimentoLista/sequencia)
2465.     tensãoCCT.append([vsdrgfvs0[start:end]])
2466. for i,item in enumerate(ohiunc2):
```

```
2467.     tensãoCCT[i].Insert(0,[item])
2468. 
2469. ohiun=[]
2470. ohiunc3=[]
2471. edgnrdhny3=[]
2472. vsdrgfvs3=[]
2473. for i,item in enumerate(nósCCT3):
2474.     if app.Project.Structure.Nodes.Exist(item) == -1:
2475.         nó = app.Project.Structure.Nodes.Get(item)
2476.         for j in range(1,bars.Count+1000):
2477.             if app.Project.Structure.Bars.Exist(j) == -1:
2478.                 barraj = app.Project.Structure.Bars.Get(j)
2479.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2480.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2481.                 if startnodej.Number == nó.Number:
2482.                     if barraj.Number not in ohiun:
2483.                         ohiun.append([nó.Number,barraj.Number])
2484.                         edgnrdhny3.append(barraj.Number)
2485.                         if nó.Number not in ohiunc3:
2486.                             ohiunc3.append(nó.Number)
2487.                 if nó.Number == endnodej.Number:
2488.                     if barraj.Number not in ohiun:
2489.                         ohiun.append([nó.Number,barraj.Number])
2490.                         edgnrdhny3.append(barraj.Number)
2491.                         if nó.Number not in ohiunc3:
2492.                             ohiunc3.append(nó.Number)
2493. 
2494. for i,item in enumerate(edgnrdhny3):
2495.     for sublist in spkdmvsk:
2496.         for itemj in sublist:
2497.             if item ==itemj[0]:
2498.                 vsdrgfvs3.append(itemj[1])
2499. 
2500. comprimentoLista=len(vsdrgfvs3)
2501. sequencia=comprimentoLista/3
2502. for i in range(sequencia):
2503.     start=(i*comprimentoLista/sequencia)
2504.     end=((i+1)*comprimentoLista/sequencia)
2505.     tensãoCCT.append([vsdrgfvs3[start:end]])
2506. for i,item in enumerate(ohiunc3):
2507.     tensãoCCT[i+ohiunc2.Count].Insert(0,[item])
2508. 
2509. ohiun4=[]
2510. ohiunc4=[]
2511. edgnrdhny4=[]
2512. vsdrgfvs4=[]
2513. for i,item in enumerate(nósCCT4):
2514.     if app.Project.Structure.Nodes.Exist(item) == -1:
2515.         nó = app.Project.Structure.Nodes.Get(item)
2516.         for j in range(1,bars.Count+1000):
2517.             if app.Project.Structure.Bars.Exist(j) == -1:
2518.                 barraj = app.Project.Structure.Bars.Get(j)
2519.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2520.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2521.                 if startnodej.Number == nó.Number:
2522.                     if barraj.Number not in ohiun4:
```

```
2523.                           ohiun4.append([nó.Number,barraj.Number])
2524.                           edgnrdhny4.append(barraj.Number)
2525.                           if nó.Number not in ohiunc4:
2526.                               ohiunc4.append(nó.Number)
2527.                   if nó.Number == endnodej.Number:
2528.                       if barraj.Number not in ohiun4:
2529.                           ohiun4.append([nó.Number,barraj.Number])
2530.                           edgnrdhny4.append(barraj.Number)
2531.                           if nó.Number not in ohiunc4:
2532.                               ohiunc4.append(nó.Number)
2533.
2534. for i,item in enumerate(edgnrdhny4):
2535.     for sublist in spkdmvsk:
2536.         for itemj in sublist:
2537.             if item ==itemj[0]:
2538.                 vsdrgfvs4.append(itemj[1])
2539.
2540. comprimentoLista=len(vsdrgfvs4)
2541. sequencia=comprimentoLista/4
2542. for i in range(sequencia):
2543.     start=(i*comprimentoLista/sequencia)
2544.     end=((i+1)*comprimentoLista/sequencia)
2545.     tensãoCCT.append([vsdrgfvs4[start:end]])
2546. for i,item in enumerate(ohiunc4):
2547.     tensãoCCT[i+ohiunc2.Count+ohiunc3.Count].Insert(0,[item])
2548.
2549. ohiun5=[]
2550. ohiunc5=[]
2551. edgnrdhny5=[]
2552. vsdrgfvs5=[]
2553. for i,item in enumerate(nósCCT5):
2554.     if app.Project.Structure.Nodes.Exist(item) == -1:
2555.         nó = app.Project.Structure.Nodes.Get(item)
2556.         for j in range(1,bars.Count+1000):
2557.             if app.Project.Structure.Bars.Exist(j) == -1:
2558.                 barraj = app.Project.Structure.Bars.Get(j)
2559.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2560.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2561.                 if startnodej.Number == nó.Number:
2562.                     if barraj.Number not in ohiun5:
2563.                         ohiun5.append([nó.Number,barraj.Number])
2564.                         edgnrdhny5.append(barraj.Number)
2565.                         if nó.Number not in ohiunc5:
2566.                             ohiunc5.append(nó.Number)
2567.                 if nó.Number == endnodej.Number:
2568.                     if barraj.Number not in ohiun5:
2569.                         ohiun5.append([nó.Number,barraj.Number])
2570.                         edgnrdhny5.append(barraj.Number)
2571.                         if nó.Number not in ohiunc5:
2572.                             ohiunc5.append(nó.Number)
2573.
2574. for i,item in enumerate(edgnrdhny5):
2575.     for sublist in spkdmvsk:
2576.         for itemj in sublist:
2577.             if item ==itemj[0]:
2578.                 vsdrgfvs5.append(itemj[1])
2579.
2580. comprimentoLista=len(vsdrgfvs5)
```

```
sequencia=comprimentoLista/5
for i in range(sequencia):
    start=(i*comprimentoLista/sequencia)
    end=((i+1)*comprimentoLista/sequencia)
    tensãoCCT.append([vsdrgfvs5[start:end]])
for i,item in enumerate(ohiunc5):
tensãoCCT[i+ohiunc2.Count+ohiunc3.Count+ohiunc4.Count].Insert(0,[item])

ohiun62=[]
ohiunc62=[]
edgnrdhny62=[]
vsdrgfvs62=[]
for i,item in enumerate(nósCCT6):
    if app.Project.Structure.Nodes.Exist(item) == -1:
        nó = app.Project.Structure.Nodes.Get(item)
        for j in range(1,bars.Count+1000):
            if app.Project.Structure.Bars.Exist(j) == -1:
                barraj = app.Project.Structure.Bars.Get(j)
                startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
                endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
                if startnodej.Number == nó.Number:
                    if barraj.Number not in ohiun62:
                        ohiun62.append([nó.Number,barraj.Number])
                        edgnrdhny62.append(barraj.Number)
                        if nó.Number not in ohiunc62:
                            ohiunc62.append(nó.Number)
                if nó.Number == endnodej.Number:
                    if barraj.Number not in ohiun62:
                        ohiun62.append([nó.Number,barraj.Number])
                        edgnrdhny62.append(barraj.Number)
                        if nó.Number not in ohiunc62:
                            ohiunc62.append(nó.Number)

for i,item in enumerate(edgnrdhny62):
    for sublist in spkdmvsk:
        for itemj in sublist:
            if item ==itemj[0]:
                vsdrgfvs62.append(itemj[1])

comprimentoLista=len(vsdrgfvs62)
sequencia=comprimentoLista/6
for i in range(sequencia):
    start=(i*comprimentoLista/sequencia)
    end=((i+1)*comprimentoLista/sequencia)
    tensãoCCT.append([vsdrgfvs62[start:end]])
for i,item in enumerate(ohiunc62):
tensãoCCT[i+ohiunc2.Count+ohiunc3.Count+ohiunc4.Count+ohiunc5.Count].Insert
(0,[item])

ohiun66=[]
ohiunc66=[]
edgnrdhny66=[]
vsdrgfvs66=[]
for i,item in enumerate(nósapoioCCT2):
    if app.Project.Structure.Nodes.Exist(item) == -1:
        nó = app.Project.Structure.Nodes.Get(item)
```

```
2636.         for j in range(1,bars.Count+1000):
2637.             if app.Project.Structure.Bars.Exist(j) == -1:
2638.                 barraj = app.Project.Structure.Bars.Get(j)
2639.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2640.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2641.                 if startnodej.Number == nó.Number:
2642.                     if barraj.Number not in ohiun66:
2643.                         ohiun66.append([nó.Number,barraj.Number])
2644.                         edgnrdhny66.append(barraj.Number)
2645.                         if nó.Number not in ohiunc66:
2646.                             ohiunc66.append(nó.Number)
2647.                 if nó.Number == endnodej.Number:
2648.                     if barraj.Number not in ohiun66:
2649.                         ohiun66.append([nó.Number,barraj.Number])
2650.                         edgnrdhny66.append(barraj.Number)
2651.                         if nó.Number not in ohiunc66:
2652.                             ohiunc66.append(nó.Number)
2653.
2654. for i,item in enumerate(edgnrdhny66):
2655.     for sublist in spkdmvsk:
2656.         for itemj in sublist:
2657.             if item ==itemj[0]:
2658.                 vsdrgfvs66.append(itemj[1])
2659.
2660. comprimentoLista=len(vsdrgfvs66)
2661. sequencia=comprimentoLista/2
2662. for i in range(sequencia):
2663.     start=(i*comprimentoLista/sequencia)
2664.     end=((i+1)*comprimentoLista/sequencia)
2665.     tensãoCCT.append([vsdrgfvs66[start:end]])
2666. for i,item in enumerate(ohiunc66):
2667.
tensãoCCT[i+ohiunc2.Count+ohiunc3.Count+ohiunc4.Count+ohiunc5.Count+ohiunc6
2.Count].Insert(0,[item])
2668.
2669. ohiun666=[]
2670. ohiunc666=[]
2671. edgnrdhny666=[]
2672. vsdrgfvs666=[]
2673. for i,item in enumerate(nósapoioCCT3):
2674.     if app.Project.Structure.Nodes.Exist(item) == -1:
2675.         nó = app.Project.Structure.Nodes.Get(item)
2676.         for j in range(1,bars.Count+1000):
2677.             if app.Project.Structure.Bars.Exist(j) == -1:
2678.                 barraj = app.Project.Structure.Bars.Get(j)
2679.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2680.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2681.                 if startnodej.Number == nó.Number:
2682.                     if barraj.Number not in ohiun666:
2683.                         ohiun666.append([nó.Number,barraj.Number])
2684.                         edgnrdhny666.append(barraj.Number)
2685.                         if nó.Number not in ohiunc666:
2686.                             ohiunc666.append(nó.Number)
2687.                 if nó.Number == endnodej.Number:
2688.                     if barraj.Number not in ohiun666:
2689.                         ohiun666.append([nó.Number,barraj.Number])
```

```
2690.                         edgnrdhny666.append(barraj.Number)
2691.                         if nó.Number not in ohiunc666:
2692.                             ohiunc666.append(nó.Number)
2693.
2694. for i,item in enumerate(edgnrdhny666):
2695.     for sublist in spkdmvsk:
2696.         for itemj in sublist:
2697.             if item ==itemj[0]:
2698.                 vsdrgfvs666.append(itemj[1])
2699.
2700. comprimentoLista=len(vsdrgfvs666)
2701. sequencia=comprimentoLista/3
2702. for i in range(sequencia):
2703.     start=(i*comprimentoLista/sequencia)
2704.     end=((i+1)*comprimentoLista/sequencia)
2705.     tensãoCCT.append([vsdrgfvs666[start:end]])
2706. for i,item in enumerate(ohiunc666):
2707.
tensãoCCT[i+ohiunc2.Count+ohiunc3.Count+ohiunc4.Count+ohiunc5.Count+ohiunc6
2.Count+ohiunc66.Count].Insert(0,[item])
2708.
2709. ohiun6666=[]
2710. ohiunc6666=[]
2711. edgnrdhny6666=[]
2712. vsdrgfvs6666=[]
2713. for i,item in enumerate(nósapoioCCT4):
2714.     if app.Project.Structure.Nodes.Exist(item) == -1:
2715.         nó = app.Project.Structure.Nodes.Get(item)
2716.         for j in range(1,bars.Count+1000):
2717.             if app.Project.Structure.Bars.Exist(j) == -1:
2718.                 barraj = app.Project.Structure.Bars.Get(j)
2719.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2720.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2721.                 if startnodej.Number == nó.Number:
2722.                     if barraj.Number not in ohiun6666:
2723.                         ohiun6666.append([nó.Number,barraj.Number])
2724.                         edgnrdhny6666.append(barraj.Number)
2725.                         if nó.Number not in ohiunc6666:
2726.                             ohiunc6666.append(nó.Number)
2727.                 if nó.Number == endnodej.Number:
2728.                     if barraj.Number not in ohiun6666:
2729.                         ohiun6666.append([nó.Number,barraj.Number])
2730.                         edgnrdhny6666.append(barraj.Number)
2731.                         if nó.Number not in ohiunc6666:
2732.                             ohiunc6666.append(nó.Number)
2733.
2734. for i,item in enumerate(edgnrdhny6666):
2735.     for sublist in spkdmvsk:
2736.         for itemj in sublist:
2737.             if item ==itemj[0]:
2738.                 vsdrgfvs6666.append(itemj[1])
2739.
2740. comprimentoLista=len(vsdrgfvs6666)
2741. sequencia=comprimentoLista/4
2742. for i in range(sequencia):
2743.     start=(i*comprimentoLista/sequencia)
2744.     end=((i+1)*comprimentoLista/sequencia)
2745.     tensãoCCT.append([vsdrgfvs6666[start:end]])
```

