

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

**ANÁLISE DE EFICIÊNCIA BASEADA EM MULTICRITÉRIO
APLICADA À UNIDADE DE PROCESSAMENTO DE GÁS
NATURAL**

DISSERTAÇÃO DE MESTRADO

Felipe Malichovsky Severo

Porto Alegre

2021

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

ANÁLISE DE EFICIÊNCIA BASEADA EM MULTICRITÉRIO APLICADA À UNIDADE DE PROCESSAMENTO DE GÁS NATURAL

Felipe Malichovsky Severo

Dissertação de Mestrado apresentada como requisito parcial para obtenção do título de Mestre em Engenharia

Área de concentração: Pesquisa e Desenvolvimento de Processos

Linha de Pesquisa: Engenharia de Sistemas – Projeto, Modelagem, Controle e Otimização de Processos

Orientadores:

Prof. Dr. Jorge Otávio Trierweiler

Prof. Dr. Pedro Rafael Bolognese Fernandes

Porto Alegre

2021

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA QUÍMICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

A Comissão Examinadora, abaixo assinada, aprova a Dissertação *Análise de Eficiência Baseado em Multicritério Aplicada à Unidade de Processamento de Gás Natural*, elaborada por Felipe Malichovsky Severo como requisito parcial para obtenção do Grau de Mestre em Engenharia.

Comissão Examinadora:

Profa. Dra. Luciane Ferreira Trierweiler

Profa. Dra. Paula Bettio Staudt

Prof. Dr. Gustavo Mendes Platt

Resumo

Diversos estudos são realizados com o objetivo de propor modificações em plantas industriais visando obter maior eficiência operacional, o que significa um menor custo econômico. Foco especial se aplica às colunas de destilação, pois estes equipamentos são responsáveis por grande parte do consumo energético de uma planta. Várias propostas de melhoria de projeto em colunas de destilação são sugeridas com base na análise de perdas de exergia no interior de colunas. Este método visa obter informações a respeito das irreversibilidades existentes e, com isto, propor alterações a fim de diminuir tais irreversibilidades. Também são propostas melhorias operacionais e de projeto para plantas industriais com base na análise das perdas de exergia.

Esta dissertação visa avaliar a efetividade de uma otimização com base na análise de perdas de exergia em uma Unidade de Processamento de Gás Natural (UPGN) frente a uma otimização energética. Esta efetividade possibilitaria a elaboração de malhas de controle de processo com *setpoint* definido para perda de exergia. Para a execução do estudo desta dissertação se modelou a UPGN no software Aspen Plus. Os resultados das simulações produzidas pelo Aspen Plus foram importados pelo software Python, através de interface configurada neste último. Por meio do Python, realizou-se otimizações na modelagem em Aspen Plus com o objetivo de se obter o mínimo consumo energético e outras otimizações com objetivo de minimizar as perdas de exergia na UPGN modelada. Comparou-se as soluções ótimas determinadas para os diferentes tipos de otimizações.

Através do estudo realizado, foi possível obter por meio da otimização de perdas de exergia, uma redução no consumo de energia de até 5,4 %. Assim, se concluiu que a análise de exergia é uma boa métrica para avaliação do desempenho operacional. Em função desta conclusão, se pode considerar o uso desta otimização com base nas perdas de exergia para se configurar um RTO (*Real-Time Optimization*) para a unidade. Também é possível ampliar o uso da perda de exergia para se tornar uma variável de controle de processos.

Abstract

Several studies are carried out with the objective of proposing modifications in industrial plants in order to obtain greater operational efficiency, which means a lower economic cost. Special focus applies to the distillation columns, as these equipments are responsible for much of the energy consumption of a plant. Several design improvement proposals in distillation columns are suggested based on the analysis of exergy losses inside columns. This method aims to obtain information about the existing irreversibilities and, with this, to propose alterations in order to diminish such irreversibilities. Also proposed are operational and design improvements for industrial plants based on the analysis of exergy losses.

This dissertation aims to evaluate the effectiveness of an optimization based on the analysis of loss of exergy in a Natural Gas Processing Unit (UPGN) against an energy optimization. This effectiveness would allow the elaboration of process control meshes with setpoint set for a loss of exergy. For the execution of the study of this dissertation, the UPGN was modeled in the Aspen Plus® software. Python software imported the results of the simulations produced by Aspen Plus through an interface configured in Python. Through this software, modeling optimizations performed in Aspen Plus with the objective of obtaining the minimum energy consumption and other optimizations in order to minimize exergy losses in the modeled UPGN. The optimum solutions determined for the different types of optimizations were compared.

Through the study carried out, it was possible to obtain, through the optimization of exergy losses, a reduction in energy consumption of up to 5.4%. Thus, it was concluded that exergy analysis is a good metric for evaluating operational performance. Based on this conclusion, one can consider the use of this optimization based on exergy losses to configure an RTO (Real-Time Optimization) for the unit. It is also possible to extend the use of exergy loss to become a process control variable.

Agradecimentos

Este mestrado tornou-se possível graças à participação de pessoas que, de diversas formas, contribuíram para a finalização desta etapa.

Agradeço aos colegas e amigos que fiz durante esta época, com os quais foi muito agradável retornar à rotina de estudos acadêmicos e de compartilhar valorosos momentos de vida.

Agradeço à UFRGS e ao Departamento de Engenharia Química pela oportunidade de realização do mestrado e a todos os professores que tive, pelo conhecimento transmitido. Agradeço especialmente aos meus professores orientadores, Jorge Trierweiler e Pedro Fernandes, cuja compreensão, dedicação e apoio foram fundamentais para conclusão deste trabalho.

Agradeço à minha esposa Fernanda, que há mais de 20 anos é o maior suporte da minha vida, e junto com nossos filhos, Helena, Matias e Pablo, são os meus maiores professores.

SUMÁRIO

1	Introdução	1
1.1	Objetivos da dissertação	2
1.2	Estrutura da dissertação	3
2	Fundamentação Teórica	5
2.1	Exergia	5
2.2	Procedimento para Construção da Grande Curva Composta da Coluna (CGCC)	9
2.2.1	Análise Pinch	9
2.2.2	Diagrama Temperatura – Entalpia (T-H) e Curva Composta (CC)	10
2.2.3	Grande Curva Composta (GCC)	11
2.2.4	Grande Curva Composta da Coluna (CGCC)	12
2.3	Visão Geral sobre a Otimização de processos	15
3	Revisão Bibliográfica	19
3.1	Exergia aplicada a projetos de unidades	19
3.1.1	Uso do CGCC em otimização	20
3.1.2	Perda de Exergia para a Análise Termodinâmica de Colunas de Destilação	22
3.1.3	Perda de Exergia para a Análise Termodinâmica de Plantas Industriais	30
3.2	Métricas de eficiência operacional	31
3.2.1	Real-Time Optimization (RTO)	32
3.2.2	Self-optimizing control (SOC)	35
3.3	Unidade de Processamento de Gás Natural (UPGN)	36
4	Metodologia Proposta	40
4.1	Modelagem do Processo	41
4.2	Interface em Python	41
4.3	Otimização	42
5	Descrição do Estudo de Caso	44
5.1	Condicionamento do Gás Natural	45
5.2	Sistema de Separação de Produtos	48
5.3	Unidade de Refrigeração	49
6	Resultados e Discussão	52
6.1	Modelagem em Aspen Plus 8.8	52
6.1.1	Ciclo de Refrigeração	55
6.1.2	Colunas de Destilação	60
6.1.3	Trocadores de calor	62
6.1.4	Compressores e turboexpansor	62
6.1.5	Vasos de separação	63
6.2	Função Objetivo das Otimizações	63
6.3	Otimização Energética	67
6.3.1	Composição Original	67
6.3.2	Composição Rica em Pesados	71
6.4	Otimização da Perda de Exergia	75
6.4.1	Composição Original	75
6.4.2	Composição Rica em Pesados	80
7	Conclusões e Sugestões para Trabalhos Futuros	86

7.1 Conclusões.....	86
7.2 Sugestões para Trabalhos Futuros	88
Referências	89
APÊNDICES – Scripts Escritos em Python	92

LISTA DE FIGURAS

Figura 2.1: Diagrama temperatura – entalpia	10
Figura 2.2: Curvas Compostas das correntes frias e quentes	11
Figura 2.3: Grande Curva Composta	12
Figura 2.4: Abordagem Topo-fundo para construção da CGCC	13
Figura 2.5: Exemplos de métodos de otimização global.....	17
Figura 2.6: Fluxograma de um algoritmo genético	18
Figura 3.1: (a) CGCC de coluna com necessidade de modificação no estágio de alimentação; (b) CGCC com necessidade de redução de refluxo	21
Figura 3.2: (a) CGCC sugerindo pré-aquecimento da alimentação; (b) CGCC demonstrando possibilidade de uso de trocador lateral.	21
Figura 3.3: Gráfico de McCabe Thiele (contagem dos estágios a partir do condensador) e perfil de perda de exergia para diferentes razões de refluxo e número de pratos (adaptado de ZEMP; DE FARIA; OLIVEIRA MAIA, 1997).....	23
Figura 3.4: Perfil de perda de exergia para a) corrente de líquido saturado e b) fração molar de vapor de 0,25, na entrada da coluna	25
Figura 3.5: Perfil de perda de exergia para diferentes estágios de alimentação	26
Figura 3.6: Perfil de perda de exergia para coluna sem e com trocador de calor lateral.....	27
Figura 3.7: Perfil de perda de exergia para coluna sem e com trocador de calor lateral.....	28
Figura 3.8: Alteração no perfil de perda exergia devido a adição de estágios na coluna	29
Figura 3.9: Alteração no perfil de perda exergia devido devido a ajustes no número de estágios da coluna e da carga térmica do refeedor lateral	30
Figura 3.10: Arquitetura típica de automação industrial (SCHULTZ, 2015).....	33
Figura 3.11: Arquitetura de funcionamento de um RTO	35
Figura 3.12: Duas configurações diferentes para unidades de processamento de gás natural	38
Figura 5.1: Diagrama de blocos simplificado da UPGN	44
Figura 5.2: Diagrama de temperatura versus composição a pressão constante à 69 bar e à 25,82 bar.....	45
Figura 5.3: Fluxograma com os equipamentos que realizam o condicionamento do gás natural no estudo de caso avaliado nesta dissertação	46
Figura 5.4: Esquema simplificado do processamento de gás natural e interligação com o sistema de refrigeração e óleo térmico (BARALDI, 2015).....	47
Figura 5.5: Fluxograma simplificado da UPGN (BARALDI, 2015)	49
Figura 5.6: Fluxograma do sistema de refrigeração com propano (adaptado de BARALDI, 2015).....	50
Figura 6.1: Diagrama de processo construído no Aspen Plus para a UPGN	54
Figura 6.2: Especificação de variáveis para simulação no modo SM	56
Figura 6.3: Situação não calculada no modo SM	56

Figura 6.4: Representação parcial do ciclo de refrigeração	57
Figura 6.5: Ciclo de refrigeração modelado	59
Figura 6.6: Fluxograma com o turboexpansor TE01 e o <i>split</i> B18.....	65
Figura 6.7: Consumo de energia da planta na condição original e ótima energética	68
Figura 6.8: Perda de exergia na planta na condição original e ótima energética....	68
Figura 6.9: Perfil de perda de exergia para a coluna T01.....	69
Figura 6.10: Consumo de energia da planta na condição original e ótima energética	72
Figura 6.11: Perda de exergia na planta na condição original e ótima energética..	72
Figura 6.12: Perfil de perda de exergia para a coluna T01.....	73
Figura 6.13: Consumo de energia da planta na condição original e ótima exergética	76
Figura 6.14: Perda de exergia na planta na condição original e ótima exergética..	77
Figura 6.15: Perfil de perda de exergia para a coluna T01.....	78
Figura 6.16: Consumo de energia da planta na condição original e ótima exergética	81
Figura 6.17: Perda de exergia na planta na condição original e ótima exergética..	81
Figura 6.18: Perfil de perda de exergia para a coluna T01.....	82
Figura 6.19: Análise dos perfis de perdas de exergias obtidos na otimização energética com a composição rica do gás bruto.....	85

LISTA DE TABELAS

Tabela 2.1: Cálculo da perda de exergia para alguns equipamentos (adaptado de SHIN; YOON; KIM, 2015).....	8
Tabela 3.1: Composição típica do gás natural (GUO; GHALAMBOR, 2005).....	37
Tabela 6.1: Composições consideradas para o gás bruto	55
Tabela 6.2: Dados operacionais da coluna T01	60
Tabela 6.3: Dados operacionais da coluna T02	61
Tabela 6.4: Dados operacionais da coluna T03	61
Tabela 6.5: Área dos trocadores de calor.....	62
Tabela 6.6: Pressões de descarga nos compressores e no turboexpansor	62
Tabela 6.7: Pressão de operação dos vasos de separação	63
Tabela 6.8: Restrições nas variáveis de decisão.....	66
Tabela 6.9: Valores das variáveis de decisão	67
Tabela 6.10: Condição operacional da coluna T01.....	70
Tabela 6.11: Valores das variáveis de decisão	71
Tabela 6.12: Condição operacional da coluna T01.....	74
Tabela 6.13: Valores das variáveis de decisão	75
Tabela 6.14: Condição operacional da coluna T01.....	79
Tabela 6.15: Valores das variáveis de decisão	80
Tabela 6.16: Condição operacional da coluna T01.....	83
Tabela 6.17: Comparativo entre resultados obtidos para as diferentes otimizações	84

ABREVIATÓES

BFGS	Broyden-Fletcher-Goldfarb-Shanno
CC	Curva Composta
CGCC	Grande Curva Composta da Coluna
DEAP	Distributed Evolutionary Algorithms in Python
GCC	Grande Curva Composta
GLP	Gás Liquefeito de Petróleo
GN	Gás Natural
PNMTC	Practical Near Minimum Thermodynamic Column
PSO	Particle Swarm Optimization
RTO	Real-Time Optimization
SOC	Self-Optimizing Control
UPGN	Unidade de Processamento de Gás Natural

NOTAÇÃO E SIMBOLOGIA

C	Capacidade calorífica molar
CO ₂	Dióxido de Carbono
D	Vazão molar de destilado
En	Energia
En _{equip}	Consumo de energia do equipamento “equip”. Equipamento de acordo com Figura 6.1
Ex	Exergia
EX _{loss}	Perda de exergia
EX _{loss,min}	Perda de exergia mínima
EX _Q	Exergia relacionada a transferência de calor
f	Restrições de igualdade de um problema de otimização
f _{min}	Função objetivo de problema de otimização que visa atingir o mínimo
F	Vazão molar da corrente de alimentação
g	Restrições de desigualdade de um problema de otimização
h	Entalpia molar
h ₁	Entalpia molar no estado termodinâmico 1
h ₂	Entalpia molar no estado termodinâmico 2
H _{CGCC}	Entalpia molar da CGCC
h _{cond}	Entalpia molar de condensação
H _D	Entalpia molar da corrente D
H _F	Entalpia molar da corrente F
h _L	Entalpia molar da corrente L _{min}
h _V	Entalpia molar da corrente L _{min}
H _{Lmin}	Entalpia molar da corrente L _{min}
H _{Vmin}	Entalpia molar da corrente V _{min}
L _{min}	Vazão mínima molar de líquido em um estágio de coluna de destilação
J	Função objetivo de um problema de otimização

P_0	Pressão da vizinhança
Q	Calor absorvido ou liberado
Q_{Cond}	Carga térmica do condensador
s	Entropia molar
s_1	Entropia molar no estado termodinâmico 1
s_2	Entropia molar no estado termodinâmico 2
s_{cond}	Entropia molar de condensação
T_0	Temperatura da vizinhança
x_{Dl}	Fração molar do componente l na corrente D
x_{Dh}	Fração molar do componente h (heavy key) na corrente D
x_l^*	Fração molar do componente “l” (light key) na fase líquida em equilíbrio termodinâmico
x_h^*	Fração molar do componente “h” na fase líquida em equilíbrio termodinâmico
V_{min}	Vazão mínima de vapor em um estágio de coluna de destilação
W	Trabalho de eixo
y_h^*	Fração molar do componente “h” (heavy key) na fase vapor em equilíbrio termodinâmico
y_l^*	Fração molar do componente “l” (light key) na fase vapor em equilíbrio termodinâmico
z_l	Fração molar do componente l na corrente F
z_h	Fração molar do componente h na corrente F
$ n $	Valor absoluto do número “n”

1 Introdução

As unidades de processamento de gás natural possuem produtos com elevado valor agregado, por exemplo: propano, butano e componentes mais pesados (SHIN; YOON; KIM, 2015). Somado a isto, estas plantas apresentam elevado custo operacional, devido aos diversos equipamentos abrangidos e elevado consumo de utilidades. Estes dois fatos mostrados justificam os esforços existentes em busca de meios, para uma melhor avaliação operacional e sugestões de modificações, que gerem maior lucratividade.