```
2746. for i,item in enumerate(ohiunc6666):
2747.
tensãoCCT[i+ohiunc2.Count+ohiunc3.Count+ohiunc4.Count+ohiunc5.Count+ohiunc6
2.Count+ohiunc66.Count+ohiunc666.Count].Insert(0,[item])
2748.
2749. ohiun66666=[]
2750. ohiunc66666=[]
2751. edgnrdhny66666=[]
2752. vsdrgfvs66666=[]
2753. for i,item in enumerate(nósapoioCCT5):
2754.     if app.Project.Structure.Nodes.Exist(item) == -1:
2755.         nó = app.Project.Structure.Nodes.Get(item)
2756.         for j in range(1,bars.Count+1000):
2757.             if app.Project.Structure.Bars.Exist(j) == -1:
2758.                 barraj = app.Project.Structure.Bars.Get(j)
2759.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2760.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2761.                 if startnodej.Number == nó.Number:
2762.                     if barraj.Number not in ohiun66666:
2763.                         ohiun66666.append([nó.Number,barraj.Number])
2764.                         edgnrdhny66666.append(barraj.Number)
2765.                         if nó.Number not in ohiunc66666:
2766.                             ohiunc66666.append(nó.Number)
2767.                 if nó.Number == endnodej.Number:
2768.                     if barraj.Number not in ohiun66666:
2769.                         ohiun66666.append([nó.Number,barraj.Number])
2770.                         edgnrdhny66666.append(barraj.Number)
2771.                         if nó.Number not in ohiunc66666:
2772.                             ohiunc66666.append(nó.Number)
2773.
2774. for i,item in enumerate(edgnrdhny66666):
2775.     for sublist in spkdmvsk:
2776.         for itemj in sublist:
2777.             if item ==itemj[0]:
2778.                 vsdrgfvs66666.append(itemj[1])
2779.
2780. comprimentoLista=len(vsdrgfvs66666)
2781. sequencia=comprimentoLista/5
2782. for i in range(sequencia):
2783.     start=(i*comprimentoLista/sequencia)
2784.     end=((i+1)*comprimentoLista/sequencia)
2785.     tensãoCCT.append([vsdrgfvs66666[start:end]])
2786. for i,item in enumerate(ohiunc66666):
2787.
tensãoCCT[i+ohiunc2.Count+ohiunc3.Count+ohiunc4.Count+ohiunc5.Count+ohiunc6
2.Count+ohiunc66.Count+ohiunc666.Count+ohiunc6666.Count].Insert(0,[item])
2788.
2789. ohiun666666=[]
2790. ohiunc666666=[]
2791. edgnrdhny666666=[]
2792. vsdrgfvs666666=[]
2793. for i,item in enumerate(nósapoioCCT6):
2794.     if app.Project.Structure.Nodes.Exist(item) == -1:
2795.         nó = app.Project.Structure.Nodes.Get(item)
2796.         for j in range(1,bars.Count+1000):
2797.             if app.Project.Structure.Bars.Exist(j) == -1:
2798.                 barraj = app.Project.Structure.Bars.Get(j)
```

```
2799.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2800.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2801.                 if startnodej.Number == nó.Number:
2802.                     if barraj.Number not in ohiun666666:
2803.                         ohiun666666.append([nó.Number,barraj.Number])
2804.                         edgnrdhny666666.append(barraj.Number)
2805.                         if nó.Number not in ohiunc666666:
2806.                             ohiunc666666.append(nó.Number)
2807.                 if nó.Number == endnodej.Number:
2808.                     if barraj.Number not in ohiun666666:
2809.                         ohiun666666.append([nó.Number,barraj.Number])
2810.                         edgnrdhny666666.append(barraj.Number)
2811.                         if nó.Number not in ohiunc666666:
2812.                             ohiunc666666.append(nó.Number)
2813.
2814. for i,item in enumerate(edgnrdhny666666):
2815.     for sublist in spkdmvsk:
2816.         for itemj in sublist:
2817.             if item ==itemj[0]:
2818.                 vsdrgfvs666666.append(itemj[1])
2819.
2820. comprimentoLista=len(vsdrgfvs666666)
2821. sequencia=comprimentoLista/6
2822. for i in range(sequencia):
2823.     start=(i*comprimentoLista/sequencia)
2824.     end=((i+1)*comprimentoLista/sequencia)
2825.     tensãoCCT.append([vsdrgfvs666666[start:end]])
2826. for i,item in enumerate(ohiunc666666):
2827.
tensãoCCT[i+ohiunc2.Count+ohiunc3.Count+ohiunc4.Count+ohiunc5.Count+ohiunc6
2.Count+ohiunc66.Count+ohiunc666.Count+ohiunc6666.Count].Insert(0,[item])
2828.
2829. # NÓS CTTeTTT
2830. ryhnrfgn2=[]
2831. nrfgbrnhgm2=[]
2832. nmrmtfhmt2=[]
2833. tensãoCTTeTTT=[]
2834. bfhnhgnmj2=[]
2835. for i,item in enumerate(nósCTTeTTT2):
2836.     if app.Project.Structure.Nodes.Exist(item) == -1:
2837.         nó = app.Project.Structure.Nodes.Get(item)
2838.         for j in range(1,bars.Count+1000):
2839.             if app.Project.Structure.Bars.Exist(j) == -1:
2840.                 barraj = app.Project.Structure.Bars.Get(j)
2841.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2842.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2843.                 if startnodej.Number == nó.Number:
2844.                     if barraj.Number not in ryhnrfgn2:
2845.                         ryhnrfgn2.append([nó.Number,barraj.Number])
2846.                         nmrmtfhmt2.append(barraj.Number)
2847.                         if nó.Number not in nrfgbrnhgm2:
2848.                             nrfgbrnhgm2.append(nó.Number)
2849.                 if nó.Number == endnodej.Number:
2850.                     if barraj.Number not in ryhnrfgn2:
2851.                         ryhnrfgn2.append([nó.Number,barraj.Number])
2852.                         nmrmtfhmt2.append(barraj.Number)
```

```
2853.                         if nó.Number not in nrfgbrnhgm2:
2854.                             nrfgbrnhgm2.append(nó.Number)
2855.
2856. for i,item in enumerate(nmrmtfhmt2):
2857.     for sublist in spkdmvsk:
2858.         for itemj in sublist:
2859.             if item ==itemj[0]:
2860.                 bfhnhgnmj2.append(itemj[1])
2861.
2862. comprimentoLista=len(bfhnhgnmj2)
2863. sequencia=comprimentoLista/2
2864. for i in range(sequencia):
2865.     start=(i*comprimentoLista/sequencia)
2866.     end=((i+1)*comprimentoLista/sequencia)
2867.     tensãoCTTeTTT.append([bfhnhgnmj2[start:end]])
2868. for i,item in enumerate(nrfgbrnhgm2):
2869.     tensãoCTTeTTT[i].Insert(0,[item])
2870.
2871. ryhnrfgn3=[]
2872. nrfgbrnhgm3=[]
2873. nmrmtfhmt3=[]
2874. bfhnhgnmj3=[]
2875. for i,item in enumerate(nósCTTeTTT3):
2876.     if app.Project.Structure.Nodes.Exist(item) == -1:
2877.         nó = app.Project.Structure.Nodes.Get(item)
2878.         for j in range(1,bars.Count+1000):
2879.             if app.Project.Structure.Bars.Exist(j) == -1:
2880.                 barraj = app.Project.Structure.Bars.Get(j)
2881.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2882.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2883.                 if startnodej.Number == nó.Number:
2884.                     if barraj.Number not in ryhnrfgn3:
2885.                         ryhnrfgn3.append([nó.Number,barraj.Number])
2886.                         nmrmtfhmt3.append(barraj.Number)
2887.                         if nó.Number not in nrfgbrnhgm3:
2888.                             nrfgbrnhgm3.append(nó.Number)
2889.                 if nó.Number == endnodej.Number:
2890.                     if barraj.Number not in ryhnrfgn3:
2891.                         ryhnrfgn3.append([nó.Number,barraj.Number])
2892.                         nmrmtfhmt3.append(barraj.Number)
2893.                         if nó.Number not in nrfgbrnhgm3:
2894.                             nrfgbrnhgm3.append(nó.Number)
2895.
2896. for i,item in enumerate(nmrmtfhmt3):
2897.     for sublist in spkdmvsk:
2898.         for itemj in sublist:
2899.             if item ==itemj[0]:
2900.                 bfhnhgnmj3.append(itemj[1])
2901.
2902. comprimentoLista=len(bfhnhgnmj3)
2903. sequencia=comprimentoLista/3
2904. for i in range(sequencia):
2905.     start=(i*comprimentoLista/sequencia)
2906.     end=((i+1)*comprimentoLista/sequencia)
2907.     tensãoCTTeTTT.append([bfhnhgnmj3[start:end]])
2908. for i,item in enumerate(nrfgbrnhgm3):
2909.     tensãoCTTeTTT[i+nrfgbrnhgm2.Count].Insert(0,[item])
2910.
```

```
2911. ryhnrfgn4=[]
2912. nrfgbrnhgm4=[]
2913. nmrmtfhmt4=[]
2914. bfhnhgnmj4=[]
2915. for i,item in enumerate(nósCTTeTTT4):
2916.     if app.Project.Structure.Nodes.Exist(item) == -1:
2917.         nó = app.Project.Structure.Nodes.Get(item)
2918.         for j in range(1,bars.Count+1000):
2919.             if app.Project.Structure.Bars.Exist(j) == -1:
2920.                 barraj = app.Project.Structure.Bars.Get(j)
2921.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2922.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2923.                 if startnodej.Number == nó.Number:
2924.                     if barraj.Number not in ryhnrfgn4:
2925.                         ryhnrfgn4.append([nó.Number,barraj.Number])
2926.                         nmrmtfhmt4.append(barraj.Number)
2927.                         if nó.Number not in nrfgbrnhgm4:
2928.                             nrfgbrnhgm4.append(nó.Number)
2929.                 if nó.Number == endnodej.Number:
2930.                     if barraj.Number not in ryhnrfgn4:
2931.                         ryhnrfgn4.append([nó.Number,barraj.Number])
2932.                         nmrmtfhmt4.append(barraj.Number)
2933.                         if nó.Number not in nrfgbrnhgm4:
2934.                             nrfgbrnhgm4.append(nó.Number)
2935.
2936. for i,item in enumerate(nmrmtfhmt4):
2937.     for sublist in spkdmvsk:
2938.         for itemj in sublist:
2939.             if item ==itemj[0]:
2940.                 bfhnhgnmj4.append(itemj[1])
2941.
2942. comprimentoLista=len(bfhnhgnmj4)
2943. sequencia=comprimentoLista/4
2944. for i in range(sequencia):
2945.     start=(i*comprimentoLista/sequencia)
2946.     end=((i+1)*comprimentoLista/sequencia)
2947.     tensãoCTTeTTT.append([bfhnhgnmj4[start:end]])
2948. for i,item in enumerate(nrfgbrnhgm4):
2949.
tensãoCTTeTTT[i+nrfgbrnhgm2.Count+nrfgbrnhgm3.Count].Insert(0,[item])
2950.
2951. ryhnrfgn5=[]
2952. nrfgbrnhgm5=[]
2953. nmrmtfhmt5=[]
2954. bfhnhgnmj5=[]
2955. for i,item in enumerate(nósCTTeTTT5):
2956.     if app.Project.Structure.Nodes.Exist(item) == -1:
2957.         nó = app.Project.Structure.Nodes.Get(item)
2958.         for j in range(1,bars.Count+1000):
2959.             if app.Project.Structure.Bars.Exist(j) == -1:
2960.                 barraj = app.Project.Structure.Bars.Get(j)
2961.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
2962.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
2963.                 if startnodej.Number == nó.Number:
2964.                     if barraj.Number not in ryhnrfgn5:
2965.                         ryhnrfgn5.append([nó.Number,barraj.Number])
```