As colunas de destilação devido à sua demasiada importância dentro dos processos petroquímicos são o foco de diversos estudos de desempenho operacional. Dhole e Linnhoff (DHOLE; LINNHOF, 1993) elaboraram um método para a construção do perfil CGCC (*Column Grand Composite Curve*) para as colunas de destilação baseada na 1^o Lei da Termodinâmica. Neste método é possível sugerir modificações de projeto e operacionais para minimizar o consumo de energia na coluna. Outros estudos se fundamentam na construção de um perfil para a perdas de exergia existentes em uma coluna (PINTO et al., 2011; ZEMP; DE FARIA, 1995; ZEMP; DE FARIA; OLIVEIRA MAIA, 1997). Este método está baseado na 2^o Lei da Termodinâmica e também gera um perfil onde há indicações para modificações de projeto e operacionais da coluna.

Uma análise com base na exergia de uma planta industrial permite visualizar as irreversibilidades existentes. Esses dados permitem propor melhorias operacionais e de projetos. Em estudos realizados por Demirel et al. (ALHAJJI; DEMIREL, 2016; NGUYEN; DEMIREL, 2010) foram propostas melhorias em plantas industriais com base na análise dos perfis de perdas de exergia e do perfil CGCC das colunas de destilação existentes nas unidades de processamentos estudadas. Já, Shin et al. (SHIN; YOON; KIM, 2015) elaboraram uma estrutura de otimização com base na análise de perdas de exergia para se obter o ponto ótimo operacional da planta estudada.

Estes fatos apresentados evidenciam a importância de estudos a serem realizados em processos petroquímicos, sendo esta dissertação também um esforço em busca de um critério para a avaliação de desempenho operacional.

1.1 Objetivos da dissertação

O presente estudo visa avaliar se uma análise operacional de uma Unidade de Processamento de Gás Natural (UPGN), com base no cálculo das perdas de exergia dos equipamentos abrangidos pela unidade, representa uma boa métrica de avaliação de desempenho da planta.

Para isto, se gerou uma estrutura de otimização com objetivo de minimizar a soma das perdas de exergia dos equipamentos da UPGN estudada. Também se realizou, uma otimização energética que visa atingir o menor consumo de energia na planta.

Este estudo considerou, além da composição original do gás bruto que entra na UPGN, uma composição enriquecida de componentes pesados neste gás. Assim, a planta opera fora de sua condição de projeto tornando os efeitos da otimização mais evidentes e gerando mais dados para análise.

1.2 Estrutura da dissertação

A presente dissertação está estruturada da seguinte forma:

- Capítulo 2 – Fundamentação Teórica

Este capítulo é dedicado a elucidar os principais conceitos aplicados no desenvolvimento desta dissertação. Primeiro, são apresentados conceitos referentes à exergia, explica-se sua definição, cálculo para correntes e trabalho térmico e cálculo da perda de exergia para diferentes equipamentos. Segundo, explica-se conceitos referentes a elaboração do perfil CGCC. Por fim, mostra-se uma visão geral de otimização de processos com foco em algoritmo genético, por ser a ferramenta utilizada nesta dissertação.

- Capítulo 3 – Revisão Bibliográfica

Capítulo que apresenta diversos trabalhos com foco na análise da eficiência de processos, principalmente de colunas de destilação, que se utilizam de recursos como: construção do perfil CGCC para sugerir melhorias em colunas; análise da perda de exergia de colunas de destilação também para sugerir melhorias em colunas; análise da perda de exergia e do CGCC de colunas para identificar melhorias na unidade de processo; e otimização das perdas de exergia existentes na planta, para atingir melhor desempenho operacional da planta.

Também é visto neste capítulo explicações referentes a métricas de eficiência operacionais, RTO (*Real-Time Optimization*) e SOC (*Self-Optimizing Control*), e uma descrição geral de Unidades de Processamento de Gás Natural.

- Capítulo 4 – Metodologia Proposta

Apresentada neste capítulo a modelagem em Aspen Plus construída para a UPGN. Após se apresenta a interface desenvolvida entre Aspen Plus e Python, sendo este último responsável, entre outras coisas, pela execução do algoritmo de otimização. Por fim, é detalhado o procedimento de otimização efetuado.

- Capítulo 5 – Descrição do Estudo de Caso

Nesta seção é descrita a planta operacional que serviu de base para a aplicação das otimizações operacionais. Para fins de explicação, dividiu-se a planta em três sistemas, sendo eles: condicionamento do gás bruto, sistema de separação e unidade de refrigeração. As funcionalidades dos principais equipamentos são exibidas nesta seção.

- Capítulo 6 – Resultados e Discussão

Os resultados são apresentados, analisados e discutidos neste capítulo.

- Capítulo 7 – Conclusões e Sugestões para Trabalhos Futuros

Capítulo dedicado a exibir as conclusões e propor sugestões para trabalhos futuros

2 Fundamentação Teórica

2.1 Exergia

A energia para a realização de um processo é formada por uma parcela que pode ser convertida em outra forma de energia, chamada de exergia, e por outra que representa as perdas de energia decorrente das irreversibilidades do processo.

Em 1956, Z. Rant propôs o termo exergia. Segundo o autor, a exergia é “a parte da energia que pode ser completamente convertida em qualquer outra forma de energia”(MOUSSA, 2001).

A análise exérgica é baseada na segunda lei da Termodinâmica e pode ser aplicada no projeto e otimização de processos químicos, uma vez que pode ser utilizada para examinar irreversibilidades inerentes nos processos, ou seja, ineficiências. O uso da análise exérgica apresenta duas vantagens práticas. Primeiro, ela é conveniente para entender as interações no sistema entre correntes de entrada, saída, trabalho e calor dentro de um processo. Segundo, as irreversibilidades do processo podem ser analisadas esquematicamente através de representações gráficas (BEJAN, 1982 apud SHIN; YOON; KIM, 2015).

Em termos de recursos energéticos, deseja-se maximizar o trabalho/calor obtido de um processo ou minimizar o trabalho/calor a ser fornecido. Sabe-se que um processo reversível, que corresponde a uma variação nula da entropia do universo, representa o máximo de produção, ou equivalente, o mínimo de fornecimento possível de calor/trabalho para uma dada mudança de estado. No entanto, uma perspectiva alternativa às vezes é útil, especialmente no contexto de se analisar sistemas complexos contendo muitas operações unitárias. Com este objetivo se apresenta a análise de exergia. Esta análise considera o trabalho máximo que podemos obter de um processo em relação à sua vizinhança. Esta abordagem é útil para uma análise metodológica de processos complexo. Através da análise

exergética é possível observar cada etapa de um processo de múltiplas etapas e determinar a magnitude relativa das irreversibilidades e perdas de trabalho existentes (KORETSKY, 2013).

Através do conceito de exergia se tem uma variável que permite medir a quantidade de trabalho máximo que é possível obter de um determinado processo (ou a quantidade mínima necessária para executar um determinado processo). A exergia de uma corrente de processo em uma dada temperatura T e pressão P é definida como a quantidade de trabalho que pode ser obtida quando esta corrente sofre modificação do seu estado de equilíbrio inicial, através de um processo reversível, para um novo estado de equilíbrio na condição de temperatura, T_0 , e pressão, P_0 , da vizinhança. A exergia de uma corrente é dada por:

$$Ex = h - T_0 \cdot s \quad (2.1)$$

onde, h é a entalpia da corrente, s a entropia e T_0 é a temperatura de referência do ambiente.

Como a exergia da corrente é dada em função da sua entalpia e entropia, sendo ambas funções de estado, logo a exergia também é função de estado. Portanto, a exergia de uma dada corrente monofásica pode ser calculada através de sua temperatura, pressão e composição. Considerando-se que em um sistema ocorra a variação a partir de uma Estado 1 para um Estado 2, o cálculo da variação da exergia do sistema é dado por:

$$\Delta Ex = (h_2 - h_1) - T_0(s_2 - s_1) \quad (2.2)$$

A Equação (2.2) se assemelha com a definição da Energia Livre de Gibbs. Deve-se observar que, para esta, a temperatura corresponde à temperatura do sistema, já para a Equação (2.2), T_0 corresponde à temperatura da vizinhança. No capítulo 13 do livro *Fundamentals of Engineering Thermodynamics* (MORAN; SHAPIRO, 2006) é introduzido o conceito de exergy química, onde são descritas as expressões para a exergia química em termos da energia de Gibbs.

Considerando como exemplo um processo de condensação de uma corrente a temperatura constante e na pressão atmosférica, o cálculo da variação da exergia seria dada por:

$$\Delta Ex = \Delta h_{cond} - T_0 \cdot \Delta s_{cond} \quad (2.3)$$

Conforme a segunda lei da Termodinâmica, a variação da entropia para um processo a pressão constante está relacionada com a quantidade de calor transferida de acordo com a equação a seguir:

$$\Delta s = \frac{Q}{T} \quad (2.4)$$

Considerando-se que à pressão constante a variação de entalpia (Δh) é igual ao calor absorvido ou liberado (Q) durante o processo, então a variação de entropia será:

$$\Delta s = \frac{\Delta h}{T} \quad (2.5)$$

Substituindo-se a Equação (2.4) na Equação (2.3) e considerando a variação de entalpia (Δh) igual ao calor absorvido ou liberado (Q), conforme Equação (2.5), obtém-se a equação da variação de exergia relacionada a transferência de calor, Ex_Q :

$$Ex_Q = Q \left(1 - \frac{T_0}{T} \right) \quad (2.6)$$


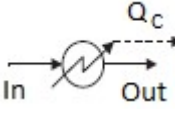

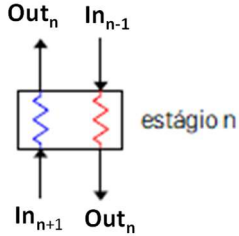
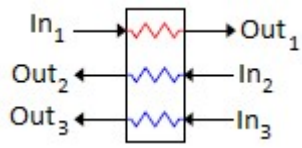
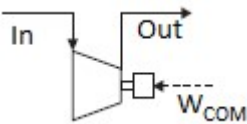
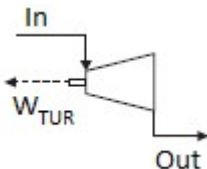
Em um processo, a diferença entre o trabalho realizado, ou sofrido, e o trabalho necessário para ocorrer a mesma mudança de estado de forma reversível, representa o trabalho desperdiçado no processo que é definido como perda de exergia. Assim a perda de exergia pode ser calculada através de um balanço, conforme Equação (2.7).

$$Ex_{in} + Ex_{Qin} = Ex_{out} + Ex_{Qout} + W + Ex_{loss} \quad (2.7)$$

Onde, Ex_{in} e Ex_{out} são as exergias de entrada e saída do sistema, Ex_{Qin} e Ex_{Qout} são as exergias térmicas de entrada e saída do sistema, W é o trabalho realizado no/pelo sistema e Ex_{loss} é a perda de exergia do sistema.

O cálculo da perda de exergia para os diferentes equipamentos utilizados deve respeitar as características inerentes a cada um dos processos envolvidos. Ou seja, em trocadores de calor utiliza-se as condições de exergia para as correntes envolvidas nas entradas e saídas do equipamento. Já para aquecedores usa-se a exergia da corrente a ser aquecida na entrada e saída do equipamento e a exergia térmica referente ao calor cedido pela utilidade quente. Na Tabela 2.1 estão apresentadas as equações utilizadas para o cálculo da perda de exergia para alguns equipamentos.

Tabela 2.1: Cálculo da perda de exergia para alguns equipamentos (adaptado de SHIN; YOON; KIM, 2015)

Válvula		$EX_{VLV,loss} = EX_{in} - EX_{out}$
Cooler (Condensador de Coluna de Destilação)		$EX_{C,loss} = Q_C \left(1 - \frac{T_0}{T_C}\right) + EX_{in} - EX_{out}$
Aquecedor (Refrvedor de Coluna de Destilação)		$EX_{A,loss} = EX_{in} - EX_{out} - Q_A \left(1 - \frac{T_0}{T_A}\right)$
Estágios de Coluna de Destilação		$EX_{TC,loss} = \sum EX_{in} - \sum EX_{out}$
Trocador de Calor		$EX_{TC,loss} = \sum EX_{in} - \sum EX_{out}$
Compressor		$EX_{COM,loss} = EX_{in} + W_{in} - EX_{out}$
Turbina		$EX_{TUR,loss} = EX_{in} - W_{out} - EX_{out}$

O cálculo da perda de exergia para colunas de destilação é realizado considerando a transferência de calor e massa em cada estágio da coluna. Assim, para o estágio do refeedor e o do condensador, o cálculo é efetuado conforme a

equação utilizada na Tabela 2.1 para o aquecedor e para o cooler, respectivamente. Caso a coluna de destilação possua algum estágio com trocador de calor lateral, calcula-se a perda de exergia considerando o balanço de exergia das correntes que entram e saem de cada estágio, assim como o calor adicionado ou removido no estágio através do trocador lateral.

Para as colunas de destilação, pode-se apresentar os resultados dos cálculos de perda de exergia para cada estágio através de um gráfico de perda de exergia versus temperatura (ou estágio), conforme demonstrado por Zemp (ZEMP; DE FARIA; OLIVEIRA MAIA, 1997). Por meio deste gráfico, chamado de Perfil de Perda de Exergia, avalia-se as condições operacionais da coluna e também se estipula modificações para sua otimização.

Quando a perda de exergia de um processo diminui, significa que a operação está ocorrendo mais próximo à reversibilidade. Portanto, a perda de exergia pode ser utilizada para mensurar o grau de ineficiência existente em um processo. Neste trabalho se utilizará a perda de exergia para avaliar a operação de uma Unidade de Processamento de Gás Natural (UPGN).

2.2 Procedimento para Construção da Grande Curva Composta da Coluna (CGCC)

A CGCC é um perfil construído com base na primeira Lei da Termodinâmica que auxilia na identificação de melhorias possíveis de se realizar em uma coluna de destilação a fim de se aumentar sua eficiência. O método de cálculo utilizado neste estudo para obtenção da CGCC baseia-se no modo apresentado no artigo de DHOLE; LINNHOFF (1993). A CGCC é construída a partir da aplicação da análise pinch e da Curva Composta para coluna. Estes conceitos serão detalhados a seguir.

2.2.1 Análise Pinch

Esta é uma metodologia baseada na Termodinâmica concebida na década de 70 para efetuar a integração energética em indústrias. Esta metodologia indica qualitativamente possibilidades de economia de energia através do uso de integração energética entre as diferentes correntes de processo presentes na indústria. Ao decorrer do tempo a análise pinch evoluiu de uma metodologia aplicada em problemas de recuperação de calor para uma análise global de processo.

2.2.2 Diagrama Temperatura – Entalpia (T-H) e Curva Composta (CC)

O diagrama Temperatura – Entalpia (T-H) é uma ferramenta utilizada pela análise Pinch para se construir a Curva Composta (CC). No diagrama T-H são plotadas as necessidades de variação de temperatura das correntes frias (aquelas que necessitam ser aquecidas) e das correntes quentes (aquelas que necessitam ser resfriadas). Conhecendo-se a capacidade calorífica, C , de cada corrente (que é considerada constante) se sabe a quantidade de calor de troca necessária para se atingir a temperatura desejada (em um processo que não ocorra mudança de fase). Neste caso o calor trocado, Q , corresponde a variação de entalpia, ΔH , da corrente.

$$Q = C \cdot \Delta T = \Delta H \quad (2.8)$$

Um exemplo de diagrama T-H é mostrado na Figura 2.1, onde o diagrama é montado para duas correntes. A inclinação das retas que representam as correntes corresponde ao inverso da capacidade calorífica. O diagrama T-H pode ser utilizado para representar as trocas de calor por ser uma ferramenta muito prática (KEMP, 2007).

Para lidar com múltiplas correntes, adicionamos no diagrama T-H os dados individuais de cada corrente. Após, se agrupa as retas correspondentes das correntes frias e as das correntes quentes. Assim, gera-se a Curva Composta (CC) das correntes frias e a CC das correntes quentes, conforme ilustrado na Figura 2.2.

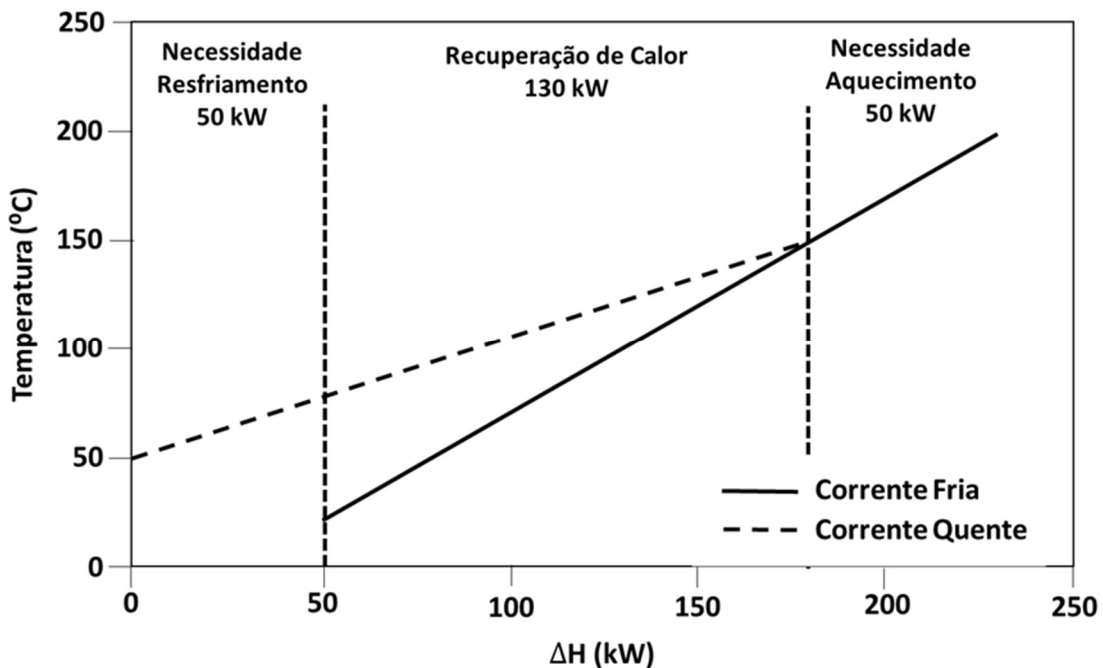


Figura 2.1: Diagrama temperatura – entalpia

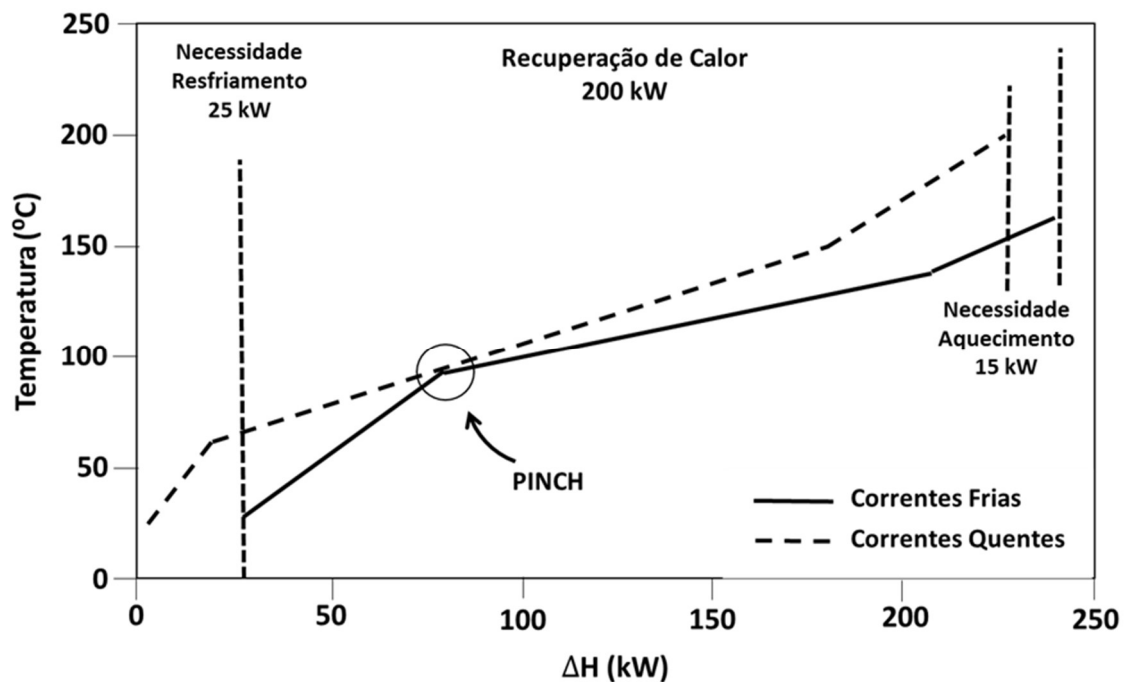


Figura 2.2: Curvas Compostas das correntes frias e quentes

O ponto de intersecção entre as CC das correntes frias e as das correntes quentes é o ponto “Pinch”. Este ponto divide o diagrama das CCs em duas regiões. A região superior ao Pinch, onde há o consumo de utilidade quente para atender a necessidade de aquecimento. Já na região inferior ao Pinch há consumo de utilidade fria para atender a necessidade de resfriamento.

O uso da Análise Pinch cria três regras a serem seguidas a fim de se atingir um consumo mínimo de utilidades (KEMP, 2007):

- Não transferir calor através do ponto de Pinch;
- Não usar utilidade fria acima do Pinch;
- Não usar utilidade quente abaixo do Pinch;

2.2.3 Grande Curva Composta (GCC)

O gráfico obtido através da diferença das entalpias entre a CC fria e quente para cada temperatura é chamada de Grande Curva Composta, conforme ilustrado na Figura 2.3. Este gráfico representa a diferença entre o calor disponível nas correntes quentes e o calor necessário pelas correntes frias para uma dada temperatura.

Desse modo, pode encontrar a quantidade mínima necessária de aquecimento ou resfriamento a ser fornecido em uma dada temperatura (KEMP, 2007).

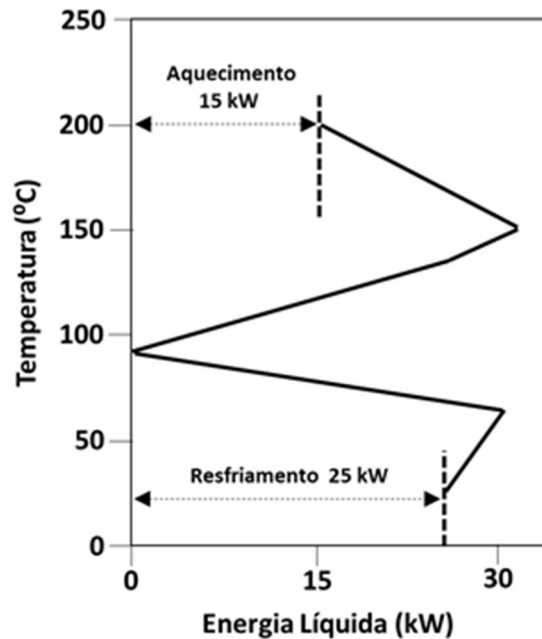


Figura 2.3: Grande Curva Composta

2.2.4 Grande Curva Composta da Coluna (CGCC)

O conceito de GCC expandiu e abordou a análise em colunas de destilação, onde é elaborado um perfil de grande curva composta para as colunas de destilação. Neste perfil, os cálculos se baseiam em uma separação binária ou pseudobinária. O refluxo é mínimo, sendo assim se tem as vazões mínimas para o vapor e o líquido em cada estágio da coluna. Nesta condição a intersecção das linhas de operação se encontram sobre a curva de equilíbrio. As linhas de operação são obtidas através do balanço de massa na coluna nas regiões de esgotamento e retificação. Na construção do perfil a partir do topo da coluna, o balanço de massa começa no topo da coluna e, progressivamente, este balanço abrangerá mais estágios em direção ao fundo da coluna (Figura 2.4). A análise realizada através da CGCC é apresentada na seção 3.1.1.

O procedimento para construção da CGCC se difere um pouco para as regiões acima do estágio de alimentação e abaixo do mesmo, conforme apresentado por Linnhoff (DHOLE; LINNHOFF, 1993) em seu trabalho.

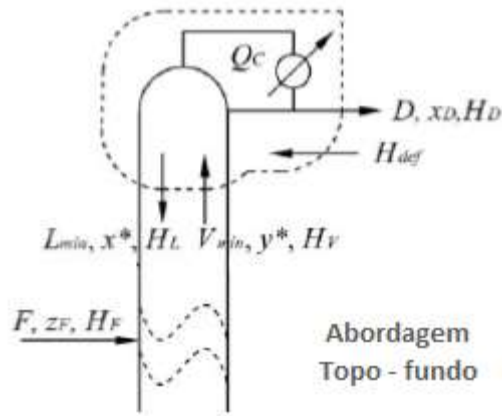


Figura 2.4: Abordagem Topo-fundo para construção da CGCC

2.2.4.1 Construção CGCC acima do estágio de alimentação

Para cada estágio acima da alimentação o balanço de massa por componente é calculado da seguinte forma:

$$V_{min} \cdot y_l^* - L_{min} \cdot x_l^* = D \cdot x_{Dl} \quad (2.9)$$

$$V_{min} \cdot y_h^* - L_{min} \cdot x_h^* = D \cdot x_{Dh} \quad (2.10)$$

Estas equações estabelecem as vazões mínimas de vapor e líquido para cada estágio da coluna. Resolvendo este sistema de equações se obtém os valores para V_{min} e L_{min} . Os demais dados das equações são obtidos através do resultado da simulação convergida.

$$L_{min} = \frac{D \cdot x_{Dh} - \frac{D \cdot x_{Dl} \cdot y_h^*}{y_l^*}}{\frac{y_h^* \cdot x_l^*}{y_l^*} - x_h^*} \quad (2.11)$$

$$V_{min} = \frac{D \cdot x_{Dl} + L_{min} \cdot x_l^*}{y_l^*} \quad (2.12)$$

A partir dos valores calculados para V_{min} e L_{min} se obtém as entalpias correspondentes a essas vazões.

$$H_{L_{min}} = L_{min} \cdot h_L \quad (2.13)$$

$$H_{V_{min}} = V_{min} \cdot h_V \quad (2.14)$$

Através de um balanço de energia se obtém a taxa de acúmulo de entalpia em cada estágio da coluna. Este acúmulo de entalpia é chamado de H_{def} .

$$H_{def} = H_{V_{min}} - H_{L_{min}} - H_D \quad (2.15)$$

Finalmente é possível se obter o valor da entalpia para a construção da CGCC.

$$H_{CGCC} = H_{def} + Q_{Cond} \quad (2.16)$$

O cálculo da H_{CGCC} deve ser realizado deste modo em cada estágio de temperatura da coluna acima da alimentação. Quando existir trocadores intermediários na coluna a H_{CGCC} deve considerá-lo.

$$H_{CGCC} = H_{def} + Q_{Cond} + Q_{Int} \quad (2.17)$$

2.2.4.2 Construção CGCC no estágio de alimentação e estágios abaixo S_{deste}

Para o estágio da alimentação e os estágios abaixo deste o balanço de massa por componente deve conter os dados da corrente que alimenta a coluna. O cálculo é efetuado da seguinte forma:

$$V_{min} \cdot y_l^* + F \cdot z_l = L_{min} \cdot x_l^* + D \cdot x_{D_l} \quad (2.18)$$

$$V_{min} \cdot y_h^* + F \cdot z_h = L_{min} \cdot x_h^* + D \cdot x_{D_h} \quad (2.19)$$

Resolvendo este sistema de equações se obtém os valores para V_{min} e L_{min} . Os demais dados das equações são obtidos através do resultado da simulação convergida.

$$L_{min} = \frac{(D \cdot x_{D_h} - F \cdot z_h) - \frac{D \cdot x_{D_l} \cdot y_h^*}{y_l^*} + \frac{F \cdot z_l \cdot y_h^*}{y_l^*}}{\frac{y_h^* \cdot x_l^*}{y_l^*} - x_h^*} \quad (2.20)$$

$$V_{min} = \frac{D \cdot x_{D_l} - F \cdot z_l + L_{min} \cdot x_l^*}{y_l^*} \quad (2.21)$$

A partir dos valores calculados para V_{min} e L_{min} se obtém as entalpias correspondentes a essas vazões.

$$H_{L_{min}} = L_{min} \cdot h_L \quad (2.22)$$

$$H_{V_{min}} = V_{min} \cdot h_V \quad (2.23)$$

Através de um balanço entálpico se obtém a taxa de acúmulo de entalpia em cada estágio da coluna.

$$H_{def} = H_{V_{min}} + H_F - H_{L_{min}} - H_D \quad (2.24)$$

Por fim se obtém o valor da entalpia para a construção da CGCC.

$$H_{CGCC} = H_{def} + Q_{Cond} \quad (2.25)$$

Quando existir trocadores intermediários na coluna a H_{CGCC} deve considerá-lo.

$$H_{CGCC} = H_{def} + Q_{Cond} + Q_{Int} \quad (2.26)$$

2.3 Visão Geral sobre a Otimização de processos

Um problema de otimização consiste em maximizar ou minimizar uma função real escolhendo sistematicamente os valores de entrada de um conjunto permitido e computando o valor da função. Esta técnica é uma das ferramentas quantitativas mais utilizadas para tomada de decisões na indústria. Uma grande variedade de problemas em projetos, construções, operação e análises de plantas químicas (como também muitos outros processos industriais) podem ser resolvidos por otimização (EDGAR; HIMMELBLAU; LASDON, 2001). O objetivo da otimização é encontrar os valores das variáveis de processo que fornecem a melhor solução para um problema estudado.