```
2966.                         nmrmtfhmt5.append(barraj.Number)
2967.                         if nó.Number not in nrfgbrnhgm5:
2968.                             nrfgbrnhgm5.append(nó.Number)
2969.                 if nó.Number == endnodej.Number:
2970.                     if barraj.Number not in ryhnrfgn5:
2971.                         ryhnrfgn5.append([nó.Number,barraj.Number])
2972.                         nmrmtfhmt5.append(barraj.Number)
2973.                         if nó.Number not in nrfgbrnhgm5:
2974.                             nrfgbrnhgm5.append(nó.Number)
2975.
2976. for i,item in enumerate(nmrmtfhmt5):
2977.     for sublist in spkdmvsk:
2978.         for itemj in sublist:
2979.             if item ==itemj[0]:
2980.                 bfhnhgnmj5.append(itemj[1])
2981.
2982. comprimentoLista=len(bfhnhgnmj5)
2983. sequencia=comprimentoLista/5
2984. for i in range(sequencia):
2985.     start=(i*comprimentoLista/sequencia)
2986.     end=((i+1)*comprimentoLista/sequencia)
2987.     tensãoCTTeTTT.append([bfhnhgnmj5[start:end]])
2988. for i,item in enumerate(nrfgbrnhgm5):
2989.
tensãoCTTeTTT[i+nrfgbrnhgm2.Count+nrfgbrnhgm3.Count+nrfgbrnhgm4.Count].Inse
rt(0,[item])
2990.
2991. ryhnrfgn6=[]
2992. nrfgbrnhgm6=[]
2993. nmrmtfhmt6=[]
2994. bfhnhgnmj6=[]
2995. for i,item in enumerate(nósCTTeTTT6):
2996.     if app.Project.Structure.Nodes.Exist(item) == -1:
2997.         nó = app.Project.Structure.Nodes.Get(item)
2998.         for j in range(1,bars.Count+1000):
2999.             if app.Project.Structure.Bars.Exist(j) == -1:
3000.                 barraj = app.Project.Structure.Bars.Get(j)
3001.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
3002.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
3003.                 if startnodej.Number == nó.Number:
3004.                     if barraj.Number not in ryhnrfgn6:
3005.                         ryhnrfgn6.append([nó.Number,barraj.Number])
3006.                         nmrmtfhmt6.append(barraj.Number)
3007.                         if nó.Number not in nrfgbrnhgm6:
3008.                             nrfgbrnhgm6.append(nó.Number)
3009.                 if nó.Number == endnodej.Number:
3010.                     if barraj.Number not in ryhnrfgn6:
3011.                         ryhnrfgn6.append([nó.Number,barraj.Number])
3012.                         nmrmtfhmt6.append(barraj.Number)
3013.                         if nó.Number not in nrfgbrnhgm6:
3014.                             nrfgbrnhgm6.append(nó.Number)
3015.
3016. for i,item in enumerate(nmrmtfhmt6):
3017.     for sublist in spkdmvsk:
3018.         for itemj in sublist:
3019.             if item ==itemj[0]:
3020.                 bfhnhgnmj6.append(itemj[1])
3021.
```

```
3022. comprimentoLista=len(bfhnhgnmj6)
3023. sequencia=comprimentoLista/6
3024. for i in range(sequencia):
3025.     start=(i*comprimentoLista/sequencia)
3026.     end=((i+1)*comprimentoLista/sequencia)
3027.     tensãoCTTeTTT.append([bfhnhgnmj6[start:end]])
3028. for i,item in enumerate(nrfgbrnhgm6):
3029.
tensãoCTTeTTT[i+nrfgbrnhgm2.Count+nrfgbrnhgm3.Count+nrfgbrnhgm4.Count+nrfgb
rnhgm5.Count].Insert(0,[item])
3030.
3031. ryhnrfgn61=[]
3032. nrfgbrnhgm61=[]
3033. nmrmtfhmt61=[]
3034. bfhnhgnmj61=[]
3035. for i,item in enumerate(nósapoioCTTeTTT2):
3036.     if app.Project.Structure.Nodes.Exist(item) == -1:
3037.         nó = app.Project.Structure.Nodes.Get(item)
3038.         for j in range(1,bars.Count+1000):
3039.             if app.Project.Structure.Bars.Exist(j) == -1:
3040.                 barraj = app.Project.Structure.Bars.Get(j)
3041.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
3042.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
3043.                 if startnodej.Number == nó.Number:
3044.                     if barraj.Number not in ryhnrfgn61:
3045.                         ryhnrfgn61.append([nó.Number,barraj.Number])
3046.                         nmrmtfhmt61.append(barraj.Number)
3047.                         if nó.Number not in nrfgbrnhgm61:
3048.                             nrfgbrnhgm61.append(nó.Number)
3049.                 if nó.Number == endnodej.Number:
3050.                     if barraj.Number not in ryhnrfgn61:
3051.                         ryhnrfgn61.append([nó.Number,barraj.Number])
3052.                         nmrmtfhmt61.append(barraj.Number)
3053.                         if nó.Number not in nrfgbrnhgm61:
3054.                             nrfgbrnhgm61.append(nó.Number)
3055.
3056. for i,item in enumerate(nmrmtfhmt61):
3057.     for sublist in spkdmvsk:
3058.         for itemj in sublist:
3059.             if item ==itemj[0]:
3060.                 bfhnhgnmj61.append(itemj[1])
3061.
3062. comprimentoLista=len(bfhnhgnmj61)
3063. sequencia=comprimentoLista/2
3064. for i in range(sequencia):
3065.     start=(i*comprimentoLista/sequencia)
3066.     end=((i+1)*comprimentoLista/sequencia)
3067.     tensãoCTTeTTT.append([bfhnhgnmj61[start:end]])
3068. for i,item in enumerate(nrfgbrnhgm61):
3069.
tensãoCTTeTTT[i+nrfgbrnhgm2.Count+nrfgbrnhgm3.Count+nrfgbrnhgm4.Count+nrfgb
rnhgm5.Count+nrfgbrnhgm6.Count].Insert(0,[item])
3070.
3071. ryhnrfgn66=[]
3072. nrfgbrnhgm66=[]
3073. nmrmtfhmt66=[]
3074. bfhnhgnmj66=[]
3075. for i,item in enumerate(nósapoioCTTeTTT3):
```

```
3076.     if app.Project.Structure.Nodes.Exist(item) == -1:
3077.         nó = app.Project.Structure.Nodes.Get(item)
3078.         for j in range(1,bars.Count+1000):
3079.             if app.Project.Structure.Bars.Exist(j) == -1:
3080.                 barraj = app.Project.Structure.Bars.Get(j)
3081.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
3082.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
3083.                 if startnodej.Number == nó.Number:
3084.                     if barraj.Number not in ryhnrfgn66:
3085.                         ryhnrfgn66.append([nó.Number,barraj.Number])
3086.                         nmrmtfhmt66.append(barraj.Number)
3087.                         if nó.Number not in nrfgbrnhgm66:
3088.                             nrfgbrnhgm66.append(nó.Number)
3089.                 if nó.Number == endnodej.Number:
3090.                     if barraj.Number not in ryhnrfgn66:
3091.                         ryhnrfgn66.append([nó.Number,barraj.Number])
3092.                         nmrmtfhmt66.append(barraj.Number)
3093.                         if nó.Number not in nrfgbrnhgm66:
3094.                             nrfgbrnhgm66.append(nó.Number)
3095.
3096. for i,item in enumerate(nmrmtfhmt66):
3097.     for sublist in spkdmvsk:
3098.         for itemj in sublist:
3099.             if item ==itemj[0]:
3100.                 bfhnhgnmj66.append(itemj[1])
3101.
3102. comprimentoLista=len(bfhnhgnmj66)
3103. sequencia=comprimentoLista/3
3104. for i in range(sequencia):
3105.     start=(i*comprimentoLista/sequencia)
3106.     end=((i+1)*comprimentoLista/sequencia)
3107.     tensãoCTTeTTT.append([bfhnhgnmj66[start:end]])
3108.
3109. for i,item in enumerate(nrfgbrnhgm66):
3110.
tensãoCTTeTTT[i+nrfgbrnhgm2.Count+nrfgbrnhgm3.Count+nrfgbrnhgm4.Count+nrfgb
rnhgm5.Count+nrfgbrnhgm6.Count+nrfgbrnhgm61.Count].Insert(0,[item])
3111.
3112. ryhnrfgn666=[]
3113. nrfgbrnhgm666=[]
3114. nmrmtfhmt666=[]
3115. bfhnhgnmj666=[]
3116. for i,item in enumerate(nósapoioCTTeTTT4):
3117.     if app.Project.Structure.Nodes.Exist(item) == -1:
3118.         nó = app.Project.Structure.Nodes.Get(item)
3119.         for j in range(1,bars.Count+1000):
3120.             if app.Project.Structure.Bars.Exist(j) == -1:
3121.                 barraj = app.Project.Structure.Bars.Get(j)
3122.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
3123.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
3124.                 if startnodej.Number == nó.Number:
3125.                     if barraj.Number not in ryhnrfgn666:
3126.                         ryhnrfgn666.append([nó.Number,barraj.Number])
3127.                         nmrmtfhmt666.append(barraj.Number)
3128.                         if nó.Number not in nrfgbrnhgm666:
3129.                             nrfgbrnhgm666.append(nó.Number)
```

```
3130.                 if nó.Number == endnodej.Number:
3131.                     if barraj.Number not in ryhnrfgn666:
3132.                         ryhnrfgn666.append([nó.Number,barraj.Number])
3133.                         nmrmtfhmt666.append(barraj.Number)
3134.                         if nó.Number not in nrfgbrnhgm666:
3135.                             nrfgbrnhgm666.append(nó.Number)
3136.
3137. for i,item in enumerate(nmrmtfhmt666):
3138.     for sublist in spkdmvsk:
3139.         for itemj in sublist:
3140.             if item ==itemj[0]:
3141.                 bfhnhgnmj666.append(itemj[1])
3142.
3143. comprimentoLista=len(bfhnhgnmj666)
3144. sequencia=comprimentoLista/4
3145. for i in range(sequencia):
3146.     start=(i*comprimentoLista/sequencia)
3147.     end=((i+1)*comprimentoLista/sequencia)
3148.     tensãoCTTeTTT.append([bfhnhgnmj666[start:end]])
3149. for i,item in enumerate(nrfgbrnhgm666):
3150.
tensãoCTTeTTT[i+nrfgbrnhgm2.Count+nrfgbrnhgm3.Count+nrfgbrnhgm4.Count+nrfgb
rnhgm5.Count+nrfgbrnhgm6.Count+nrfgbrnhgm61.Count+nrfgbrnhgm66.Count].Inser
t(0,[item])
3151.
3152. ryhnrfgn6666=[]
3153. nrfgbrnhgm6666=[]
3154. nmrmtfhmt6666=[]
3155. bfhnhgnmj6666=[]
3156. for i,item in enumerate(nósapoioCTTeTTT5):
3157.     if app.Project.Structure.Nodes.Exist(item) == -1:
3158.         nó = app.Project.Structure.Nodes.Get(item)
3159.         for j in range(1,bars.Count+1000):
3160.             if app.Project.Structure.Bars.Exist(j) == -1:
3161.                 barraj = app.Project.Structure.Bars.Get(j)
3162.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
3163.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
3164.                 if startnodej.Number == nó.Number:
3165.                     if barraj.Number not in ryhnrfgn6666:
3166.
ryhnrfgn6666.append([nó.Number,barraj.Number])
3167.                         nmrmtfhmt6666.append(barraj.Number)
3168.                         if nó.Number not in nrfgbrnhgm6666:
3169.                             nrfgbrnhgm6666.append(nó.Number)
3170.                 if nó.Number == endnodej.Number:
3171.                     if barraj.Number not in ryhnrfgn6666:
3172.
ryhnrfgn6666.append([nó.Number,barraj.Number])
3173.                         nmrmtfhmt6666.append(barraj.Number)
3174.                         if nó.Number not in nrfgbrnhgm6666:
3175.                             nrfgbrnhgm6666.append(nó.Number)
3176.
3177. for i,item in enumerate(nmrmtfhmt6666):
3178.     for sublist in spkdmvsk:
3179.         for itemj in sublist:
3180.             if item ==itemj[0]:
3181.                 bfhnhgnmj6666.append(itemj[1])
3182.
```

```
3183. comprimentoLista=len(bfhnhgnmj6666)
3184. sequencia=comprimentoLista/5
3185. for i in range(sequencia):
3186.     start=(i*comprimentoLista/sequencia)
3187.     end=((i+1)*comprimentoLista/sequencia)
3188.     tensãoCTTeTTT.append([bfhnhgnmj6666[start:end]])
3189. for i,item in enumerate(nrfgbrnhgm6666):
3190.
tensãoCTTeTTT[i+nrfgbrnhgm2.Count+nrfgbrnhgm3.Count+nrfgbrnhgm4.Count+nrfgb
rnhgm5.Count+nrfgbrnhgm6.Count+nrfgbrnhgm61.Count+nrfgbrnhgm66.Count+nrfgbr
nhgm666.Count].Insert(0,[item])
3191.
3192. ryhnrfgn6=[]
3193. nrfgbrnhgm6=[]
3194. nmrmtfhmt6=[]
3195. bfhnhgnmj6=[]
3196. for i,item in enumerate(nósCTTeTTT6):
3197.     if app.Project.Structure.Nodes.Exist(item) == -1:
3198.         nó = app.Project.Structure.Nodes.Get(item)
3199.         for j in range(1,bars.Count+1000):
3200.             if app.Project.Structure.Bars.Exist(j) == -1:
3201.                 barraj = app.Project.Structure.Bars.Get(j)
3202.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
3203.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
3204.                 if startnodej.Number == nó.Number:
3205.                     if barraj.Number not in ryhnrfgn6:
3206.                         ryhnrfgn6.append([nó.Number,barraj.Number])
3207.                         nmrmtfhmt6.append(barraj.Number)
3208.                         if nó.Number not in nrfgbrnhgm6:
3209.                             nrfgbrnhgm6.append(nó.Number)
3210.                 if nó.Number == endnodej.Number:
3211.                     if barraj.Number not in ryhnrfgn6:
3212.                         ryhnrfgn6.append([nó.Number,barraj.Number])
3213.                         nmrmtfhmt6.append(barraj.Number)
3214.                         if nó.Number not in nrfgbrnhgm6:
3215.                             nrfgbrnhgm6.append(nó.Number)
3216.
3217. for i,item in enumerate(nmrmtfhmt6):
3218.     for sublist in spkdmvsk:
3219.         for itemj in sublist:
3220.             if item ==itemj[0]:
3221.                 bfhnhgnmj6.append(itemj[1])
3222.
3223. comprimentoLista=len(bfhnhgnmj6)
3224. sequencia=comprimentoLista/6
3225. for i in range(sequencia):
3226.     start=(i*comprimentoLista/sequencia)
3227.     end=((i+1)*comprimentoLista/sequencia)
3228.     tensãoCTTeTTT.append([bfhnhgnmj6[start:end]])
3229. for i,item in enumerate(nrfgbrnhgm6):
3230.
tensãoCTTeTTT[i+nrfgbrnhgm2.Count+nrfgbrnhgm3.Count+nrfgbrnhgm4.Count+nrfgb
rnhgm5.Count+nrfgbrnhgm6.Count+nrfgbrnhgm61.Count+nrfgbrnhgm66.Count+nrfgbr
nhgm666.Count+nrfgbrnhgm6666.Count].Insert(0,[item])
3231.
3232. ryhnrfgn66666=[]
3233. nrfgbrnhgm66666=[]
3234. nmrmtfhmt66666=[]
```