Para efetuar uma otimização são necessários definir uma função objetivo a solucionar e quais são as restrições existentes no problema de otimização. A função objetivo representa fatores tais como, custo, lucro, energia e esta função é escrita em termos das variáveis chaves do processo. Estas variáveis chaves são definidas de acordo com o modelo do processo, o qual restringe a faixa de variação dessas variáveis. Por exemplo, caso possua uma variável chave que representa a pressão de uma corrente, através do modelo do processo conhece-se os valores máximo e mínimo, ou seja, as restrições desta variável. Esta pressão poderá ter um mínimo definido para que seja suficiente escoar para outro equipamento, ou para se evitar

mudança de fase, e poderá ter um máximo definido para não ocorrer contrafluxo, ou para não exceder a pressão de projeto de algum equipamento.

Um problema de otimização contém três categorias essenciais:

1. Pelo menos uma função objetivo a ser otimizada;
2. Restrições de igualdade (equações);
3. Restrições de desigualdade (inequações).

Com estas três categorias se determina qual é a região viável de soluções, ou seja, um conjunto de variáveis de decisão que satisfazem as condições igualdade e desigualdade. A solução ótima é o conjunto de valores das variáveis de decisão que satisfazem as condições de restrições e fornecem a solução ótima para a função objetivo. Os problemas de otimização normalmente possuem a seguinte formulação:

$$\begin{cases} \min J(x_1, x_2, \dots, x_n) \\ \bar{f}(x_1, x_2, \dots, x_n) = 0 \\ \bar{g}(x_1, x_2, \dots, x_n) \leq 0 \end{cases}$$

Onde, S

J é a função objetivo;

x_1, x_2, \dots, x_n são as variáveis do modelo;

$\bar{f}(x_1, x_2, \dots, x_n)$ representam as restrições de igualdade do problema de otimização;

$\bar{g}(x_1, x_2, \dots, x_n)$ representam as restrições de desigualdade do problema de otimização.

Existem diversos métodos de otimização que são aplicados de acordo com as características do sistema a ser otimizado. Alguns métodos são indicados para localizar mínimos locais do problema, por exemplo: BFGS, Simplex, Newton, Método da Maior Descida, entre outros. Esses métodos podem ser aplicados para problemas convexos, ou seja, o mínimo local será o mínimo global. Caso o problema tenha mais de um mínimo deve-se ter uma estimativa Sinicial que garanta que o método de otimização se encaminhe para a solução global do problema.

Para problemas de otimização que possuam diversos mínimos e são mais complexos se faz necessários métodos de otimização globais. A Figura 2.5 mostra alguns métodos de otimização globais (EDGAR; HIMMELBLAU; LASDON, 2001).

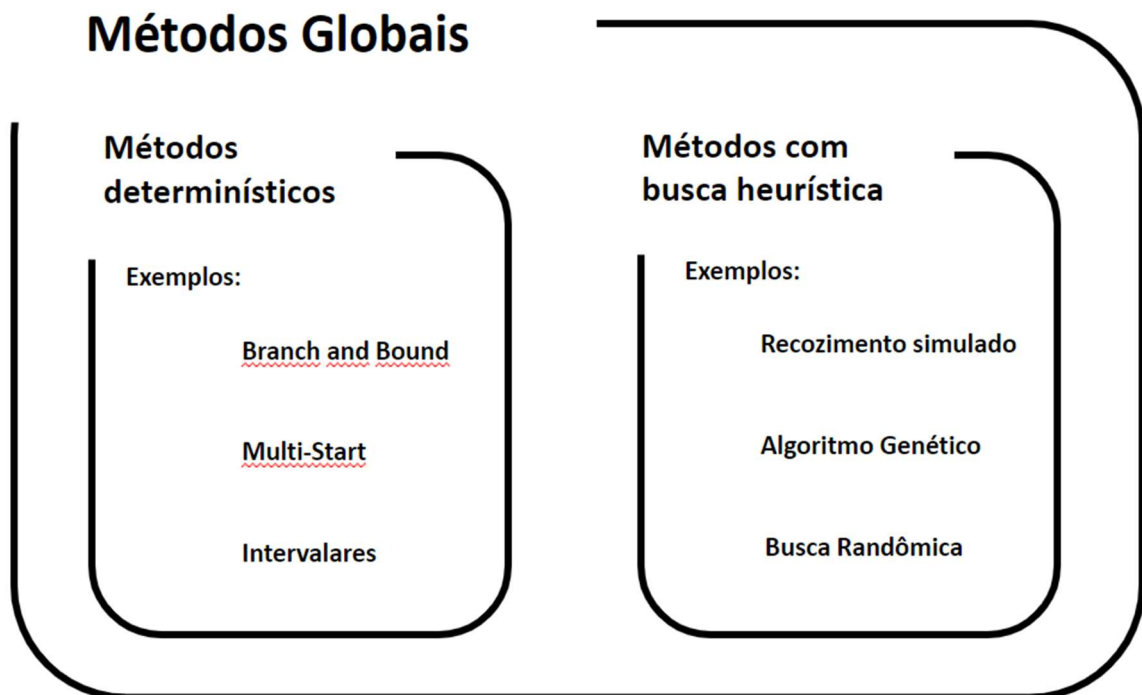


Figura 2.5: Exemplos de métodos de otimização global

Os algoritmos de otimização globais com métodos exatos executam o problema de otimização até encontrar o critério de terminação estabelecido, esses métodos garantem encontrar um resultado próximo ao mínimo global do problema e é possível verificar a solução encontrada. Os métodos de busca heurística podem, e frequentemente, encontram a solução ótima global, mas esses métodos não garantem que encontrarão a solução ótima e, normalmente, não se pode provar que se atingiu a solução ótima, mesmo quando se consegue. O método com busca heurística inicia seu procedimento com uma solução atual e soluciona o problema para outros diversos pontos na vizinhança da solução atual à procura de uma solução melhor. Esse procedimento é repetido caso se encontre um ponto melhor (EDGAR; HIMMELBLAU; LASDON, 2001).

O método de otimização utilizada nesta dissertação é a de algoritmo genético. Este é um método heurístico que imita em seu algoritmo o processo biológico de cruzamento, seleção e mutação. Este método gera aleatoriamente uma população com as variáveis chaves do problema de otimização, respeitando as restrições dessas variáveis. O algoritmo gera posteriormente uma nova população com novas variáveis baseada em sua estratégia evolucionária (mutação e cruzamento) e testa os resultados para esta nova população. O processo de gerar nova população é repetido iterativamente até que o indivíduo ótimo seja determinado.

A Figura 2.6 ilustra os procedimentos executados em um algoritmo genético.

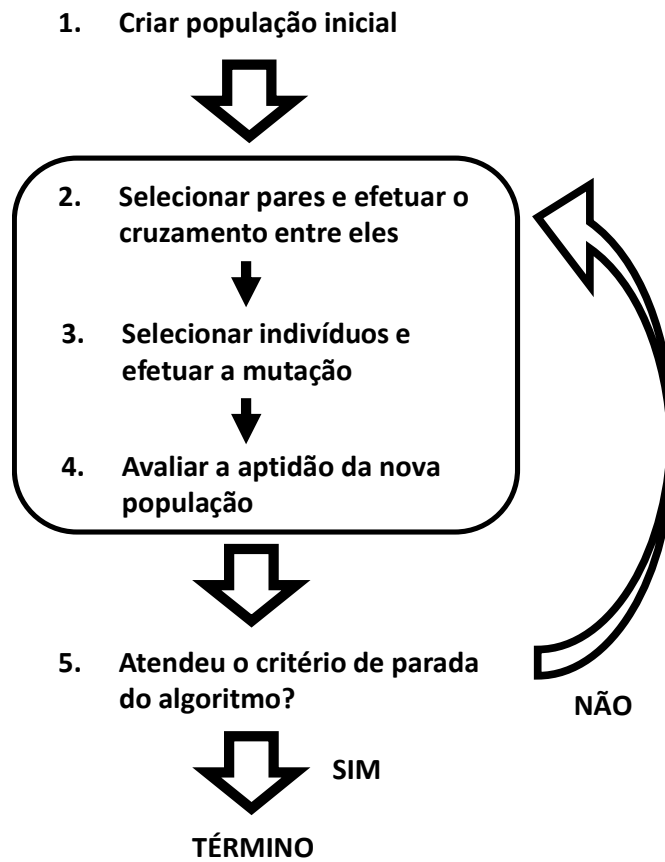


Figura 2.6: Fluxograma de um algoritmo genético

3 Revisão Bibliográfica

Nos itens seguintes deste capítulo serão apresentados uma visão geral de estudos realizados com foco na utilização do CGCC e análise de perdas de exergia como parâmetros para avaliação de eficiência operacional de colunas de destilação e de unidades de processo. Também será discutido métricas de eficiência operacional que possam se beneficiar com a análise de perdas de exergia. Por fim é apresentada uma introdução do gás natural e de sua unidade de processamento.

3.1 Exergia aplicada a projetos de unidades

A exergia vem sendo largamente utilizada em estudos para avaliação da eficiência operacional de equipamentos. Isto devido a capacidade desta variável de medir, através da perda de exergia, ineficiências operacionais. Assim, através dos resultados calculados para as perdas é possível avaliar soluções para otimizar o projeto dos equipamentos.

Foco especial do uso da análise exergética vem sendo dado a colunas de destilação devido a importância destes equipamentos. Segundo o Departamento de Energia dos Estados Unidos da América, de 90 a 95 % de todas as separações de fluidos realizadas em plantas de processamento nos EUA são efetuadas através de colunas de destilação (OFFICE OF ENERGY EFFICIENCY & RENEWABLE ENERGY, 2016). Também, segundo mesma fonte, o custo capital e de operação das colunas de destilação para uma típica planta de processamento representa de 40 a 70 % do custo total da planta.

Trabalhos anteriores para o projeto de colunas de destilação abrangem a construção do perfil da Grande Curva Composta da Coluna (CGCC, sigla em inglês) para a visualização de variáveis a serem otimizadas na coluna buscando uma

diminuição do consumo de energia. Linnhoff (DHOLE; LINNHOFF, 1993) apresentou em seu trabalho um modo de se construir a CGCC para colunas multicomponentes e também como se utilizar a CGCC para se otimizar uma coluna. Na seção 2.2.4 se explica a construção da CGCC, e na seção 3.1.1 explica seu uso em otimização.

A CGCC é uma boa ferramenta a ser agregada junto com a análise exérgica, pois enquanto esta evidencia os pontos de irreversibilidade no interior de uma coluna, a CGCC mostra uma referência da quantidade de calor que poderemos adicionar em um estágio específico. O perfil resultante da CGCC indica quais são as modificações necessárias na operação da coluna para operá-la em condição ótima.

3.1.1 *Uso do CGCC em otimização*

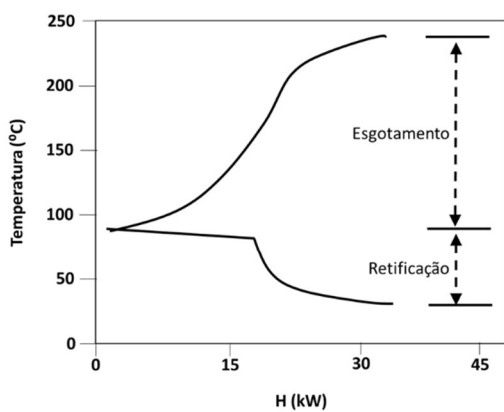
O CGCC é utilizado para estipular metas para diferentes modificações em colunas.

A otimização de colunas de destilação está diretamente relacionada com a escolha mais conveniente de parâmetros como: condição térmica da alimentação, razão de refluxo e trocadores laterais. Esta metodologia proposta mostra que o estudo de um perfil entalpia e temperatura de uma coluna de destilação pode identificar, para os fatores anteriormente citados, qual a configuração da coluna que proporciona maior eficiência termodinâmica. É possível identificar qual desses parâmetros poderá ser apropriado para um determinado sistema de estudo (MOUSSA, 2001). As modificações na coluna visam a redução das forças motrizes existentes nessa. A sequência recomendada por Moussa para as modificações na coluna é a seguinte:

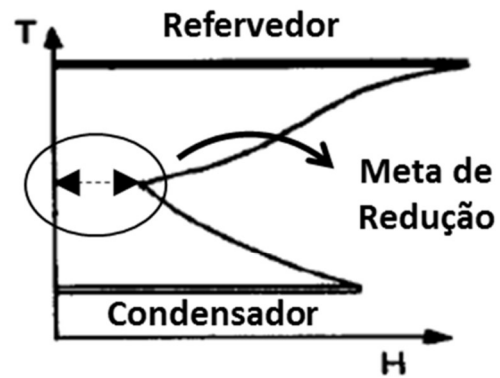
- Localização da alimentação: A necessidade de modificar o estágio de alimentação da coluna é percebida pela presença de patamares na região de esgotamento ou na de retificação no gráfico da CGCC. Na Figura 3.1.a é apresentado o CGCC de uma coluna com necessidade de modificação no estágio de alimentação. Percebe-se pelo patamar existente na região de retificação que a alimentação está muito próxima ao topo da coluna e esta deve ser deslocada mais para o fundo da coluna;
- Redução de refluxo: Escopo para redução do refluxo é percebida na CGCC pela distância da curva em relação ao eixo da temperatura. Modificações no refluxo visam aproximar a CGCC deste eixo (Figura 3.1.b);
- Condição térmica da alimentação: Se após ajustar os dois primeiros parâmetros ainda permanecer patamar na CGCC o próximo parâmetro a

ser ajustado será a mudança da condição térmica da alimentação. A extensão dessa região indica, aproximadamente, a meta para pré-aquecimento (Figura 3.2.a);

- Uso de trocadores laterais: A Figura 3.2.b apresenta um potencial para uso de trocador lateral, mesmo após as demais modificações. Nesta situação, a alocação de um trocador intermediário, permite que seja realizado, em uma temperatura superior, parte da troca térmica que seria feita pelo condensador.

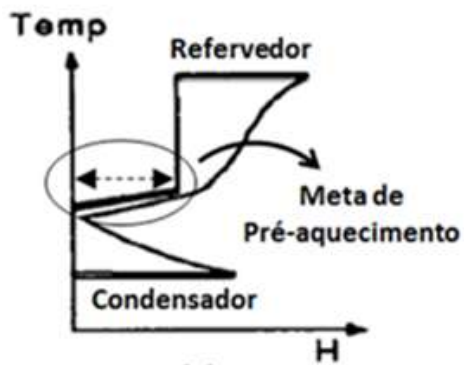


(a)

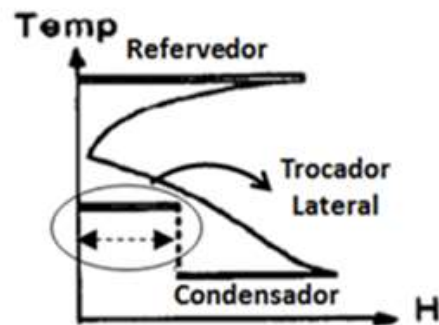


(b)

Figura 3.1: (a) CGCC de coluna com necessidade de modificação no estágio de alimentação; (b) CGCC com necessidade de redução de refluxo



(a)



(b)

Figura 3.2: (a) CGCC sugerindo pré-aquecimento da alimentação; (b) CGCC demonstrando possibilidade de uso de trocador lateral.