```
3235. bfhnhgnmj66666=[]
3236. for i,item in enumerate(nósapoioCTTeTTT6):
3237.     if app.Project.Structure.Nodes.Exist(item) == -1:
3238.         nó = app.Project.Structure.Nodes.Get(item)
3239.         for j in range(1,bars.Count+1000):
3240.             if app.Project.Structure.Bars.Exist(j) == -1:
3241.                 barraj = app.Project.Structure.Bars.Get(j)
3242.                 startnodej =
app.Project.Structure.Nodes.Get(barraj.StartNode)
3243.                 endnodej =
app.Project.Structure.Nodes.Get(barraj.EndNode)
3244.                 if startnodej.Number == nó.Number:
3245.                     if barraj.Number not in ryhnrfgn66666:
3246.
ryhnrfgn66666.append([nó.Number,barraj.Number])
3247.                         nmrmtfhmt66666.append(barraj.Number)
3248.                         if nó.Number not in nrfgbrnhgm66666:
3249.                             nrfgbrnhgm66666.append(nó.Number)
3250.                 if nó.Number == endnodej.Number:
3251.                     if barraj.Number not in ryhnrfgn66666:
3252.
ryhnrfgn66666.append([nó.Number,barraj.Number])
3253.                         nmrmtfhmt66666.append(barraj.Number)
3254.                         if nó.Number not in nrfgbrnhgm66666:
3255.                             nrfgbrnhgm66666.append(nó.Number)
3256.
3257. for i,item in enumerate(nmrmtfhmt66666):
3258.     for sublist in spkdmvsk:
3259.         for itemj in sublist:
3260.             if item ==itemj[0]:
3261.                 bfhnhgnmj66666.append(itemj[1])
3262.
3263. comprimentoLista=len(bfhnhgnmj66666)
3264. sequencia=comprimentoLista/6
3265. for i in range(sequencia):
3266.     start=(i*comprimentoLista/sequencia)
3267.     end=((i+1)*comprimentoLista/sequencia)
3268.     tensãoCTTeTTT.append([bfhnhgnmj66666[start:end]])
3269. for i,item in enumerate(nrfgbrnhgm66666):
3270.
tensãoCTTeTTT[i+nrfgbrnhgm2.Count+nrfgbrnhgm3.Count+nrfgbrnhgm4.Count+nrfgb
rnhgm5.Count+nrfgbrnhgm6.Count+nrfgbrnhgm61.Count+nrfgbrnhgm66.Count+nrfgbr
nhgm666.Count+nrfgbrnhgm6666.Count+nrfgbrnhgm6.Count].Insert(0,[item])
3271.
3272. # VERIFICAÇÃO DOS NÓS
3273. nóOk=[]
3274. nãoOK=[]
3275. PesopróprioCCC=[]
3276. Pesoprópriocccc=[]
3277. nóscomforçaCCC=[]
3278. fgnfgnmjum=[]
3279. fgnfhgnmjum=[]
3280. otknlrfbe=[]
3281. for i,item in enumerate(tensãoCCC):
3282.     if app.Project.Structure.Nodes.Exist(tensãoCCC[i][0][0]) == -1:
3283.         nó = app.Project.Structure.Nodes.Get(tensãoCCC[i][0][0])
3284.         if nó.Z>0:
3285.             for m,itemm in enumerate(Nóseforçaas):
3286.                 if nó.Number == Nóseforçaas[m][0]:
```

```
3287.
nóscomforçaCCC.append([Nóseforçaas[m][0],Nóseforçaas[m][1]])
3288.
3289.             for n,itemn in enumerate(tensãoCCC):
3290.                 for o,itemo in enumerate(nóscomforçaCCC):
3291.                     fgnfgnmjum.append(nóscomforçaCCC[o][1])
3292.                     if nóscomforçaCCC[o][0] == tensãoCCC[n][0][0]:
3293.                         fgnfhgnmjum.append(tensãoCCC[n][0][0])
3294.
tensão_força=nóscomforçaCCC[o][1]/(EspessuraForça*LarguraForça)
3295.                         if abs(tensão_força) not in tensãoCCC[n][1]:
3296.
tensãoCCC[n][1].insert(0,abs(tensão_força))
3297.
3298.             for j,itemj in enumerate(tensãoCCC[i][1]):
3299.                 otknlrfbe.append(itemj)
3300.                 if tensãoCCC[i][1].Count==2:
3301.                     if itemj>0:
3302.                         if itemj > 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3303.                             if nó.Number not in nãoOK:
3304.                                 nãoOK.append(nó.Number)
3305.                         if itemj<= 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3306.                             if EspessuraForça<Espessura:
3307.                                     if itemj <
(Espessura/EspessuraForça)*fcd:
3308.                                         if itemj < 3.3*fcd:
3309.                                             if nó.Number not in nóOk:
3310.
nóOk.append(nó.Number)
3311.                                         if itemj > 3.3*fcd:
3312.                                             if nó.Number not in
nãoOK:
3313.
nãoOK.append(nó.Number)
3314.                                     if itemj >
(Espessura/EspessuraForça)*fcd:
3315.                                         if nó.Number not in nãoOK:
3316.                                             nãoOK.append(nó.Number)
3317.                             if EspessuraForça>=Espessura:
3318.                                 if nó.Number not in nóOk:
3319.                                     nóOk.append(nó.Number)
3320.
3321.                 if tensãoCCC[i][1].Count==3:
3322.                     if itemj>0:
3323.                         if itemj > 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3324.                             if nó.Number not in nãoOK:
3325.                                 nãoOK.append(nó.Number)
3326.                         if itemj<= 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3327.                             if EspessuraForça<Espessura:
3328.                                     if itemj <
(Espessura/EspessuraForça)*fcd:
3329.                                         if itemj < 3.3*fcd:
3330.                                             if nó.Number not in nóOk:
3331.
nóOk.append(nó.Number)
3332.                                         if itemj > 3.3*fcd:
```

```
3333.                                                   if nó.Number not in
nãoOK:
3334.
nãoOK.append(nó.Number)
3335.                                           if itemj >
(Espessura/EspessuraForça)*fcd:
3336.                                               if nó.Number not in nãoOK:
3337.                                                   nãoOK.append(nó.Number)
3338.                                   if EspessuraForça>=Espessura:
3339.                                       if nó.Number not in nóOk:
3340.                                           nóOk.append(nó.Number)
3341.
3342.                       if tensãoCCC[i][1].Count==4:
3343.                           if itemj>0:
3344.                               if itemj > 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3345.                                   if nó.Number not in nãoOK:
3346.                                       nãoOK.append(nó.Number)
3347.                               if itemj<= 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3348.                                   if EspessuraForça<Espessura:
3349.                                           if itemj <
(Espessura/EspessuraForça)*fcd:
3350.                                               if itemj < 3.3*fcd:
3351.                                                   if nó.Number not in nóOk:
3352.
nóOk.append(nó.Number)
3353.                                               if itemj > 3.3*fcd:
3354.                                                   if nó.Number not in
nãoOK:
3355.
nãoOK.append(nó.Number)
3356.                                           if itemj >
(Espessura/EspessuraForça)*fcd:
3357.                                               if nó.Number not in nãoOK:
3358.                                                   nãoOK.append(nó.Number)
3359.                                   if EspessuraForça>=Espessura:
3360.                                       if nó.Number not in nóOk:
3361.                                           nóOk.append(nó.Number)
3362.
3363.                       if tensãoCCC[i][1].Count==5:
3364.                           if itemj>0:
3365.                               if itemj > 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3366.                                   if nó.Number not in nãoOK:
3367.                                       nãoOK.append(nó.Number)
3368.                               if itemj<= 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3369.                                   if EspessuraForça<Espessura:
3370.                                           if itemj <
(Espessura/EspessuraForça)*fcd:
3371.                                               if itemj < 3.3*fcd:
3372.                                                   if nó.Number not in nóOk:
3373.
nóOk.append(nó.Number)
3374.                                               if itemj > 3.3*fcd:
3375.                                                   if nó.Number not in
nãoOK:
3376.
nãoOK.append(nó.Number)
```

```
3377.                                       if itemj >
(Espessura/EspessuraForça)*fcd:
3378.                                           if nó.Number not in nãoOK:
3379.                                               nãoOK.append(nó.Number)
3380.                                   if EspessuraForça>=Espessura:
3381.                                       if nó.Number not in nóOk:
3382.                                           nóOk.append(nó.Number)
3383.
3384.                       if tensãoCCC[i][1].Count==6:
3385.                           if itemj>0:
3386.                               if itemj > 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3387.                                   if nó.Number not in nãoOK:
3388.                                       nãoOK.append(nó.Number)
3389.                               if itemj<= 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3390.                                   if EspessuraForça<Espessura:
3391.                                       if itemj <
(Espessura/EspessuraForça)*fcd:
3392.                                           if itemj < 3.3*fcd:
3393.                                               if nó.Number not in nóOk:
3394.
nóOk.append(nó.Number)
3395.                                           if itemj > 3.3*fcd:
3396.                                               if nó.Number not in
nãoOK:
3397.
nãoOK.append(nó.Number)
3398.                                       if itemj >
(Espessura/EspessuraForça)*fcd:
3399.                                           if nó.Number not in nãoOK:
3400.                                               nãoOK.append(nó.Number)
3401.                                   if EspessuraForça>=Espessura:
3402.                                       if nó.Number not in nóOk:
3403.                                           nóOk.append(nó.Number)
3404.           if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
3405.               if nó.Z==0:
3406.                   for j in range(1, cases.Count+1):
3407.                       casos= app.Project.Structure.Cases.Get(j)
3408.                       forças1 = reações.Value(nó.Number, casos.Number)
3409.                       fZ=forças1.FZ
3410.                       Pesopróprioccc.append(fZ)
3411.                   Peso=0
3412.                   for l in Pesopróprioccc:
3413.                       Peso=Peso+l
3414.                   PesopróprioCCC.append(Peso)
3415.
tensão_apoio=PesopróprioCCC[0]/(EspessuraApoio*LarguraApoio)
3416.                   tensãoCCC[i][1].append(tensão_apoio)
3417.                   for j,itemj in enumerate(tensãoCCC[i][1]):
3418.                       if tensãoCCC[i][1].Count==3:
3419.                           if itemj>=0:
3420.                               if itemj > 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3421.                                   if nó.Number not in nãoOK:
3422.                                       nãoOK.append(nó.Number)
3423.                               if itemj<= 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3424.                                   if EspessuraApoio<Espessura:
```

```
3425.                                       if itemj <
(Espessura/EspessuraApoio)*fcd:
3426.                                           if itemj < 3.3*fcd:
3427.                                               if nó.Number not in nóOk:
3428.
nóOk.append(nó.Number)
3429.                                           if itemj > 3.3*fcd:
3430.                                               if nó.Number not in
nãoOK:
3431.
nãoOK.append(nó.Number)
3432.                                       if itemj >
(Espessura/EspessuraApoio)*fcd:
3433.                                           if nó.Number not in nãoOK:
3434.                                               nãoOK.append(nó.Number)
3435.                                   if EspessuraApoio>=Espessura:
3436.                                       if nó.Number not in nóOk:
3437.                                           nóOk.append(nó.Number)
3438.
3439.                       if tensãoCCC[i][1].Count==4:
3440.                           if itemj>=0:
3441.                               if itemj > 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3442.                                   if nó.Number not in nãoOK:
3443.                                       nãoOK.append(nó.Number)
3444.                               if itemj<=0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3445.                                   if EspessuraApoio<Espessura:
3446.                                       if itemj <
(Espessura/EspessuraApoio)*fcd:
3447.                                           if itemj < 3.3*fcd:
3448.                                               if nó.Number not in nóOk:
3449.
nóOk.append(nó.Number)
3450.                                           if itemj > 3.3*fcd:
3451.                                               if nó.Number not in
nãoOK:
3452.
nãoOK.append(nó.Number)
3453.                                       if itemj >
(Espessura/EspessuraApoio)*fcd:
3454.                                           if nó.Number not in nãoOK:
3455.                                               nãoOK.append(nó.Number)
3456.                                   if EspessuraApoio>=Espessura:
3457.                                       if nó.Number not in nóOk:
3458.                                           nóOk.append(nó.Number)
3459.
3460.                       if tensãoCCC[i][1].Count==5:
3461.                           if itemj>0:
3462.                               if itemj > 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3463.                                   if nó.Number not in nãoOK:
3464.                                       nãoOK.append(nó.Number)
3465.                               if itemj<= 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3466.                                   if EspessuraApoio<Espessura:
3467.                                       if itemj <
(Espessura/EspessuraApoio)*fcd:
3468.                                           if itemj < 3.3*fcd:
3469.                                               if nó.Number not in nóOk:
```

```
3470.
nóOk.append(nó.Number)
3471.                                           if itemj > 3.3*fcd:
3472.                                               if nó.Number not in
nãoOK:
3473.
nãoOK.append(nó.Number)
3474.                                       if itemj >
(Espessura/EspessuraApoio)*fcd:
3475.                                           if nó.Number not in nãoOK:
3476.                                               nãoOK.append(nó.Number)
3477.                                   if EspessuraApoio>=Espessura:
3478.                                       if nó.Number not in nóOk:
3479.                                           nóOk.append(nó.Number)
3480.                       if tensãoCCC[i][1].Count==6:
3481.                           if itemj>0:
3482.                               if itemj > 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3483.                                   if nó.Number not in nãoOK:
3484.                                       nãoOK.append(nó.Number)
3485.                               if itemj<= 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3486.                                   if EspessuraApoio<Espessura:
3487.                                       if itemj <
(Espessura/EspessuraApoio)*fcd:
3488.                                           if itemj < 3.3*fcd:
3489.                                               if nó.Number not in nóOk:
3490.
nóOk.append(nó.Number)
3491.                                           if itemj > 3.3*fcd:
3492.                                               if nó.Number not in
nãoOK:
3493.
nãoOK.append(nó.Number)
3494.                                       if itemj >
(Espessura/EspessuraApoio)*fcd:
3495.                                           if nó.Number not in nãoOK:
3496.                                               nãoOK.append(nó.Number)
3497.                                   if EspessuraApoio>=Espessura:
3498.                                       if nó.Number not in nóOk:
3499.                                           nóOk.append(nó.Number)
3500.
3501. PesopróprioCCT=[]
3502. Pesopróprioccct=[]
3503. gbrfgbthyn=[]
3504. nóscomforçaCCT=[]
3505. for i,item in enumerate(tensãoCCT):
3506.     if app.Project.Structure.Nodes.Exist(tensãoCCT[i][0][0]) == -1:
3507.         nó = app.Project.Structure.Nodes.Get(tensãoCCT[i][0][0])
3508.         gbrfgbthyn.append(nó.Number)
3509.         if nó.Z>0:
3510.             for m,itemm in enumerate(Nóseforçaas):
3511.                 if nó.Number == Nóseforçaas[m][0]:
3512.
nóscomforçaCCT.append([Nóseforçaas[m][0],Nóseforçaas[m][1]])
3513.
3514.             for n,itemn in enumerate(tensãoCCT):
3515.                 for o,itemo in enumerate(nóscomforçaCCT):
3516.                     fgnfgnmjum.append(nóscomforçaCCT[o][1])
3517.                     if nóscomforçaCCT[o][0] == tensãoCCT[n][0][0]:
```

```
3518.                           fgnfhgnmjum.append(tensãoCCT[n][0][0])
3519.
tensão_força=nóscomforçaCCT[o][1]/(EspessuraForça*LarguraForça)
3520.                           if abs(tensão_força) not in tensãoCCT[n][1]:
3521.
tensãoCCT[n][1].insert(0,abs(tensão_força))
3522.
3523.               for j,itemj in enumerate(tensãoCCT[i][1]):
3524.                   if tensãoCCT[i][1].Count==2:
3525.                       if itemj >0:
3526.                           if itemj > 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3527.                               if nó.Number not in nãoOK:
3528.                                   nãoOK.append(nó.Number)
3529.                           if itemj<= 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3530.                               if EspessuraForça<Espessura:
3531.                                   if itemj <
(Espessura/EspessuraForça)*fcd:
3532.                                       if itemj < 3.3*fcd:
3533.                                           if nó.Number not in nóOk:
3534.                                               nóOk.append(nó.Number)
3535.                                       if itemj > 3.3*fcd:
3536.                                           if nó.Number not in nãoOK:
3537.                                               nãoOK.append(nó.Number)
3538.                                   if itemj >
(Espessura/EspessuraForça)*fcd:
3539.                                       if nó.Number not in nãoOK:
3540.                                           nãoOK.append(nó.Number)
3541.                               if EspessuraForça>=Espessura:
3542.                                   if nó.Number not in nóOk:
3543.                                       nóOk.append(nó.Number)
3544.
3545.                   if tensãoCCT[i][1].Count==3:
3546.                       if itemj >0:
3547.                           if itemj > 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3548.                               if nó.Number not in nãoOK:
3549.                                   nãoOK.append(nó.Number)
3550.                           if itemj<= 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3551.                               if EspessuraForça<Espessura:
3552.                                   if itemj <=
(Espessura/EspessuraForça)*fcd:
3553.                                       if itemj < 3.3*fcd:
3554.                                           if nó.Number not in nóOk:
3555.                                               nóOk.append(nó.Number)
3556.                                       if itemj > 3.3*fcd:
3557.                                           if nó.Number not in nãoOK:
3558.                                               nãoOK.append(nó.Number)
3559.                                   if itemj >
(Espessura/EspessuraForça)*fcd:
3560.                                       if nó.Number not in nãoOK:
3561.                                           nãoOK.append(nó.Number)
3562.                               if EspessuraForça>=Espessura:
3563.                                   if nó.Number not in nóOk:
3564.                                       nóOk.append(nó.Number)
3565.
3566.                   if tensãoCCT[i][1].Count==4:
3567.                       if itemj>0:
```

```
3568.                           if itemj > 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3569.                               if nó.Number not in nãoOK:
3570.                                   nãoOK.append(nó.Number)
3571.                           if itemj<= 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3572.                               if EspessuraForça<Espessura:
3573.                                   if itemj <
(Espessura/EspessuraForça)*fcd:
3574.                                       if itemj < 3.3*fcd:
3575.                                           if nó.Number not in nóOk:
3576.                                               nóOk.append(nó.Number)
3577.                                       if itemj > 3.3*fcd:
3578.                                           if nó.Number not in nãoOK:
3579.                                               nãoOK.append(nó.Number)
3580.                                   if itemj >
(Espessura/EspessuraForça)*fcd:
3581.                                       if nó.Number not in nãoOK:
3582.                                           nãoOK.append(nó.Number)
3583.                               if EspessuraForça>=Espessura:
3584.                                   if nó.Number not in nóOk:
3585.                                       nóOk.append(nó.Number)
3586.
3587.                   if tensãoCCT[i][1].Count==5:
3588.                       if itemj>0:
3589.                           if itemj > 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3590.                               if nó.Number not in nãoOK:
3591.                                   nãoOK.append(nó.Number)
3592.                           if itemj<= 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3593.                               if EspessuraForça<Espessura:
3594.                                   if itemj <
(Espessura/EspessuraForça)*fcd:
3595.                                       if itemj < 3.3*fcd:
3596.                                           if nó.Number not in nóOk:
3597.                                               nóOk.append(nó.Number)
3598.                                       if itemj > 3.3*fcd:
3599.                                           if nó.Number not in nãoOK:
3600.                                               nãoOK.append(nó.Number)
3601.                                   if itemj >
(Espessura/EspessuraForça)*fcd:
3602.                                       if nó.Number not in nãoOK:
3603.                                           nãoOK.append(nó.Number)
3604.                               if EspessuraForça>=Espessura:
3605.                                   if nó.Number not in nóOk:
3606.                                       nóOk.append(nó.Number)
3607.                   if tensãoCCT[i][1].Count==6:
3608.                       if itemj>0:
3609.                           if itemj > 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3610.                               if nó.Number not in nãoOK:
3611.                                   nãoOK.append(nó.Number)
3612.                           if itemj<= 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3613.                               if EspessuraForça<Espessura:
3614.                                   if itemj <
(Espessura/EspessuraForça)*fcd:
3615.                                       if itemj < 3.3*fcd:
3616.                                           if nó.Number not in nóOk:
```

```
3617.                                                   nóOk.append(nó.Number)
3618.                                               if itemj > 3.3*fcd:
3619.                                                   if nó.Number not in nãoOK:
3620.                                                       nãoOK.append(nó.Number)
3621.                                           if itemj >
(Espessura/EspessuraForça)*fcd:
3622.                                               if nó.Number not in nãoOK:
3623.                                                   nãoOK.append(nó.Number)
3624.                                       if EspessuraForça>=Espessura:
3625.                                           if nó.Number not in nóOk:
3626.                                               nóOk.append(nó.Number)
3627.             if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
3628.                 if nó.Z==0:
3629.                     for j in range(1, cases.Count+1):
3630.                         casos= app.Project.Structure.Cases.Get(j)
3631.                         forças1 = reações.Value(nó.Number, casos.Number)
3632.                         fZ=forças1.FZ
3633.                         Pesopróprioccт.append(fZ)
3634.                     Peso=0
3635.                     for l in Pesopróprioccт:
3636.                         Peso=Peso+l
3637.                     PesopróprioCCT.append(Peso)
3638.
tensão_apoio=PesopróprioCCT[0]/(EspessuraApoio*LarguraApoio)
3639.                     tensãoCCT[i][1].append(tensão_apoio)
3640.                     for j,itemj in enumerate(tensãoCCT[i][1]):
3641.                         if tensãoCCT[i][1].Count==3:
3642.                             if itemj>0:
3643.                                 if itemj > 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3644.                                     if nó.Number not in nãoOK:
3645.                                         nãoOK.append(nó.Number)
3646.                                 if itemj<= 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3647.                                     if EspessuraApoio<Espessura:
3648.                                         if itemj <
(Espessura/EspessuraApoio)*fcd:
3649.                                             if itemj < 3.3*fcd:
3650.                                                 if nó.Number not in nóOk:
3651.
nóOk.append(nó.Number)
3652.                                             if itemj > 3.3*fcd:
3653.                                                 if nó.Number not in
nãoOK:
3654.
nãoOK.append(nó.Number)
3655.                                         if itemj >
(Espessura/EspessuraApoio)*fcd:
3656.                                             if nó.Number not in nãoOK:
3657.                                                 nãoOK.append(nó.Number)
3658.                                     if EspessuraApoio>=Espessura:
3659.                                         if nó.Number not in nóOk:
3660.                                             nóOk.append(nó.Number)
3661.
3662.                         if tensãoCCT[i][1].Count==4:
3663.                             if itemj>0:
3664.                                 if itemj > 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3665.                                     if nó.Number not in nãoOK:
3666.                                         nãoOK.append(nó.Number)
```

```
3667.                                 if itemj<= 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3668.                                     if EspessuraApoio<Espessura:
3669.                                         if itemj <
(Espessura/EspessuraApoio)*fcd:
3670.                                             if itemj < 3.3*fcd:
3671.                                                 if nó.Number not in nóOk:
3672.
nóOk.append(nó.Number)
3673.                                             if itemj > 3.3*fcd:
3674.                                                 if nó.Number not in
nãoOK:
3675.
nãoOK.append(nó.Number)
3676.                                         if itemj >
(Espessura/EspessuraApoio)*fcd:
3677.                                             if nó.Number not in nãoOK:
3678.                                                 nãoOK.append(nó.Number)
3679.                                     if EspessuraApoio>=Espessura:
3680.                                         if nó.Number not in nóOk:
3681.                                             nóOk.append(nó.Number)
3682.
3683.                         if tensãoCCT[i][1].Count==5:
3684.                             if itemj>0:
3685.                                 if itemj > 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3686.                                     if nó.Number not in nãoOK:
3687.                                         nãoOK.append(nó.Number)
3688.                                 if itemj<= 0.85*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3689.                                     if EspessuraApoio<Espessura:
3690.                                         if itemj <
(Espessura/EspessuraApoio)*fcd:
3691.                                             if itemj < 3.3*fcd:
3692.                                                 if nó.Number not in nóOk:
3693.
nóOk.append(nó.Number)
3694.                                             if itemj > 3.3*fcd:
3695.                                                 if nó.Number not in
nãoOK:
3696.
nãoOK.append(nó.Number)
3697.                                         if itemj >
(Espessura/EspessuraApoio)*fcd:
3698.                                             if nó.Number not in nãoOK:
3699.                                                 nãoOK.append(nó.Number)
3700.                                     if EspessuraApoio>=Espessura:
3701.                                         if nó.Number not in nóOk:
3702.                                             nóOk.append(nó.Number)
3703.                         if tensãoCCT[i][1].Count==6:
3704.                             if itemj>0:
3705.                                 if itemj > 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3706.                                     if nó.Number not in nãoOK:
3707.                                         nãoOK.append(nó.Number)
3708.                                 if itemj<= 0.72*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3709.                                     if EspessuraApoio<Espessura:
3710.                                         if itemj <
(Espessura/EspessuraApoio)*fcd:
```