3.1.2 Perda de Exergia para a Análise Termodinâmica de Colunas de Destilação

Zemp *et al.* (ZEMP; DE FARIA; OLIVEIRA MAIA, 1997) apresentaram em seu trabalho o uso do cálculo da perda de exergia no interior de uma coluna de destilação para analisar termodinamicamente a eficiência da coluna. Nesse trabalho se explicou como esse perfil demonstra modificações possíveis em uma coluna a fim de melhorar sua eficiência e se comparou o resultado obtido com o uso do método da CGCC. Nesta comparação percebe-se que os métodos geram como resultado modificações diferentes a serem realizadas na coluna para sua otimização.

Processos de destilação ocorrem em condições de irreversibilidade havendo perda de exergia. Assim, a exergia líquida que entra através das utilidades em uso na coluna é maior que a exergia do trabalho necessário para o processo ocorrer de forma reversível. Calculando-se o balanço de exergia neste sistema obtém-se a perda de exergia global da coluna. Este mesmo cálculo pode ser realizado para cada estágio da coluna e, assim, se obter a perda de exergia existente em um estágio para se efetuar a transferência de massa. Com os resultados do balanço de exergia em cada estágio, se pode elaborar o perfil de perda de exergia, o qual evidencia a distribuição das forças motrizes na coluna.

A relação entre perda de exergia e força motriz é visualizada comparando-se o perfil de perda de exergia com um gráfico que represente as forças motrizes na coluna, por exemplo: McCabe-Thiele. A Figura 3.3 apresenta esta comparação, onde foi selecionada uma coluna para separar uma mistura equimolar de n-butano e i-pentano, com variação da razão de refluxo de total ao mínimo. Nesta figura os estágios estão numerados a partir do condensador. Claramente se percebe que existe uma relação entre perda de exergia e forças motrizes, sendo esta observada através da distância entre a linha de operação e a de equilíbrio. Estágios da coluna com maiores mudanças na composição e temperatura são os que apresentam maiores perda de exergia.

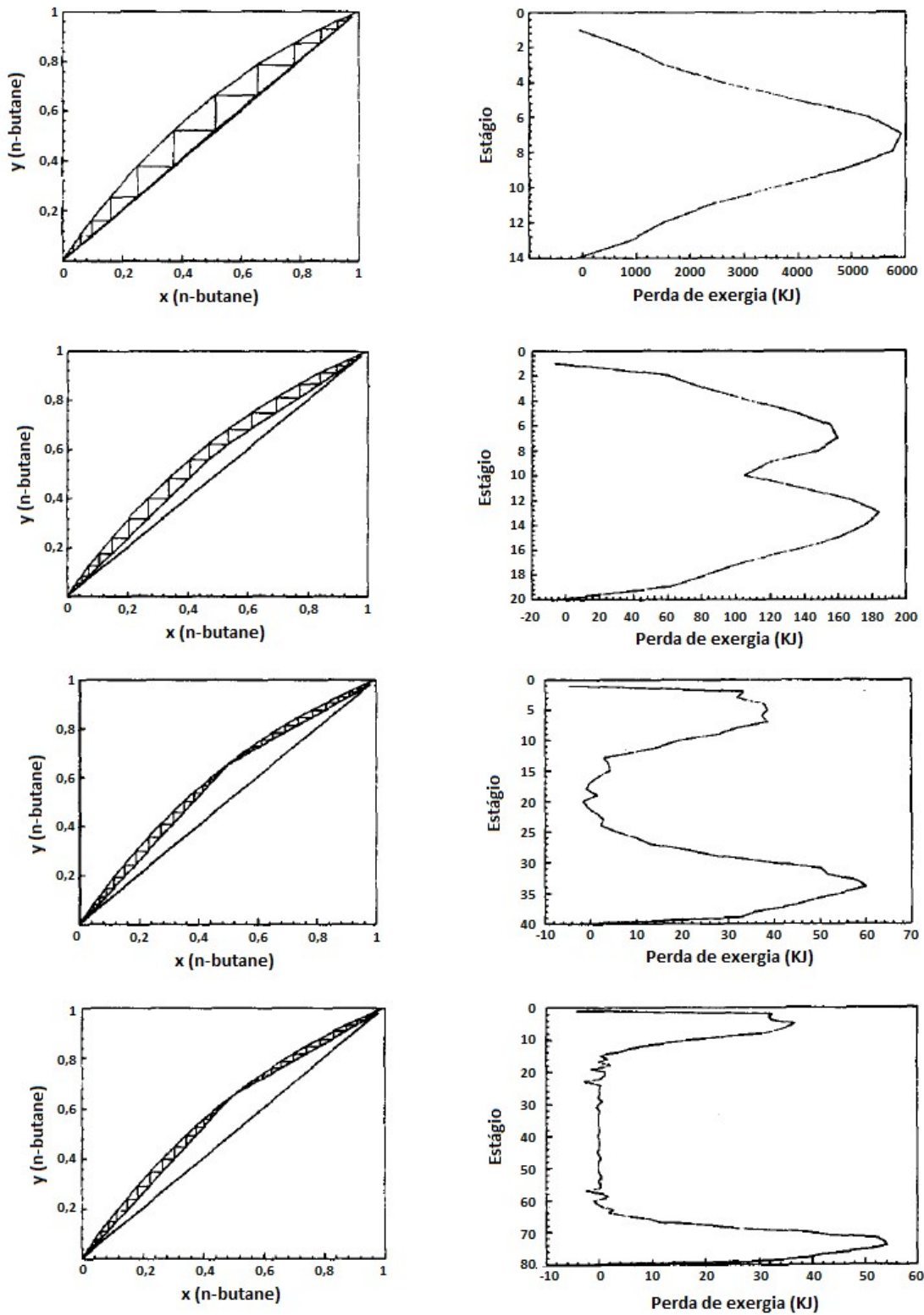


Figura 3.3: Gráfico de McCabe Thiele (contagem dos estágios a partir do condensador) e perfil de perda de exergia para diferentes razões de refluxo e número de pratos (adaptado de ZEMP; DE FARIA; OLIVEIRA MAIA, 1997).

Através do perfil de perda de exergia se observa com uma só variável as forças motrizes da transferência de calor e massa. Seções da coluna ou estágios individuais que operam ineficientemente são facilmente indentificados.

Estudos realizados por Zemp e de Faria (ZEMP; DE FARIA, 1995 apud ZEMP; DE FARIA; OLIVEIRA MAIA, 1997) concluíram que colunas de destilação que apresentam um perfil de perda de exergia distribuído equilibradamente são mais eficientes termodinamicamente que colunas com perfil não uniforme. Dessa forma, as modificações a serem realizadas na coluna visam obter uma distribuição mais equilibrada do perfil de perda de exergia. Ou seja, se deseja aproximar o valor da soma das perdas de exergia da seção de retificação à soma da seção de esgotamento da coluna.

Para se otimizar uma coluna através do perfil de perda de exergia, primeiro se gera o perfil para a coluna e se identifica a seção com maior perda de exergia, após, se modifica alguma condição na coluna a fim de se equilibrar as perdas de exergias nas regiões de retificação e esgotamento. Exemplos de modificações que são possíveis de realizar: alteração do estágio de alimentação na coluna, modificação das condições térmicas na corrente de alimentação, uso de trocadores laterais, entre outros.

A Figura 3.4 mostra o perfil de perda de exergia obtido para uma mistura binária equimolar de benzeno e tolueno que alimenta no 13^o estágio uma coluna de destilação com 26 estágios. A vazão de alimentação na coluna é de 1000 kmol/h, a vazão de destilado é de 500 kmol/h e a fração molar de benzeno no destilado é de 0,99. Na Figura 3.4(a), a corrente de entrada está a 1 bar e na condição de líquido saturado. Percebe-se que o perfil para a perda de exergia gerado não está equilibrado, sendo que a perda de exergia na seção de esgotamento (após o estágio de alimentação) é superior ao da seção de retificação. Observar que para essa figura, e para as demais seguintes de perfil de perda de exergia, o topo da coluna é representado na parte inferior do gráfico.

Na Figura 3.4(b), a corrente de entrada está a 1 bar e possui uma fração molar de vapor igual a 0,25. Com esta modificação se diminui a quantidade de líquido na alimentação, diminuindo a demanda térmica no refeedor da coluna e aumentando a demanda no condensador. Observa-se que o perfil gerado está mais equilibrado que o do caso (a).

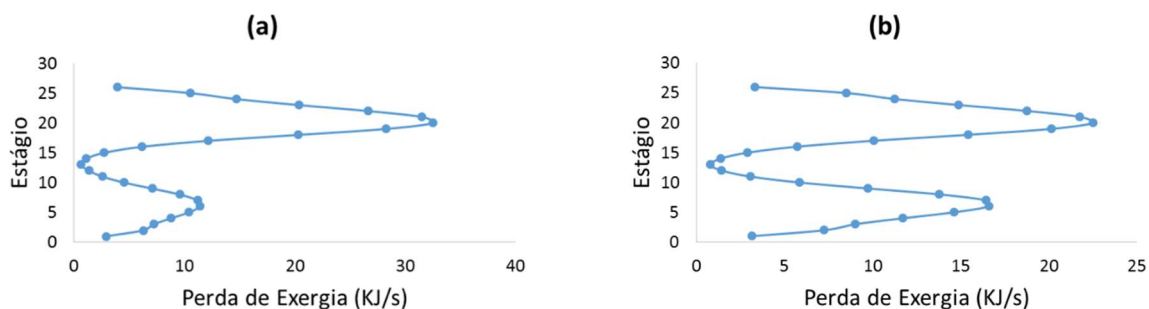


Figura 3.4: Perfil de perda de exergia para a) corrente de líquido saturado e b) fração molar de vapor de 0,25, na entrada da coluna

A sequência de modificações para otimização da coluna de destilação baseado no perfil de perdas de exergia, segundo utilizada por Moussa (MOUSSA, 2001), segue a sequência aplicada por Dhole e Linnhoff (DHOLE; LINNHOFF, 1993) baseada na CGCC, exceto que através do perfil de perda de exergia não possui a etapa de redução do refluxo. Os gráficos apresentados na Figura 3.5, Figura 3.6, Figura 3.7, Figura 3.8 e Figura 3.9 foram construídos em consonância aos estudos referenciados. As etapas consideradas para a otimização através do perfil de perda de exergia são:

1. Localização da alimentação

A posição da alimentação pode ser otimizada através do perfil de perda de exergia. O perfil ideal a ser obtido deve apresentar curvas suaves. Caso ocorra a presença de patamares no perfil, a posição da alimentação deve ser alterada. Se o patamar ocorrer na seção de esgotamento, o estágio de alimentação deve ser deslocado mais próximo do topo da coluna. A Figura 3.5 apresenta o perfil de perda de exergia obtido para a mesma coluna descrita no exemplo da Figura 3.4, sendo que a corrente neste novo caso está na condição de líquido saturado e alimenta a coluna no estágio 7, em Figura 3.5(a), e no estágio 13, em Figura 3.5(b). Por meio desta figura, percebe-se qual o estágio de alimentação ideal na coluna.

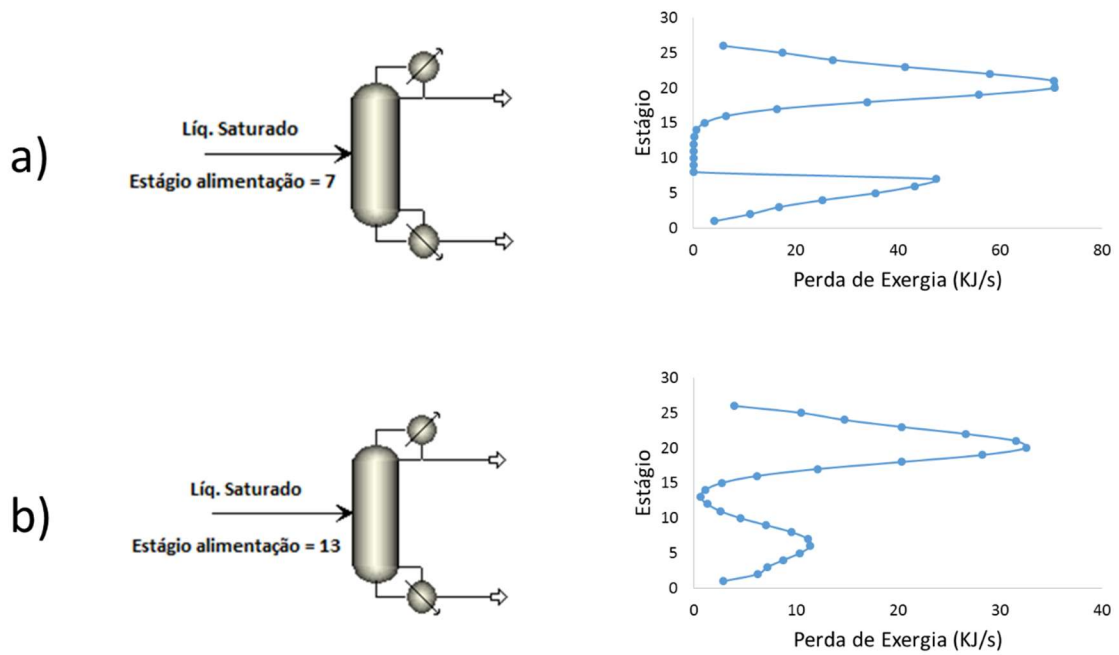


Figura 3.5: Perfil de perda de exergia para diferentes estágios de alimentação

2. Condição térmica da corrente de alimentação

Através da alteração da condição térmica da corrente de alimentação da coluna, busca-se equilibrar o tamanho dos picos de perda de exergia presentes na região de esgotamento e refino da coluna. O exemplo da Figura 3.4 descreve esta modificação.

3. Alocação de trocadores de calor laterais

Caso as etapas anteriores não forem suficientes para equilibrar as perdas de exergias entre as seções de esgotamento e retificação da coluna, utiliza-se trocadores de calor laterais em estágios específicos na coluna.

Para se determinar a localização do trocador de calor lateral observa-se o perfil de perda de exergia e localiza-se neste o maior pico de perda de exergia. Nesta região insere-se o trocador de calor lateral a fim de se diminuir o pico formado. Na seção de esgotamento da coluna o trocador de calor lateral será um refeedor. Já para a seção de retificação o trocador será um condensador. A Figura 3.6.a exibe o perfil de perda de exergia para a mesma condição apresentada no exemplo da Figura 3.4.b em comparação para um processo semelhante, onde a única diferença é o uso de um refeedor intermediário, Figura 3.6.b. Percebe-se, que com o uso do refeedor intermediário, conseguiu-se aproximar o tamanho dos picos de perda de exergia na seção de esgotamento e retificação da coluna. No exemplo

mostrado na Figura 3.6.b, fez-se uso de um refeedor intermediário no estágio 17, cuja carga térmica é 3 MJ/s. Como referência de valor para a carga térmica no trocador lateral, pode-se utilizar o valor da entalpia obtido através do perfil CGCC no estágio escolhido para se adicionar o trocador de calor. Esta diferença será o valor máximo de calor a ser adicionado no trocador.

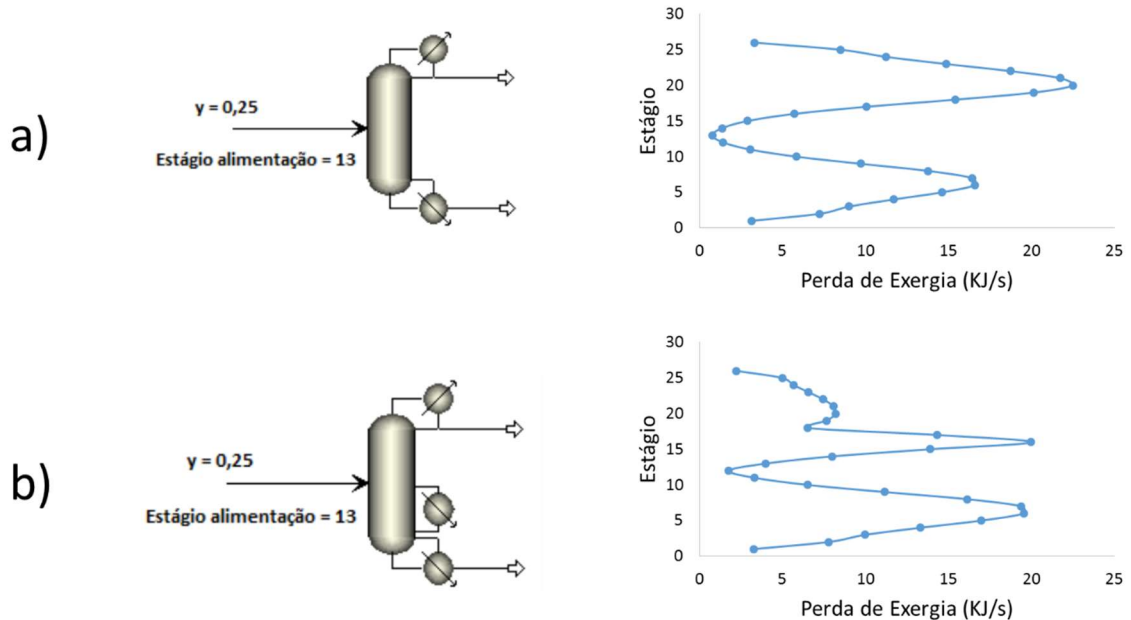


Figura 3.6: Perfil de perda de exergia para coluna sem e com trocador de calor lateral

O método proposto por Pinto et al. (PINTO et al., 2011) também é útil na otimização para o projeto de colunas de destilação. Nele se utiliza o perfil de perdas de exergia para se observar a perda mínima de exergia existente em algum estágio da coluna. Este valor mínimo é utilizado como referência e as modificações realizadas ao longo da coluna devem possuir perdas de exergia maior ou igual a desta referência. É desejável manter esta perda de exergia mínima, pois como a perda de exergia representa irreversibilidade na coluna e, sabendo-se que quanto menor a irreversibilidade menor também é a força motriz existente no estágio para se realizar a separação. Logo, quanto menor a perda de exergia, maior será o número de estágios na coluna necessários para se realizar a mesma separação, encarecendo a coluna. Respeita-se, portanto, a perda de exergia mínima já utilizada na coluna, pois assim se respeita também a relação de custo capital e custo operacional originalmente estipulada para a coluna.

A determinação da quantidade de calor a ser trocado em trocadores laterais e a quantidade de pratos a serem adicionados na coluna é efetuada respeitando a referência de perda de exergia mínima. Os exemplos a seguir demonstram o efeito

no perfil de perda de exergia devido a modificação do número de estágios e do calor adicionado em um refeedor lateral.

Observando a Figura 3.7, a qual se refere à mesma situação do exemplo da Figura 3.6, se determina em (a) o valor da perda de exergia mínima na coluna ($Ex_{loss,min}$). Verifica-se também que, com a utilização do refeedor lateral em (b), diminui-se o pico de perda de exergia da seção de esgotamento e aumenta-se o pico da seção de retificação.

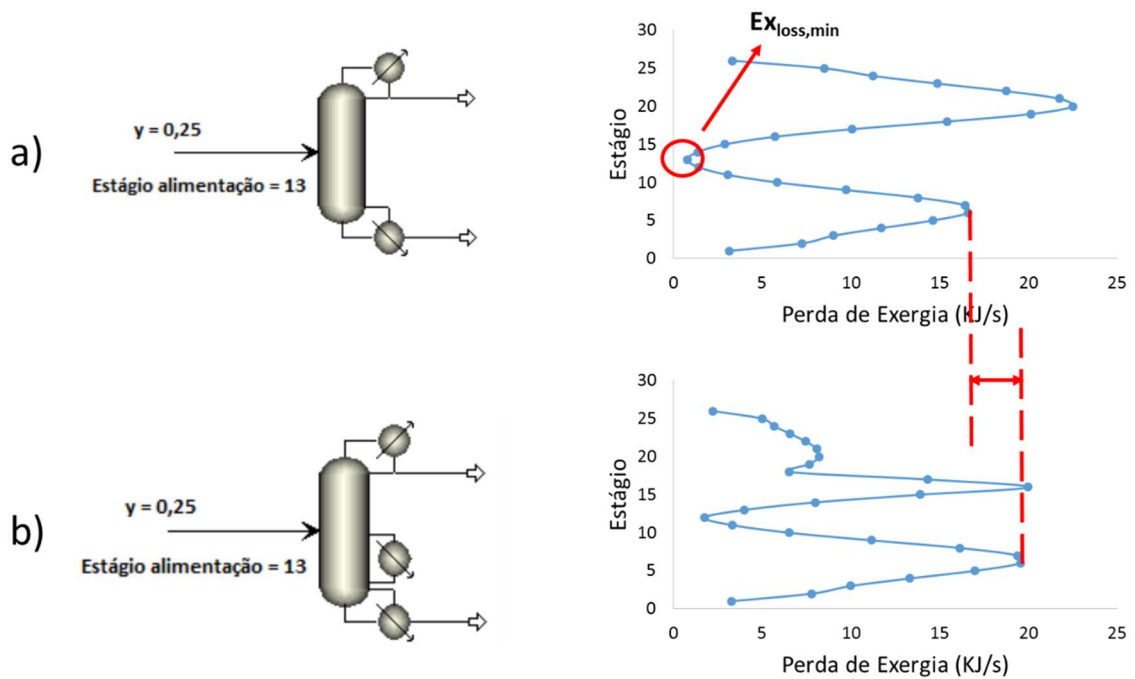


Figura 3.7: Perfil de perda de exergia para coluna sem e com trocador de calor lateral

No exemplo da Figura 3.8 aumentou-se o número de estágios na coluna a fim de se reduzir o aumento ocorrido no pico de perda de exergia na seção de retificação. Porém, nota-se que esta modificação causou uma diferença elevada entre os picos da seção de esgotamento e retificação. Também se observa que a exergia mínima, ($Ex_{loss,min}$), em (b) é inferior a em (a), indicando que o custo capital da coluna com esta modificação está mais elevado que à da coluna original.

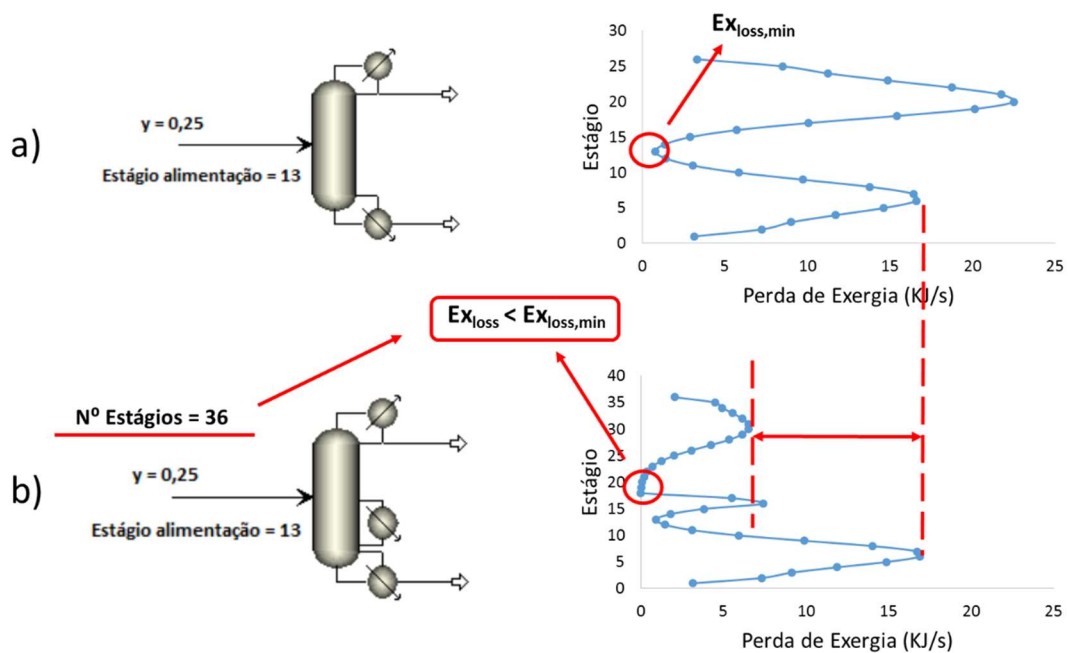


Figura 3.8: Alteração no perfil de perda exergia devido a adição de estágios na coluna

Para elevar o valor da Ex_{loss} da coluna acima da $Ex_{loss,min}$ e tornar os picos de perda de exergia mais equilibrados na coluna, deve-se variar com a quantidade de calor no trocador lateral e o número de estágios na coluna. Na Figura 3.9(b), em relação à Figura 3.8.b, o número de estágios foi reduzido de 36 para 27 e a quantidade de calor foi reduzida de 3 MJ/s para 1 MJ/s. Estes valores foram atingidos através de tentativa e erro. O resultado obtido na Figura 3.9(b) corresponde a uma coluna com picos de perda de exergia equilibrados e com um valor para $Ex_{loss,min}$ superior ao da coluna de referência Figura 3.9(a).

O objetivo do método proposto por Pinto et al. (PINTO et al., 2011) é obter para a coluna a quantidade de calor no trocador lateral e a quantidade de estágios adicionais mantendo-se as seguintes condições:

- A coluna modificada deve atender as condições de separação determinadas no caso original;
- Nenhum estágio da coluna modificada deve operar com Ex_{loss} maior ao $Ex_{loss,min}$ obtido na coluna original;
- A energia consumida pela coluna modificada não pode exceder à da coluna original.

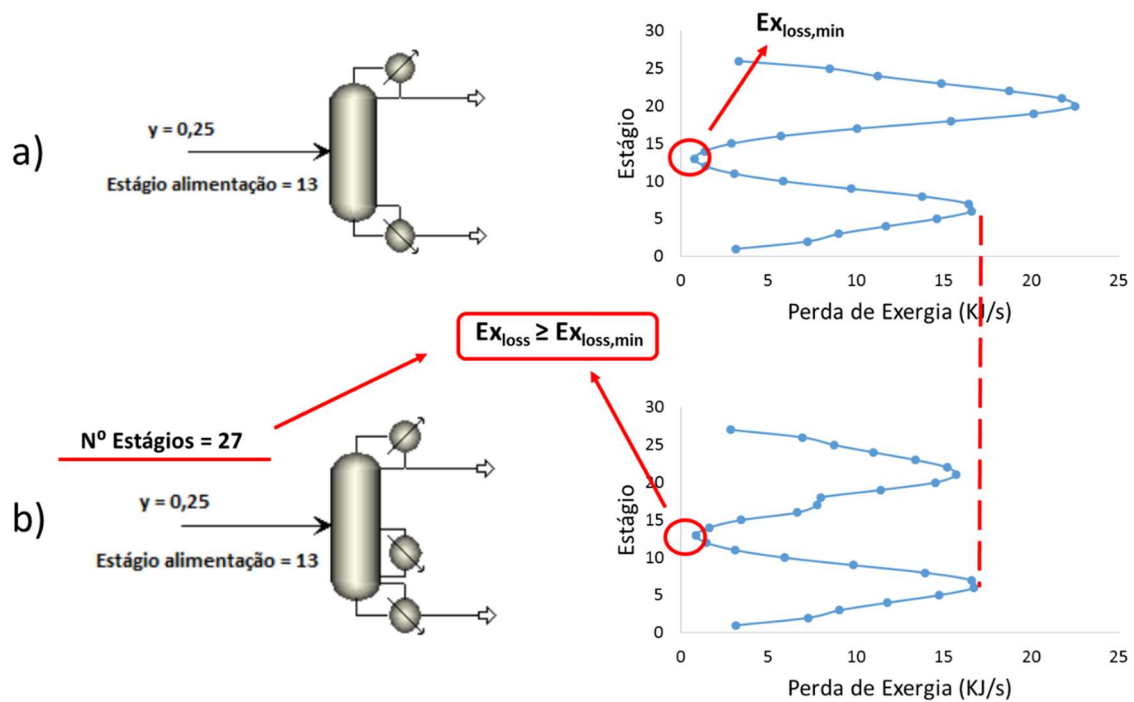


Figura 3.9: Alteração no perfil de perda exergia devido a ajustes no número de estágios da coluna e da carga térmica do refeedor lateral

3.1.3 Perda de Exergia para a Análise Termodinâmica de Plantas Industriais

A análise exérgica vista na seção 3.1.2 para colunas de destilação pode ser aplicada para a plantas industriais, sendo que as perdas de exergias seriam utilizadas para avaliar as irreversibilidades existentes nos diferentes equipamentos que constituem uma determinada planta. Verifica-se, que os estudos realizados focam em modificações no projeto original na planta industrial.

No estudo de Demirel e Nguyen (NGUYEN; DEMIREL, 2010) foi realizada a otimização de uma planta de produção de biodiesel. Neste artigo os autores utilizaram como recurso o perfil de perda de exergia e a CGCC das colunas existentes na unidade para propor modificações a fim de se obter ganhos econômicos e redução na emissão de CO_2 . Como resultado, este estudo, altera o projeto original da planta de biodiesel. As principais modificações realizadas foram:

- Adição de três novos trocadores de calor, sendo dois para pré-aquecimento da carga de suas respectivas colunas e o terceiro trocador para aquecimento lateral de outra coluna;
- Alteração no número de estágios em uma coluna;
- Alteração no estágio de alimentação de outra coluna;

- Modificação no alinhamento de tubulações de processo para integração energética.

Já em outro estudo de Demirel e Alhajji (ALHAJJI; DEMIREL, 2016), foi realizada a otimização de uma planta de etileno. A mesma lógica foi seguida neste caso, onde se avaliou as colunas de destilação existente na planta e através destas se concluiu quais são as modificações de projeto necessárias para se obter ganhos econômicos na planta industrial.

Shin et al. (SHIN; YOON; KIM, 2015), realizaram uma otimização operacional de três plantas de recuperação de gás natural com base na perda de exergia total da planta. Neste estudo foi definido uma função objetivo que visava minimizar as perdas de exergia dos diferentes equipamentos e instrumentos abrangidos pelos três diferentes processos. A otimização foi realizada com base em algoritmo genético e as variáveis de decisão foram definidas de acordo com uma análise de sensibilidade.

Apesar deste artigo apresentar uma otimização baseada em dados operacionais, percebe-se um viés de projeto das plantas industriais. Isto fica evidenciado pelo uso de duas colunas diferentes, uma com 10 estágios e outra com 30, nas unidades de processamento avaliadas. Também se realiza modificação no projeto da planta quando se alterou o posicionamento do refeedor lateral utilizado nas colunas.