```
3711.                                                   if itemj < 3.3*fcd:
3712.                                                       if nó.Number not in nóOk:
3713.
nóOk.append(nó.Number)
3714.                                                   if itemj > 3.3*fcd:
3715.                                                       if nó.Number not in
nãoOK:
3716.
nãoOK.append(nó.Number)
3717.                                               if itemj >
(Espessura/EspessuraApoio)*fcd:
3718.                                                   if nó.Number not in nãoOK:
3719.                                                       nãoOK.append(nó.Number)
3720.                                           if EspessuraApoio>=Espessura:
3721.                                               if nó.Number not in nóOk:
3722.                                                   nóOk.append(nó.Number)
3723.
3724. PesopróprioCTTeTTT=[]
3725. Pesopróprioccttettt=[]
3726. nóscomforçaCTTeTTT=[]
3727. fgnfgnmjummm=[]
3728. fgnfhgnmjummm=[]
3729. for i,item in enumerate(tensãoCTTeTTT):
3730.     if app.Project.Structure.Nodes.Exist(tensãoCTTeTTT[i][0][0]) == -
1:
3731.         nó = app.Project.Structure.Nodes.Get(tensãoCTTeTTT[i][0][0])
3732.         if nó.Z>0:
3733.             for m,itemm in enumerate(Nóseforçaas):
3734.                 if nó.Number == Nóseforçaas[m][0]:
3735.
nóscomforçaCTTeTTT.append([Nóseforçaas[m][0],Nóseforçaas[m][1]])
3736.
3737.             for n,itemn in enumerate(tensãoCTTeTTT):
3738.                 for o,itemo in enumerate(nóscomforçaCTTeTTT):
3739.                     fgnfgnmjum.append(nóscomforçaCTTeTTT[o][1])
3740.                     if nóscomforçaCTTeTTT[o][0] ==
tensãoCTTeTTT[n][0][0]:
3741.                         fgnfhgnmjum.append(tensãoCTTeTTT[n][0][0])
3742.
tensão_força=nóscomforçaCTTeTTT[o][1]/(EspessuraForça*LarguraForça)
3743.                         if abs(tensão_força) not in
tensãoCTTeTTT[n][1]:
3744.
tensãoCTTeTTT[n][1].insert(0,abs(tensão_força))
3745.
3746.             for j,itemj in enumerate(tensãoCTTeTTT[i][1]):
3747.                 if tensãoCTTeTTT[i][1].Count==3:
3748.                     if itemj>0:
3749.                         if itemj > 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3750.                             if nó.Number not in nãoOK:
3751.                                 nãoOK.append(nó.Number)
3752.                         if itemj <= 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3753.                             if EspessuraForça<Espessura:
3754.                                 if itemj <
(Espessura/EspessuraForça)*fcd:
3755.                                     if itemj < 3.3*fcd:
3756.                                         if nó.Number not in nóOk:
3757.                                             nóOk.append(nó.Number)
```

```
3758.                                       if itemj > 3.3*fcd:
3759.                                           if nó.Number not in nóOk:
3760.                                               nãoOK.append(nó.Number)
3761.                                   if itemj >
(Espessura/EspessuraForça)*fcd:
3762.                                       if nó.Number not in nóOk:
3763.                                           nãoOK.append(nó.Number)
3764.                               if EspessuraForça>=Espessura:
3765.                                   if nó.Number not in nóOk:
3766.                                       nóOk.append(nó.Number)
3767.
3768.                   if tensãoCTTeTTT[i][1].Count==3:
3769.                       if itemj>0:
3770.                           if itemj > 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3771.                               if nó.Number not in nãoOK:
3772.                                   nãoOK.append(nó.Number)
3773.                           if itemj<= 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3774.                               if EspessuraForça<Espessura:
3775.                                   if itemj <
(Espessura/EspessuraForça)*fcd:
3776.                                       if itemj < 3.3*fcd:
3777.                                           if nó.Number not in nóOk:
3778.                                               nóOk.append(nó.Number)
3779.                                       if itemj > 3.3*fcd:
3780.                                           if nó.Number not in nóOk:
3781.                                               nãoOK.append(nó.Number)
3782.                                   if itemj >
(Espessura/EspessuraForça)*fcd:
3783.                                       if nó.Number not in nóOk:
3784.                                           nãoOK.append(nó.Number)
3785.                               if EspessuraForça>=Espessura:
3786.                                   if nó.Number not in nóOk:
3787.                                       nóOk.append(nó.Number)
3788.
3789.                   if tensãoCTTeTTT[i][1].Count==4:
3790.                       if itemj>0:
3791.                           if itemj > 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3792.                               if nó.Number not in nãoOK:
3793.                                   nãoOK.append(nó.Number)
3794.                           if itemj<= 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3795.                               if EspessuraForça<Espessura:
3796.                                   if itemj <
(Espessura/EspessuraForça)*fcd:
3797.                                       if itemj < 3.3*fcd:
3798.                                           if nó.Number not in nóOk:
3799.                                               nóOk.append(nó.Number)
3800.                                       if itemj > 3.3*fcd:
3801.                                           if nó.Number not in nóOk:
3802.                                               nãoOK.append(nó.Number)
3803.                                   if itemj >
(Espessura/EspessuraForça)*fcd:
3804.                                       if nó.Number not in nóOk:
3805.                                           nãoOK.append(nó.Number)
3806.                               if EspessuraForça>=Espessura:
3807.                                   if nó.Number not in nóOk:
3808.                                       nóOk.append(nó.Number)
```

```
3809.
3810.                   if tensãoCTTeTTT[i][1].Count==5:
3811.                       if itemj>0:
3812.                           if itemj > 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3813.                               if nó.Number not in nãoOK:
3814.                                   nãoOK.append(nó.Number)
3815.                           if itemj<= 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3816.                               if EspessuraForça<Espessura:
3817.                                   if itemj <
(Espessura/EspessuraForça)*fcd:
3818.                                       if itemj < 3.3*fcd:
3819.                                           if nó.Number not in nóOk:
3820.                                               nóOk.append(nó.Number)
3821.                                       if itemj > 3.3*fcd:
3822.                                           if nó.Number not in nóOk:
3823.                                               nãoOK.append(nó.Number)
3824.                                   if itemj >
(Espessura/EspessuraForça)*fcd:
3825.                                       if nó.Number not in nóOk:
3826.                                           nãoOK.append(nó.Number)
3827.                               if EspessuraForça>=Espessura:
3828.                                   if nó.Number not in nóOk:
3829.                                       nóOk.append(nó.Number)
3830.                   if tensãoCTTeTTT[i][1].Count==6:
3831.                       if itemj>0:
3832.                           if itemj > 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3833.                               if nó.Number not in nãoOK:
3834.                                   nãoOK.append(nó.Number)
3835.                           if itemj<= 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3836.                               if EspessuraForça<Espessura:
3837.                                   if itemj <
(Espessura/EspessuraForça)*fcd:
3838.                                       if itemj < 3.3*fcd:
3839.                                           if nó.Number not in nóOk:
3840.                                               nóOk.append(nó.Number)
3841.                                       if itemj > 3.3*fcd:
3842.                                           if nó.Number not in nóOk:
3843.                                               nãoOK.append(nó.Number)
3844.                                   if itemj >
(Espessura/EspessuraForça)*fcd:
3845.                                       if nó.Number not in nóOk:
3846.                                           nãoOK.append(nó.Number)
3847.                               if EspessuraForça>=Espessura:
3848.                                   if nó.Number not in nóOk:
3849.                                       nóOk.append(nó.Number)
3850.           if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
3851.               if nó.Z==0:
3852.                   for j in range(1, cases.Count+1):
3853.                       casos= app.Project.Structure.Cases.Get(j)
3854.                       forças1 = reações.Value(nó.Number, casos.Number)
3855.                       fZ=forças1.FZ
3856.                       Pesopróprioccttettt.append(fZ)
3857.                   Peso=0
3858.                   for l in Pesopróprioccttettt:
3859.                       Peso=Peso+l
3860.                   PesopróprioCTTeTTT.append(Peso)
```

```
3861.
tensão_apoio=PesopróprioCTTeTTT[0]/(EspessuraApoio*LarguraApoio)
3862.                   tensãoCTTeTTT[i][1].append(tensão_apoio)
3863.                   for j,itemj in enumerate(tensãoCTTeTTT[i][1]):
3864.                       if tensãoCTTeTTT[i][1].Count==3:
3865.                           if itemj>0:
3866.                               if itemj > 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3867.                                   if nó.Number not in nãoOK:
3868.                                       nãoOK.append(nó.Number)
3869.                               if itemj<= 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3870.                                   if EspessuraApoio<Espessura:
3871.                                       if itemj <
(Espessura/EspessuraApoio)*fcd:
3872.                                           if itemj < 3.3*fcd:
3873.                                               if nó.Number not in nóOk:
3874.
nóOk.append(nó.Number)
3875.                                           if itemj > 3.3*fcd:
3876.                                               if nó.Number not in nóOk:
3877.
nãoOK.append(nó.Number)
3878.                                       if itemj >
(Espessura/EspessuraApoio)*fcd:
3879.                                           if nó.Number not in nóOk:
3880.                                               nãoOK.append(nó.Number)
3881.                                   if EspessuraApoio>=Espessura:
3882.                                       if nó.Number not in nóOk:
3883.                                           nóOk.append(nó.Number)
3884.
3885.                       if tensãoCTTeTTT[i][1].Count==4:
3886.                           if itemj>0:
3887.                               if itemj > 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3888.                                   if nó.Number not in nãoOK:
3889.                                       nãoOK.append(nó.Number)
3890.                               if itemj<= 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3891.                                   if EspessuraApoio<Espessura:
3892.                                       if itemj <
(Espessura/EspessuraApoio)*fcd:
3893.                                           if itemj < 3.3*fcd:
3894.                                               if nó.Number not in nóOk:
3895.
nóOk.append(nó.Number)
3896.                                           if itemj > 3.3*fcd:
3897.                                               if nó.Number not in nóOk:
3898.
nãoOK.append(nó.Number)
3899.                                       if itemj >
(Espessura/EspessuraApoio)*fcd:
3900.                                           if nó.Number not in nóOk:
3901.                                               nãoOK.append(nó.Number)
3902.                                   if EspessuraApoio>=Espessura:
3903.                                       if nó.Number not in nóOk:
3904.                                           nóOk.append(nó.Number)
3905.
3906.                       if tensãoCTTeTTT[i][1].Count==5:
3907.                           if itemj>0:
```

```
3908.                                   if itemj > 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3909.                                       if nó.Number not in nãoOK:
3910.                                           nãoOK.append(nó.Number)
3911.                                   if itemj<= 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3912.                                       if EspessuraApoio<Espessura:
3913.                                           if itemj <
(Espessura/EspessuraApoio)*fcd:
3914.                                               if itemj < 3.3*fcd:
3915.                                                   if nó.Number not in nóOk:
3916.
nóOk.append(nó.Number)
3917.                                               if itemj > 3.3*fcd:
3918.                                                   if nó.Number not in nóOk:
3919.
nãoOK.append(nó.Number)
3920.                                           if itemj >
(Espessura/EspessuraApoio)*fcd:
3921.                                               if nó.Number not in nóOk:
3922.                                                   nãoOK.append(nó.Number)
3923.                                       if EspessuraApoio>=Espessura:
3924.                                           if nó.Number not in nóOk:
3925.                                               nóOk.append(nó.Number)
3926.                           if tensãoCTTeTTT[i][1].Count==6:
3927.                               if itemj>0:
3928.                                   if itemj > 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3929.                                       if nó.Number not in nãoOK:
3930.                                           nãoOK.append(nó.Number)
3931.                                   if itemj<= 0.6*(1-
(((fcd*1.5)/1000000)/250))*fcd:
3932.                                       if EspessuraApoio<Espessura:
3933.                                           if itemj <
(Espessura/EspessuraApoio)*fcd:
3934.                                               if itemj < 3.3*fcd:
3935.                                                   if nó.Number not in nóOk:
3936.
nóOk.append(nó.Number)
3937.                                               if itemj > 3.3*fcd:
3938.                                                   if nó.Number not in nóOk:
3939.
nãoOK.append(nó.Number)
3940.                                           if itemj >
(Espessura/EspessuraApoio)*fcd:
3941.                                               if nó.Number not in nóOk:
3942.                                                   nãoOK.append(nó.Number)
3943.                                       if EspessuraApoio>=Espessura:
3944.                                           if nó.Number not in nóOk:
3945.                                               nóOk.append(nó.Number)
3946. Aprovados=[]
3947. for i,item in enumerate(nóOk):
3948.     if item not in nãoOK:
3949.         if item not in Aprovados:
3950.             Aprovados.append(item)
3951. Reprovados=[]
3952. for i,item in enumerate(nãoOK):
3953.     if item not in Reprovados:
3954.         Reprovados.append(item)
3955.
```

```
#AVALIAR ESTRUTURA
weight_scoreNÓ=0
if Reprovados.Count>=0:
    weight_scoreNÓ=Reprovados.Count*(penalty_weight_max*5)

weight_scoreÁreaAço=0
Área_mínima_Aço=(0.04*(Espessura*1000)*((altura*1000)-
(cobrimento*1000)))
if (ÁreaBarrasTracionadas*1000*1000)<Área_mínima_Aço:
    weight_scoreÁreaAço=penalty_weight_max
woeirfsoigwg=[]
for i,item in enumerate(spkdmvsk):
    if spkdmvsk[i][0][1] >= 0.85*(1-(((fcd*1.5)/1000000)/250))*fcd:
        woeirfsoigwg.append(spkdmvsk[i][0][0])
Score_Bielas=0
if woeirfsoigwg.Count>=0:
    Score_Bielas=woeirfsoigwg.Count*(penalty_weight_max*2)

# CALCULAR
app.Project.ViewMngr.Refresh()
app.Project.CalcEngine.GenerateModel()
app.Project.CalcEngine.Calculate()
app.Project.ViewMngr.Refresh()
#PONTOS INICIAL E FINAL DE CADA BARRA RESULTANTE
pontofinalbarras=[]
pontoinicialbarras=[]
for i in range(1, bars.Count+800):
    if app.Project.Structure.Bars.Exist(i) == -1:
        barra = app.Project.Structure.Bars.Get(i)
        startnode = app.Project.Structure.Nodes.Get(barra.StartNode)
        endnode = app.Project.Structure.Nodes.Get(barra.EndNode)
        pontoinicialbarras.append(startnode.Number)
        pontofinalbarras.append(endnode.Number)