Neste estudo, o uso do perfil de perdas de exergia da coluna ficou limitado à comparação de resultado obtido no caso otimizado em relação ao caso original. Não se identifica o uso do perfil para auxiliar na definição dos valores das variáveis de decisão.

Verifica-se através dos estudos mencionados a utilização da exergia tanto para determinar melhorias no projeto e operação de colunas de destilação, como também para se determinar melhorias em plantas industriais. Assim, concluiu-se que a exergia é considerada um instrumento apropriado para a avaliação da eficiência operacional e de projeto nas colunas, indicando modificações possíveis para a otimização da coluna e também para a avaliação da eficiência global de um determinado processo. Mas pouco se verifica na utilização desta variável para a otimização de uma planta química completa, onde a análise da perda de exergia seria o indicador dos equipamentos mais ineficientes compreendidos em uma unidade.

3.2 Métricas de eficiência operacional

O cenário econômico mundial e o atual patamar de preços do petróleo pressionam as empresas do ramo petroquímico a otimizar cada vez mais suas

unidades. Nesse cenário de estagnação econômica e redução de investimentos do setor, a operação ótima deixou de ser considerada uma melhoria no processo, mas sim uma necessidade para manter as unidades viáveis economicamente. Isso está levando as unidades industriais aos seus limites operacionais, considerando a necessidade crescente de produção, de modo a suprir a demanda de mercado, alinhado a margens de lucro cada vez menores. Deste modo, os processos industriais têm que ser cada vez mais eficientes, reduzindo, assim, as perdas durante a operação (SCHULTZ, 2015).

Para atender esta necessidade de eficiência operacional deve ser aplicada na planta industrial técnicas eficientes de controle e otimização de processos. Deste modo, torna-se possível a operação da planta em um ponto mais próximo de seu limite operacional. A busca desta operação ótima deve considerar as restrições de projeto dos equipamentos, os riscos ambientais, os limites ambientais legais de emissões dos resíduos/gases industriais e as especificações dos produtos. Neste contexto surgiram novas arquiteturas para controle de processos que visam o controle operacional da unidade e a sua otimização econômica (MIZOGUSHI; MARLINS; HRYMAK, 1995).

Na literatura se encontram diversas metodologias que possuem o objetivo de realizar esta otimização operacional, permitindo a automação da operação com a finalidade de maximizar o lucro da unidade operacional. Dentre estas metodologias destaca-se neste estudo a técnica de Real-time optimization (RTO) e Self-optimizing control (SOC).

Dentro destas técnicas existe a possibilidade da utilização para otimização da planta uma função objetivo baseada em análise termodinâmica da unidade de processo que atinja o ponto mínimo de perda de exergia.

3.2.1 *Real-Time Optimization (RTO)*

RTO visa otimizar a operação de processos considerando um ótimo econômico. RTOs são sistemas de controle de processos em malha fechada baseadas em modelagem computacional da unidade que tem por finalidade manter a operação o mais próximo possível do ponto ótimo operacional (TRIERWEILER, 2014).

Vários fatores de desenvolvimento tecnológico ocorridos durante meados da década de 1980, permitiram pela primeira vez a utilização de RTO baseado em modelagem rigorosa em estado estacionário de processo. Alguns destes fatores são o aumento da capacidade computacional utilizada, a modelagem de operações e a tecnologia de controle MPC (Model Predictive Control) (DARBY et al., 2011). A partir da década de 1990 apresentou um significativo aumento na aplicação do RTO em processos industriais (ALKAYA; VASAMTHARAJAN; BIEGLER, 2008).

A aplicação do RTO em unidades possui o benefício de melhorar economicamente a operação com a diminuição do consumo de energia e de emissões poluentes. Também é possível se obter benefícios indiretos com um entendimento aprimorado do RTO e do processo em questão. Exemplos desses benefícios indiretos são (TRIERWEILER, 2014):

- Redução de dificuldades operacionais;
- Informações de medições anormais obtidas por detecção de erros: auxiliam engenheiros de processo e instrumentação na resolução de problemas causadas por medições incorretas por instrumentos;
- Auxílio na estimação de parâmetros: útil para os engenheiros de processo avaliarem as condições dos equipamentos e identificar a perda de eficiência;
- Modelagem computacional da unidade de processo: a modelagem utilizada pelo RTO na otimização pode ser usada para o monitoramento de processo e para o treinamento de novos operadores.

O desenho típico da estrutura de automação de uma unidade é composto por uma pirâmide com diversas camadas, onde as camadas inferiores trabalham com dados de nível operacional e à medida que se sobe na pirâmide, os dados operacionais são substituídos por informações gerenciais. Um desenho típico dessa arquitetura é mostrado na Figura 3.10, onde os dados das variáveis de processo se concentram no nível 1 e, à medida que é realizada a transferência de informações ao longo dos demais níveis, esses dados são informações importantes para o planejamento estratégico da empresa (SCHULTZ, 2015).

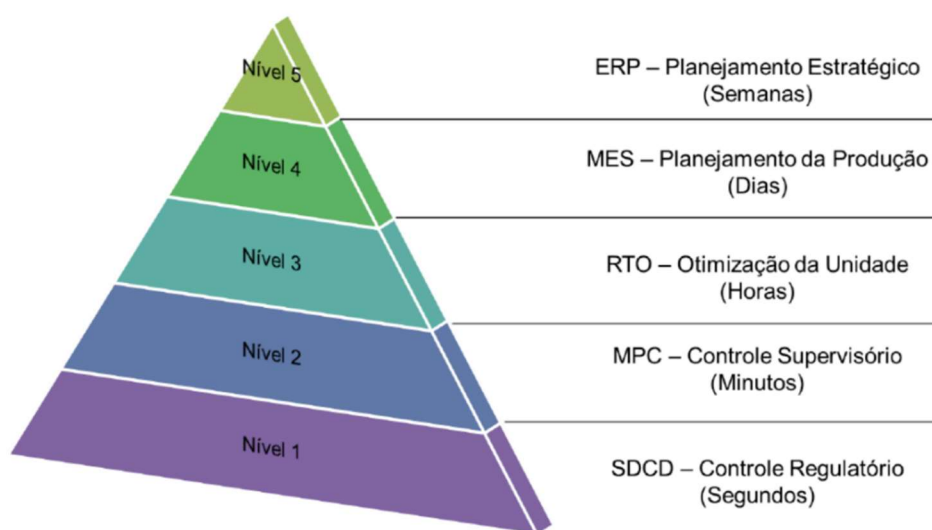


Figura 3.10: Arquitetura típica de automação industrial (SCHULTZ, 2015)

Esta pirâmide representa a hierarquia de controle de uma planta industrial. Cada uma das camadas que compõe a pirâmide possui objetivos de controle e tempo de atuação diferentes. As camadas inferiores possuem função de operação da planta e seu tempo de atuação são menores. As camadas superiores são responsáveis pelo planejamento de produção da planta e são definidas de acordo com a estratégia comercial da empresa.

A camada da pirâmide que representa o RTO realiza a comunicação entre as camadas que realizam a operação (camadas inferiores) com as camadas que realizam o planejamento de produção (camadas superiores). Como pode ser visto na Figura 3.10 e Figura 3.11, o sistema RTO deve possuir ligação com a camada de planejamento, de onde são obtidos os dados relativos à qualidade de produtos e matérias primas, custos das correntes e restrições relativas às especificações e demandas de produção. Por outro lado, na camada inferior, o sistema deve possuir comunicação com o sistema de controle industrial para obter os valores das principais variáveis do processo, e também, para poder repassar os setpoints das variáveis controladas que otimizam o custo da operação (ENGELL, 2007).

A estrutura típica de um sistema RTO é uma extensão de um sistema de controle feedback e é constituído por quatro subsistemas (SEQUEIRA S.E.; GRAELLS, 2002):

- Detecção de estado estacionário: esta etapa identifica se o processo está em regime permanente. A próxima etapa será realizada apenas quando o processo estiver em estado estacionário.
- Reconciliação de dados: uma vez que o processo atinja o estado estacionário, os dados da planta são coletados e validados para evitar erros oriundos de medições incorretas. As medições podem ser reconciliadas através do uso de balanços de massa e energia para garantir a consistência dos dados a serem utilizados na próxima etapa de atualização do modelo do processo.
- Atualização da modelagem do processo: as medições validadas na etapa anterior são utilizadas para estimar os parâmetros do modelo do processo para garantir que o modelo representa corretamente a planta industrial no seu atual ponto de operação.
- Otimização do modelo e implantação de solução na planta: são calculados os *setpoints* ótimos dos controladores da planta. Estes cálculos são realizados baseando-se no modelo atualizado da planta. Os novos *setpoints* calculados são validados pelo sistema supervisor (normalmente com a participação da equipe de operação) e é realizada uma nova verificação do processo, de modo a verificar se, durante o cálculo, houve mudança no estado estacionário. Caso nenhuma mudança

significativa tenha ocorrido, é realizada atualização dos valores dos *setpoints* das variáveis controladas.

A Figura 3.11 ilustra a representação da arquitetura de funcionamento de um RTO.

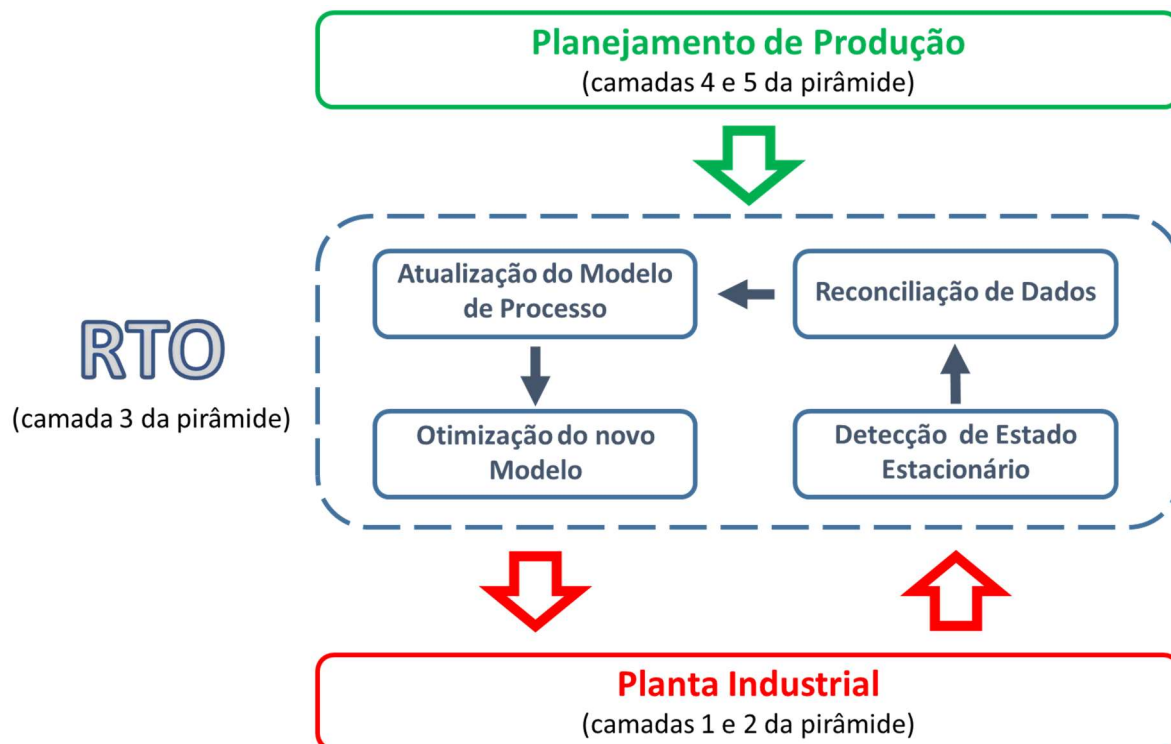


Figura 3.11: Arquitetura de funcionamento de um RTO

A otimização proposta neste trabalho pode ser inserida na etapa de otimização pertencente à estrutura típica de um sistema RTO.

3.2.2 *Self-optimizing control (SOC)*

A metodologia SOC está baseada na escolha de variáveis de controle que possam possuir *setpoint* constante. No caso do sistema RTO ocorre o cálculo de um algoritmo de otimização que encontra o valor ótimo para as variáveis de controle, definindo-se, portanto, novos *setpoints*. Para possibilitar isso, o sistema RTO apresenta uma estrutura de software mais complexa que execute as etapas necessárias, de acordo com a Figura 3.11. Já para o caso do SOC o objetivo é atingir um desempenho econômico aceitável mantendo-se as variáveis de controle constantes (SKOGESTAD, 2000a). Isso permite que o problema complexo de otimização do caso de um RTO seja substituído por um problema simples de malha feedback.

Conforme a definição apresentada por Skogestad (SKOGESTAD, 2000a) o Controle auto-otimizável (SOC) ocorre quando se pode alcançar uma perda aceitável com valores constantes para as variáveis controladas (sem a necessidade de re-otimizar quando um distúrbio ocorre).

A perda aceitável referida na definição de SOC é devido a utilização de um valor fixo para as variáveis controladas, mesmo após a presença de distúrbio. Após este, a planta não estará mais operando em seu ponto ótimo pois não é realizado uma nova otimização considerando o distúrbio. Por outro lado, o SOC apresenta a vantagem de operar tanto em estado estacionário como em regime transiente, sendo esta uma das principais vantagens em relação ao sistema RTO (SCHULTZ, 2015).

Para auxiliar na escolha de boas candidatas para variáveis de controle os seguintes requisitos devem ser satisfeitos (SKOGESTAD, 2000b):

- Requisito 1: o valor ótimo da variável deve ser insensível aos distúrbios do processo; assim, erros no *setpoint* se tornam pequenos.
- Requisito 2: o valor da variável deve ser fácil de se medir e de se controlar, assim o erro de implementação é reduzido.
- Requisito 3: o valor da variável deve ser sensível a mudanças no valor da variável manipulada, com um grande ganho relativo.
- Requisito 4: para os casos de duas ou mais variáveis controladas, essas variáveis não devem possuir grande correlação.

Considerando a arquitetura típica de automação industrial representada na Figura 3.10, atuação da metodologia SOC estaria nas camadas inferiores 1 e 2 da pirâmide.

A utilização do estudo presente nesta dissertação em um controle SOC seria o uso de uma variável virtual referindo-se a exergia do processo como variável de controle.

3.3 Unidade de Processamento de Gás Natural (UPGN)

O gás natural é um combustível fóssil com alta eficiência energética e oferece importante benefício em economia de energia quando comparado com óleo ou carvão. Embora o uso primário do gás natural ser o de gás combustível, ele também é fonte de hidrocarbonetos, que são matérias-primas para indústrias petroquímicas, e enxofre, substância importante para indústria química (MOKHTAB; POE; MAK, 2015). Existem diferentes teorias quanto às origens dos combustíveis fósseis. A

teoria mais amplamente aceita da origem do gás natural pressupõe que os hidrocarbonetos deste gás provêm de matéria orgânica (os restos de terra e plantas aquáticas, animais e microorganismos) que foram presos nos sedimentos enquanto eram depositados e transformados por longos períodos de tempo em sua forma atual (MOKHTAB; POE; MAK, 2015).