#EXTRAIR REAÇÃO PARA OBTER PESO-PRÓPRIO
Pesopróprio=[]
PesopróprioKG=[]
PesopróprioKG1=[]
app.Project.CalcEngine.Calculate()
for i in range(1, nodes.Count+800):
    if app.Project.Structure.Nodes.Exist(i) == -1:
        nó = app.Project.Structure.Nodes.Get(i)
        if nó.HasLabel(IRobotLabelType.I_LT_SUPPORT) != 0:
            casos= app.Project.Structure.Cases.Get(3)
            forças1 = reações.Value(nó.Number, casos.Number)
            fZ=forças1.FZ
            Pesopróprio.append(sum([fZ]))

PesopróprioKG.append(sum(Pesopróprio))
for bf,item in enumerate(PesopróprioKG):
    PesopróprioKG1.append(round((item/9.81),5))

#EXTRAIR RESULTADOS
saida1=[]
app.Project.ViewMngr.Refresh()
for i in range(1, bars.Count+1000):
    if app.Project.Structure.Bars.Exist(i) == -1:
        barra = app.Project.Structure.Bars.Get(i).Number
        for j in range(1, cases.Count+1):
            casos= app.Project.Structure.Cases.Get(j).Number
```

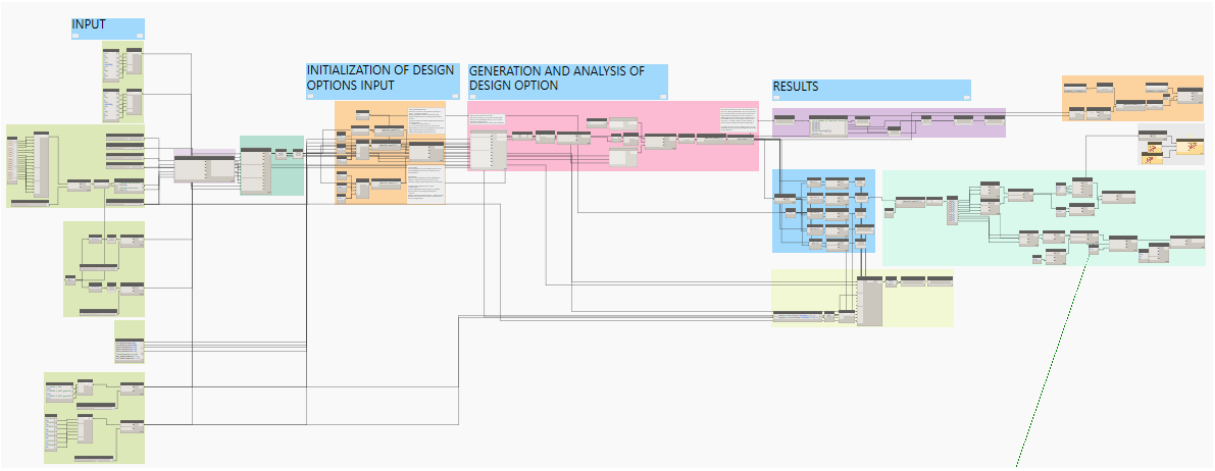
```
4015.             forças1 = stress.Value(barra, casos, 1.0)
4016.             fx1=forças1.FXSX
4017.             forças2 = stress.Value(barra, casos, 0.0)
4018.             fx2=forças2.FXSX
4019.             saida1.append(max(fx2,fx1))
4020.
4021. splitedlist1=[]
4022. comprimentoLista=len(saida1)
4023. sequencia=comprimentoLista/numerodecarregamentos
4024. for i in range(sequencia):
4025.     start=(i*comprimentoLista/sequencia)
4026.     end=((i+1)*comprimentoLista/sequencia)
4027.     splitedlist1.append(round(sum(saida1[start:end]),2))
4028. #SEPARAR BARRAS
4029. Nóss=[]
4030. Barrras=[]
4031. for ak in range(1, nodes.Count+800):
4032.     if app.Project.Structure.Nodes.Exist(ak) == -1:
4033.         nós = app.Project.Structure.Nodes.Get(ak)
4034.         Nóss.append(nós.Number)
4035. for al in range(1, bars.Count+800):
4036.     if app.Project.Structure.Bars.Exist(al) == -1:
4037.         barras = app.Project.Structure.Bars.Get(al)
4038.         Barrras.append(barras.Number)
4039. Barrras1=[]
4040. for am,item in enumerate(Barrras):
4041.     if item not in Barrras1:
4042.         Barrras1.append(item)
4043. listabarravertical13=[]
4044. listabarrahorizontal13=[]
4045. listabarradiagonal13=[]
4046. for ao in Barrras1:
4047.     if app.Project.Structure.Bars.Exist(ao) == -1:
4048.         barra = app.Project.Structure.Bars.Get(ao)
4049.         startnode = app.Project.Structure.Nodes.Get(barra.StartNode)
4050.         endnode = app.Project.Structure.Nodes.Get(barra.EndNode)
4051.         Y0=startnode.Y
4052.         Z0=startnode.Z
4053.         Y1=endnode.Y
4054.         Z1=endnode.Z
4055.         if Y0==Y1:
4056.             listabarravertical13.append(barra.Number)
4057.         if Z0==Z1:
4058.             listabarrahorizontal13.append(barra.Number)
4059.         if Z0!=Z1:
4060.             if Y0!=Y1:
4061.                 listabarradiagonal13.append(barra.Number)
4062. #EXTRAIR RESULTADOS BARRAS HORIZONTAIS
4063. valoresbarrashorizontais=[]
4064. for ax,item in enumerate(listabarrahorizontal13):
4065.     barrahorizontal = splitedlist1.__getitem__(ax)
4066.     valoresbarrashorizontais.append(barrahorizontal)
4067. compressãohorizontal=[]
4068. traçãohorizontal=[]
4069. for az,item in enumerate(valoresbarrashorizontais):
4070.     if item > 0:
4071.         compressãohorizontal.append(item)
4072.     else:
4073.         traçãohorizontal.append(item)
4074. #EXTRAIR RESULTADOS BARRAS VERTICAIS
```

```
4075. valoresbarrasverticais=[]
4076. for aw,item in enumerate(listabarravertical13):
4077.     barravertical =
splitedlist1.__getitem__(aw+listabarrahorizontal13.Count)
4078.     valoresbarrasverticais.append(barravertical)
4079. compressãovertical=[]
4080. traçãovertical=[]
4081. for ba,item in enumerate(valoresbarrasverticais):
4082.     if item > 0:
4083.         compressãovertical.append(item)
4084.     else:
4085.         traçãovertical.append(item)
4086.
4087. #EXTRAIR RESULTADOS BARRAS DIAGONAIS
4088. valoresbarrasdiagonais=[]
4089. for bc,item in enumerate(listabarradiagonal13):
4090.     barradiagonals =
splitedlist1.__getitem__(bc+listabarrahorizontal13.Count+listabarravertical
13.Count)
4091.     valoresbarrasdiagonais.append(barradiagonals)
4092. compressãodiagonal=[]
4093. traçãodiagonal=[]
4094. for bd,item in enumerate(valoresbarrasdiagonais):
4095.     if item > 0:
4096.         compressãodiagonal.append(item)
4097.     else:
4098.         traçãodiagonal.append(item)
4099.
4100. #AVALIAR ESTRUTURA TENSÃO ADMISSÍVEL
4101. weight_scoreC=PesopróprioKG1[0]
4102. weight_scoreT=PesopróprioKG1[0]
4103.
4104. for i in compressãohorizontal:
4105.     ratio = abs(i) / AllowableStressCompressãoH
4106.     if ratio >= max_ratio:
4107.         weight_scoreC=weight_scoreC+penalty_weight_max
4108.     if ratio <= min_ratio:
4109.         weight_scoreC=weight_scoreC+penalty_weight_min
4110.     else:
4111.         weight_scoreC=weight_scoreC
4112. for s in traçãohorizontal:
4113.     ratio = abs(s) / AllowableStressTraçãoH
4114.     if ratio >= max_ratio:
4115.         weight_scoreC=weight_scoreC+penalty_weight_max
4116.     if ratio <= min_ratio:
4117.         weight_scoreC=weight_scoreC+penalty_weight_min
4118.     else:
4119.         weight_scoreC=weight_scoreC
4120. for i in compressãovertical:
4121.     ratio = abs(i) / AllowableStressCompressãoV
4122.     if ratio >= max_ratio:
4123.         weight_scoreC=weight_scoreC+penalty_weight_max
4124.     if ratio <= min_ratio:
4125.         weight_scoreC=weight_scoreC+penalty_weight_min
4126.     else:
4127.         weight_scoreC=weight_scoreC
4128. for s in traçãovertical:
4129.     ratio = abs(s) / AllowableStressTraçãoV
4130.     if ratio >= max_ratio:
4131.         weight_scoreC=weight_scoreC+penalty_weight_max
```

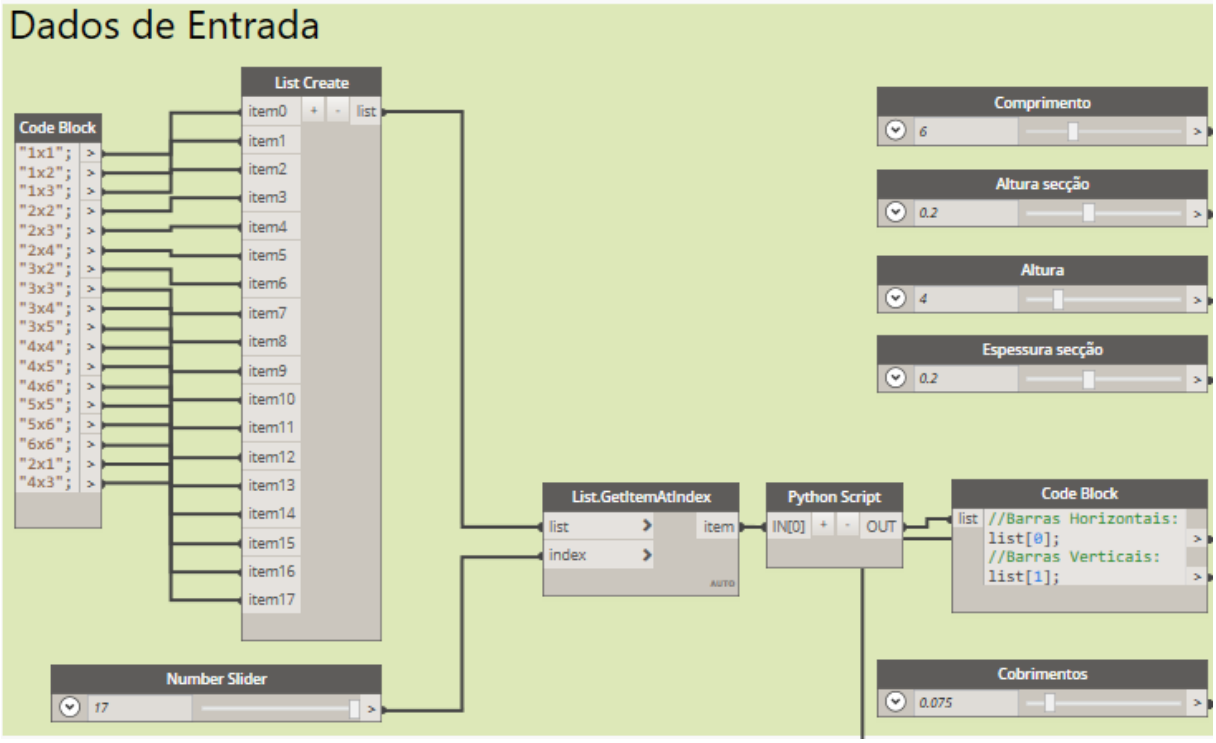
```
4132.     if ratio <= min_ratio:
4133.         weight_scoreC=weight_scoreC+penalty_weight_min
4134.     else:
4135.         weight_scoreC=weight_scoreC
4136. for i in compressãodiagonal:
4137.     ratio = abs(i) / AllowableStressCompressãoD
4138.     if ratio >= max_ratio:
4139.         weight_scoreC=weight_scoreC+penalty_weight_max
4140.     if ratio <= min_ratio:
4141.         weight_scoreC=weight_scoreC+penalty_weight_min
4142.     else:
4143.         weight_scoreC=weight_scoreC
4144. for s in traçãodiagonal:
4145.     ratio = abs(s) / AllowableStressTraçãoD
4146.     if ratio >= max_ratio:
4147.         weight_scoreC=weight_scoreC+penalty_weight_max
4148.     if ratio <= min_ratio:
4149.         weight_scoreC=weight_scoreC+penalty_weight_min
4150.     else:
4151.         weight_scoreC=weight_scoreC
4152. #PREPARAR RESULTADOS PARA EXPORTAÇÃO
4153. #scorefinal=float(int(weight_scoreC)+int(weight_scoreT)-
int(PesopróprioKG1[0]))
4154. scorefinal=float(weight_scoreC+weight_scoreT-PesopróprioKG1[0])
4155. Score_Avaliação_nós=int(weight_scoreNÓ)
4156. weight_scoreÁreaAço=int(weight_scoreÁreaAço)
4157. weight_scoreBielas=int(Score_Bielas)
4158. PontuaçãoFinal=scorefinal+weight_scoreÁreaAço+Score_Avaliação_nós+wei
ght_scoreBielas
4159. #SALVAR
4160. p=p+1
4161. i=(round(Espessura,5)*100)
4162. j=(round(altura,5)*100)
4163. l=(round(Taxa_aço,5))
4164. app.Project.SaveAs(r"C:\Users\mateu\Desktop\Nova
pasta\Base_"+str(i)+"x"+str(j)+"mm"+str(l)+".rtd")
4165. # Atribua a sua saída para a variável OUT.
4166. OUT = PontuaçãoFinal,PesopróprioKG1[0]
```

B. VISUALIZAÇÃO DO *WORKFLOW*

1. *WORKFLOW* COMPLETO

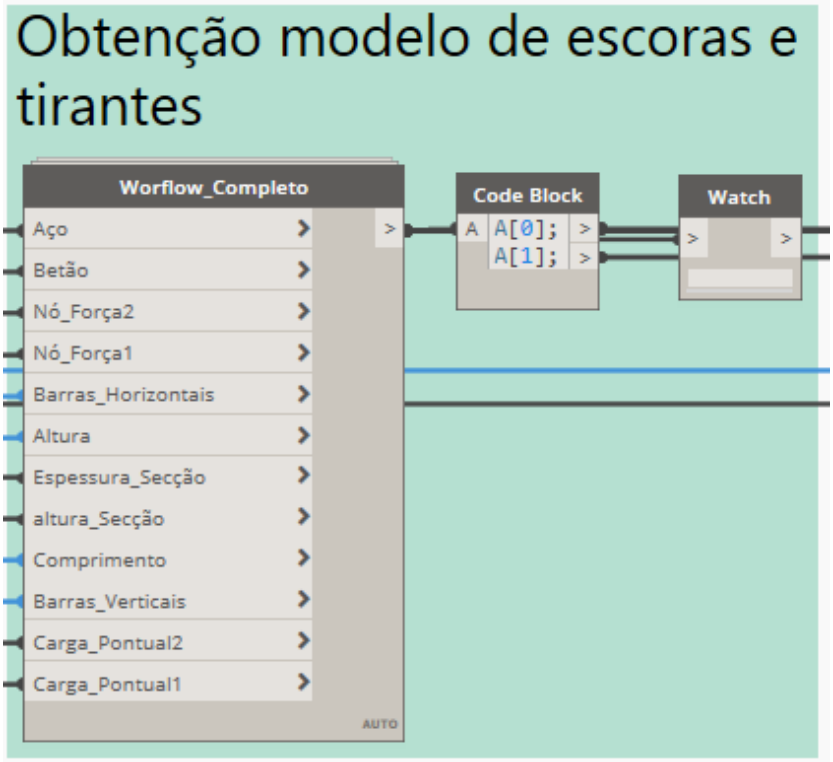


2. EXEMPLO DE DADOS DE ENTRADA

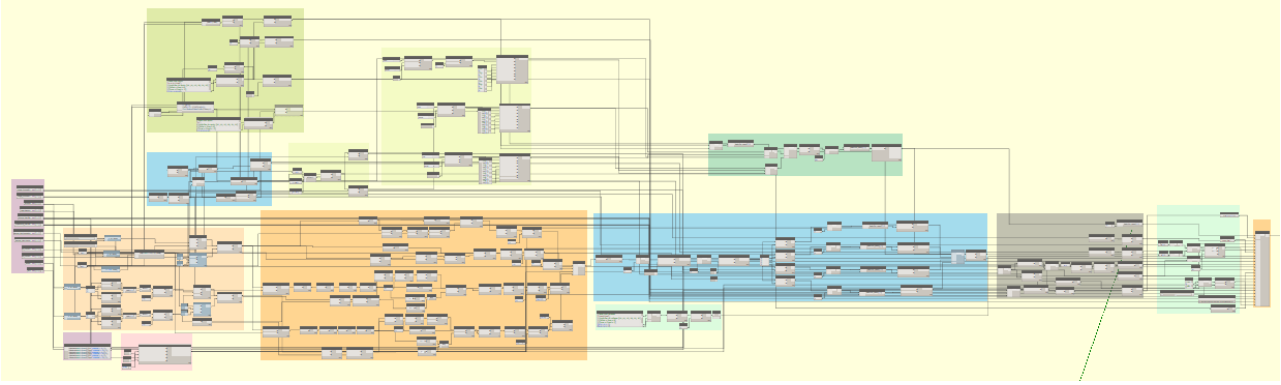


3. NÓ DE VERIFICAÇÃO

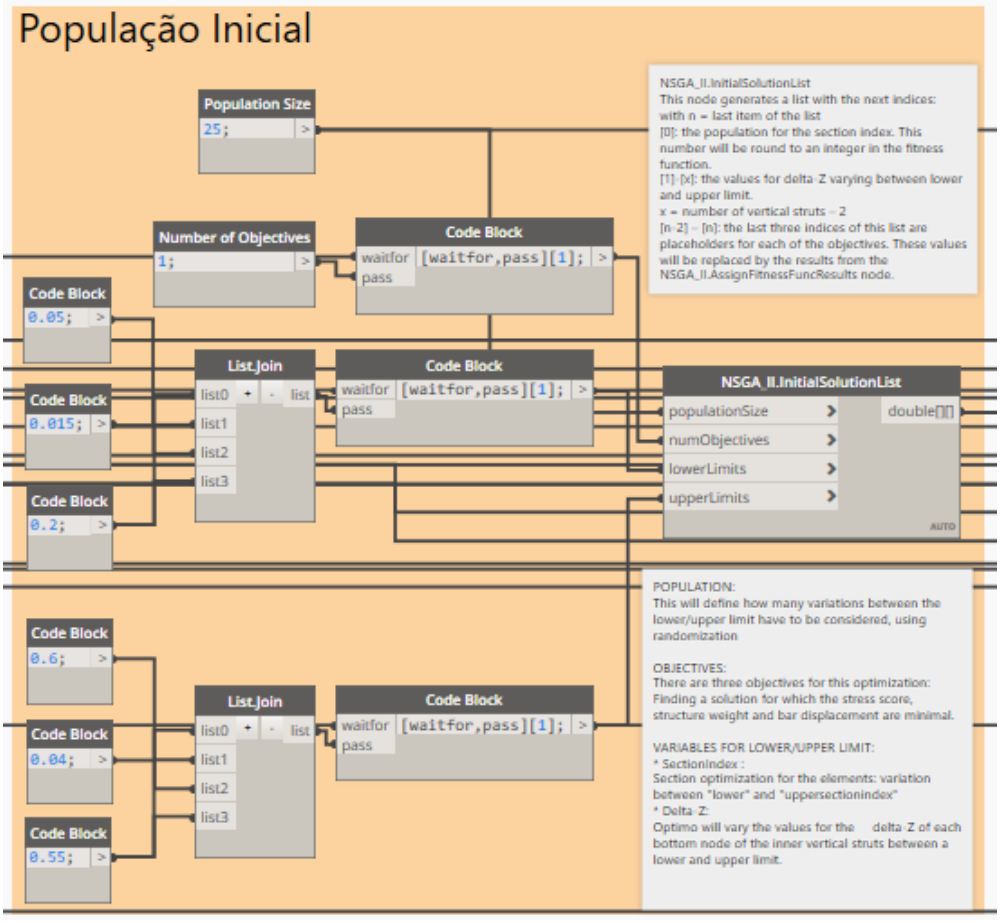
4. NÓ PARA OBTENÇÃO MODELO DE ESCORAS E TIRANTES (EXTERNO)



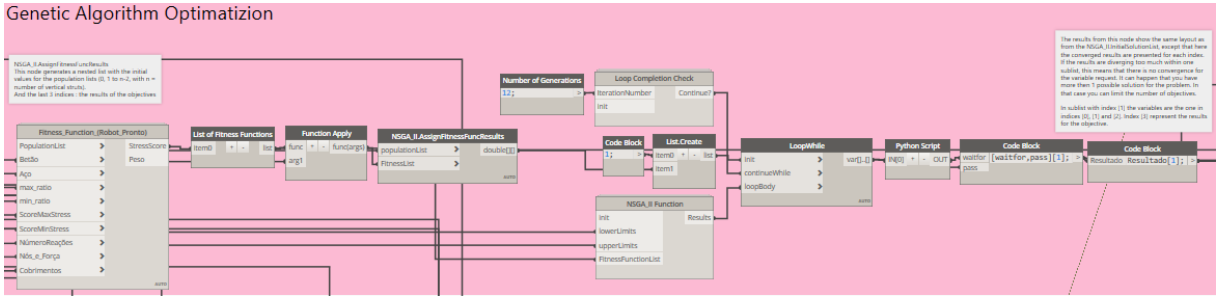
5. NÓ PARA OBTENÇÃO MODELO DE ESCORAS E TIRANTES (INTERNO)



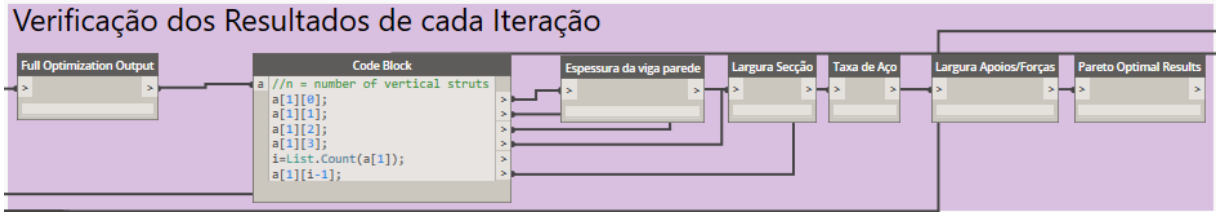
6. NÓS PARA CRIAÇÃO DA POPULAÇÃO INICIAL OU “PAIS” (PACOTE OPTIMO)



7. NÓS PARA CRIAÇÃO DA POPULAÇÃO DE "FILHOS" (PACOTE OPTIMO)

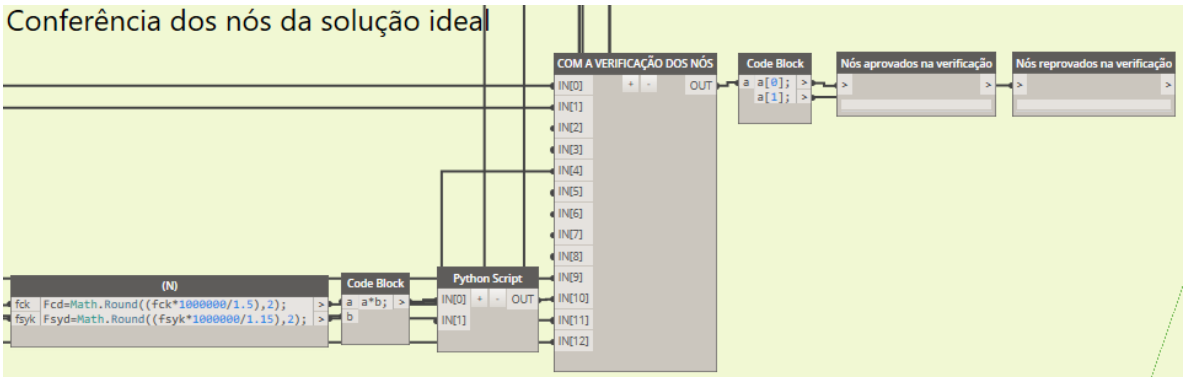


8. NÓS PARA VERIFICAÇÃO DOS RESULTADOS DE TODAS AS ITERAÇÕES

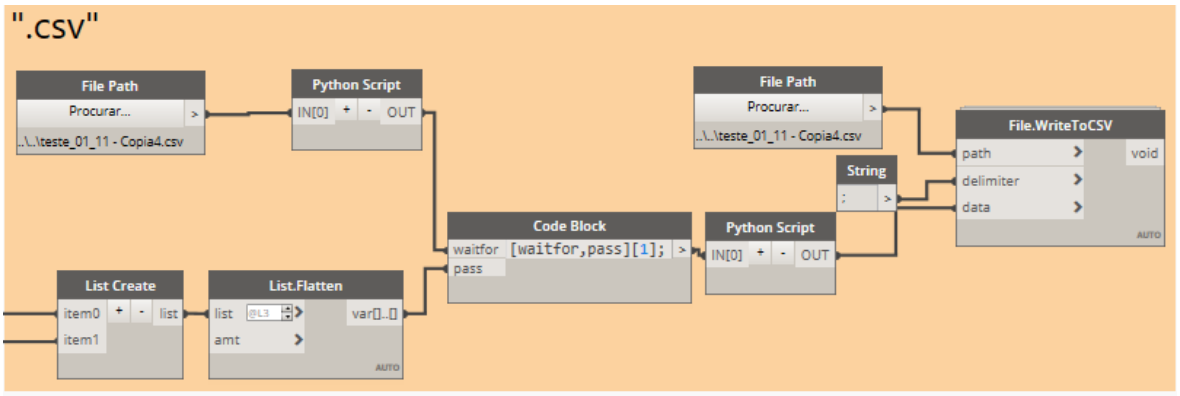


9. NÓS PARA VISUALIZAÇÃO DO RESULTADO FINAL

10. NÓS PARA CONFERÊNCIA DOS NÓS DA SOLUÇÃO IDEAL



11. NÓS PARA SALVAR OS RESULTADOS DE CADA ITERAÇÃO EM UM ARQUIVO ".CSV"



12. ENVIO DO RESULTADO PARA O REVIT