A Tabela 3.1 apresenta a composição típica do gás natural. Esta composição indica que metano é o componente mais abundante na mistura gasosa. Existem também a presença de compostos inorgânicos como dióxido de carbono e sulfato de hidrogênio, os quais não são desejados pois não são combustíveis e causam problemas na unidade de processamento, por exemplo: corrosão (GUO; GHALAMBOR, 2005).

Tabela 3.1: Composição típica do gás natural (GUO; GHALAMBOR, 2005)

Componentes	Fração Molar
Metano	0,8407
Etano	0,0586
Propano	0,0220
i-Butano	0,0035
n-Butano	0,0058
i-Pentano	0,0027
n-Pentano	0,0025
Hexano	0,0028
Heptano e mais pesados	0,0076
Dióxido de Carbono	0,0130
Sulfato de Hidrogênio	0,0063
Nitrogênio	0,0345

A complexidade e configuração de uma planta de processamento de gás natural depende da composição do gás e do tratamento e processamento requerido para se atingir as especificações dos produtos formados e os limites legais de emissões (MOKHTAB; POE; MAK, 2015). A Figura 3.12 demonstra dois esquemas simplificados de processamento de gás natural. O primeiro esquema tem com função remover condensados, enxofre e componentes pesados para se atingir a especificação de gás

comercial. O segundo esquema é para processar o gás a fim de se obter GLP e outros componentes com maior valor agregado. O gás residual deste segundo processo pode ter diferentes destinos, como: ser enviado para uma planta de liquefação de gás natural, ou pode ser utilizado como combustível para a planta industrial ou também, pode ser matéria-prima de plantas petroquímicas.

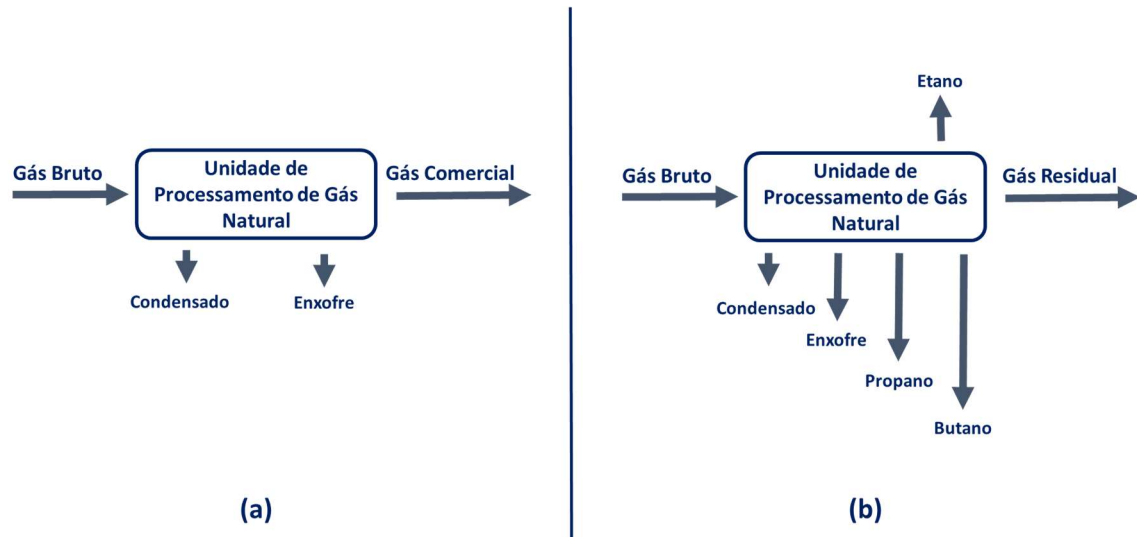


Figura 3.12: Duas configurações diferentes para unidades de processamento de gás natural

O ponto mais importante de uma unidade de processamento de gás natural é o sistema de refrigeração, responsável pela liquefação dos componentes pesados do gás natural. O processo escolhido nessa etapa define o tipo de unidade a ser utilizada. No Brasil, encontram-se principalmente unidades de turbo-expansão e absorção refrigerada. O processo de turbo-expansão é atualmente o mais eficiente e mais utilizado, consistindo basicamente na refrigeração e expansão do gás em uma turbina para liquefação dos hidrocarbonetos mais pesados, podendo-se atingir temperaturas abaixo de $-95\text{ }^{\circ}\text{C}$. O processo de absorção refrigerada consiste primeiramente na condensação dos componentes mais pesados por refrigeração e, após essa etapa, na absorção do restante através da lavagem do gás natural com um solvente em contracorrente (VAZ; MAIA; DOS SANTOS, 2008 apud FLECK, 2012).

No estudo de caso realizado neste trabalho, a unidade de processamento de gás natural em questão se assemelha com o da Figura 3.12.b e possui um turbo-expansor. Uma explicação mais detalhada sobre esta unidade é apresentada no capítulo 4. Neste estudo buscou-se uma otimização operacional de uma unidade de processamento de gás natural com base em uma análise de exergia da planta.

Embora existam benefícios consideráveis da utilização da análise exérgica no projeto de processos químicos, poucas tentativas foram aplicadas desta análise para o projeto e otimização de processos de processamento de gás natural. Isto é principalmente devido a complexibilidade das colunas utilizadas neste processo,

sendo ainda que estas colunas são altamente integradas com outras operações unitárias presentes na planta (SHIN; YOON; KIM, 2015).

4 Metodologia Proposta

A metodologia utilizada neste estudo é baseada na abordagem proposta por Shin et al. (SHIN; YOON; KIM, 2015), onde foi realizada a avaliação de uma otimização cujo objetivo visava minimizar a perda de exergia existente em uma planta de recuperação de gás natural. Neste estudo de Shin, realizou-se a otimização definindo variáveis de processo para as variáveis de decisão, caracterizando uma otimização operacional. Mas esta otimização foi conduzida para diferentes arranjos de processo, ou seja, houve alterações no projeto original da planta. Essas variações no arranjo original foram realizadas com base na análise de perda de exergia da planta. Sendo assim, este estudo apresenta um viés para melhoria no projeto da unidade estudada.

Esta dissertação pretende avaliar a otimização exclusivamente operacional de uma Unidade de Processamento de Gás Natural (UPGN). Deste modo, deseja-se avaliar se a otimização com base na menor perda de exergia da unidade operacional acarreta melhor desempenho energético da planta. Não foi possível concluir através dos artigos estudados a relação direta entre a otimização energética e a otimização para minimizar a perda de exergia.

Na maioria dos estudos, referenciados nesta dissertação, a análise exérgica é utilizada para melhoria no projeto da planta. Assim, se realizam modificações nos equipamentos da planta, normalmente nas colunas de destilação. No estudo do Shin, onde foi realizada uma otimização operacional com base na análise de exergia, não é possível concluir se esta otimização gerou economia de energia, pois não são

apresentados os dados de consumo de energia da planta original em relação à otimizada. Para poder se obter este resultado de economia de energia em uma planta otimizada com base na análise de exergia, esta dissertação realizou a metodologia descrita nos itens seguintes.

4.1 Modelagem do Processo

A primeira etapa da metodologia aplicada consiste em se elaborar o modelo do processo da planta industrial que se deseja estudar. Isto pode ser realizado através de algum software comercial de simulador de processos. Neste software, deve-se selecionar um pacote termodinâmico apropriado para a composição das correntes abrangidas no modelo. Após finalizar o modelo, é importante realizar a validação do mesmo com alguma referência, como por exemplo, com os dados reais da planta de referência para o modelo. Neste estudo se modelou a UPGN no software comercial Aspen Plus 8.8, conforme apresentado no capítulo 6.1.

4.2 Interface em Python

A segunda etapa da metodologia é a elaboração de uma interface entre o software simulador do processo e o software que executará o algoritmo de otimização.

No caso desta dissertação, os resultados gerados pela simulação em Aspen Plus foram exportados para o software Python. A interface criada entre estes dois programas possibilitou controlar a simulação em Aspen através do Python. Com isso, foi possível fazer uso de diversos recursos avançados desenvolvidos em pacotes específicos adicionados no Python.

Python é uma linguagem de programação de alto nível, desenvolvida com o propósito de uso geral. Devido às suas características, ela é principalmente utilizada para processamento de textos, dados científicos e criação de programas para interface com páginas dinâmicas para a web. Atualmente o Python possui um modelo de desenvolvimento comunitário e aberto.

Para tornar possível importar os dados da simulação, e posterior realização de cálculos através do Python, foram utilizados os seguintes pacotes desenvolvidos para o Python:

- NumPy (versão 1.11.1): pacote que permite ao Python trabalhar com computação numérica, realizando operações vetoriais e matriciais.

- Pandas (versão 0.18.1): linguagem desenvolvida para manipulação e análise de dados. Oferece a possibilidade de estruturar dados e realizar operações matemáticas com séries e tabelas numéricas.
- SciPy (versão 0.17.1): é uma biblioteca utilizada em computação técnica e científica. Possui módulos para otimização, álgebra linear, integração, interpolação, equações diferenciais, entre outros recursos focados na produção científica e de engenharia.
- Matplotlib (versão 1.5.1): é uma biblioteca que conta com diversos recursos de construção gráfica.
- DEAP (versão 1.0): DEAP é sigla para *Distributed Evolutionary Algorithms in Python* e é uma estrutura de cálculos computacionais que abrangem técnicas de computação evolucionária, como algoritmos genético, programação genética, PSO (*Particle Swarm Optimization*), entre outros.

Os códigos escritos em Python 3.5 utilizados na dissertação estão apresentados no capítulo dos Apêndices desta dissertação.

As principais rotinas executadas através do Python foram:

- Interface com Aspen Plus;
- Importação de dados de processo obtidos no simulador;
- Cálculos para se obter o consumo de energia total na planta;
- Cálculos para se obter o total de perda de exergia na planta;
- Execução do algoritmo de otimização, sendo o Python responsável por alterar as variáveis de decisão no Aspen Plus.

4.3 Otimização

A última etapa da metodologia consiste em aplicar um algoritmo de otimização apropriado para o caso a ser estudado, onde deve se definir as funções objetivos, para otimização energética e a de perdas de exergia, com base nos equipamentos e instrumentos envolvidos na planta modelada.

Este estudo tem o objetivo avaliar se a otimização visando obter um mínimo de perdas de exergia na UPGN se reflete em menor consumo de energia da unidade. Assim, efetuou-se quatro otimizações, duas delas com a função objetivo buscando um mínimo de perda de exergia, sendo uma para a composição do gás bruto original

e a outra para a composição rica em pesados. As outras duas otimizações possuem o objetivo de se atingir um mínimo consumo de energia, também sendo uma otimização para a composição original do gás bruto e a outra para a composição rica. Os resultados de otimização obtidos para cada caso de composição do gás bruto foram comparados.

A função objetivo utilizada na otimização para minimizar a perda de exergia da planta está descrita na Equação (4.1). Nesta são somadas as perdas de exergia individuais de cada equipamento/instrumento da planta modelada, no caso a UPGN. Os cálculos para perda de exergia dos equipamentos são efetuados conforme a Tabela 2.1.

$$f_{min} = \sum_{Equip} Ex_{loss} \quad (4.1)$$

Já para a otimização com objetivo de minimizar o consumo de energia deve-se considerar apenas os equipamentos que consomem energia de utilidades da planta, conforme descrito na Equação (4.2)

$$f_{min} = \sum_{Equip} |En| \quad (4.2)$$

As variáveis a serem modificadas pelo algoritmo de otimização deverão ser definidas de acordo com algum critério estabelecido. No estudo elaborado por Shin et al. (SHIN; YOON; KIM, 2015), estas variáveis foram definidas de acordo com um estudo de análise de sensibilidade, onde se concluiu quais eram as variáveis com impactos mais relevantes na planta estudada de recuperação de gás natural. Em plantas que tenham colunas de destilação, a elaboração da CGCC e do perfil de perdas de exergia da coluna auxiliam na definição das variáveis relevantes a serem consideradas na otimização. Com base no modelo estudado e nas variáveis selecionadas, devem ser definidas as restrições destas variáveis de decisões.

O método de otimização, as funções objetivos, as variáveis de decisões e suas restrições, utilizadas neste estudo são apresentadas no capítulo 6.2.

5 Descrição do Estudo de Caso

A Unidade de Processamento de Gás Natural (UPGN) no qual foi baseada a elaboração do modelo em Aspen Plus pertence a uma refinaria brasileira. Esta unidade tem como objetivo a recuperação e separação dos hidrocarbonetos existentes na corrente de gás natural. Os principais produtos obtidos neste processamento são:

- Gás Industrial e Gás Combustível;
- Gasolina Natural;
- GLP;
- Propano.

Uma característica da UPGN estudada é que ela apresenta um turbo expansor dentro do sistema de refrigeração. Para o gás chegar ao turbo expansor com a menor temperatura possível, o gás passa por diversos vasos de separação gás-líquido e trocadores de calor ao entrar na unidade.

Para promover a separação desejada na UPGN, esta unidade é dividida em dois sistemas, sendo eles:

- Condicionamento do gás natural;
- Sistema de separação dos produtos.

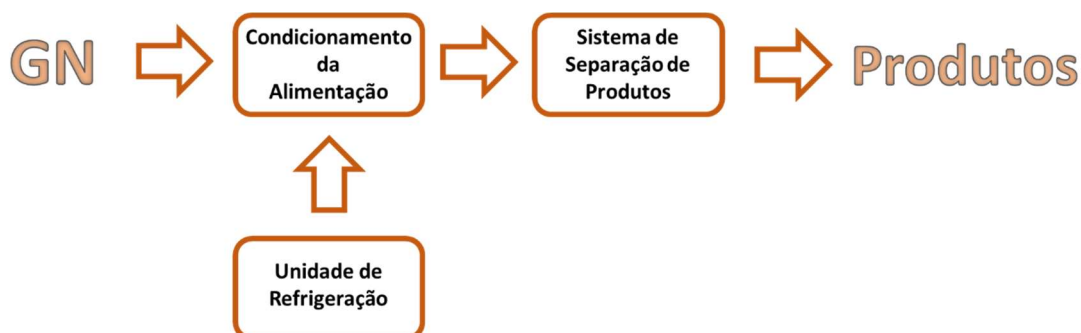


Figura 5.1: Diagrama de blocos simplificado da UPGN

5.1 Condicionamento do Gás Natural

O condicionamento do gás natural viabiliza a posterior separação dos produtos, sendo a etapa de refrigeração realizada no início do condicionamento a principal etapa para condicionar em temperatura e pressão o gás natural e viabilizar a separação nas colunas de destilação.

O processamento do gás natural ocorre através de uma diminuição de sua temperatura e pressão para promover a condensação dos compostos mais pesados. Ou seja, para viabilizar a separação dos diferentes produtos nas colunas de destilação é necessário que a pressão de operação destas sejam suficientemente baixas para facilitar a separação dos diferentes produtos desejados.

Na Figura 5.2 estão ilustradas estas situações. Nesta figura tem-se o diagrama de temperatura e composição binário, para os componentes etano e propano, a uma pressão constante de 69 bar, a qual corresponde a pressão do gás natural na entrada da UPGN e também se tem a curva para o etano e propano a 25,82 bar, valor correspondente a operação da coluna. Observa-se pelo gráfico, que na condição de 69 bar, a região de equilíbrio líquido-vapor é praticamente inexistente, pois esta região é formada por duas curvas coincidentes. Isto resulta em dificuldade de separação dos componentes. Já para a condição de operação a 25,82 bar a região de equilíbrio líquido-vapor, formada entre as curvas verde e azul, é muito maior em relação a da região de equilíbrio formada na pressão de 69 bar.

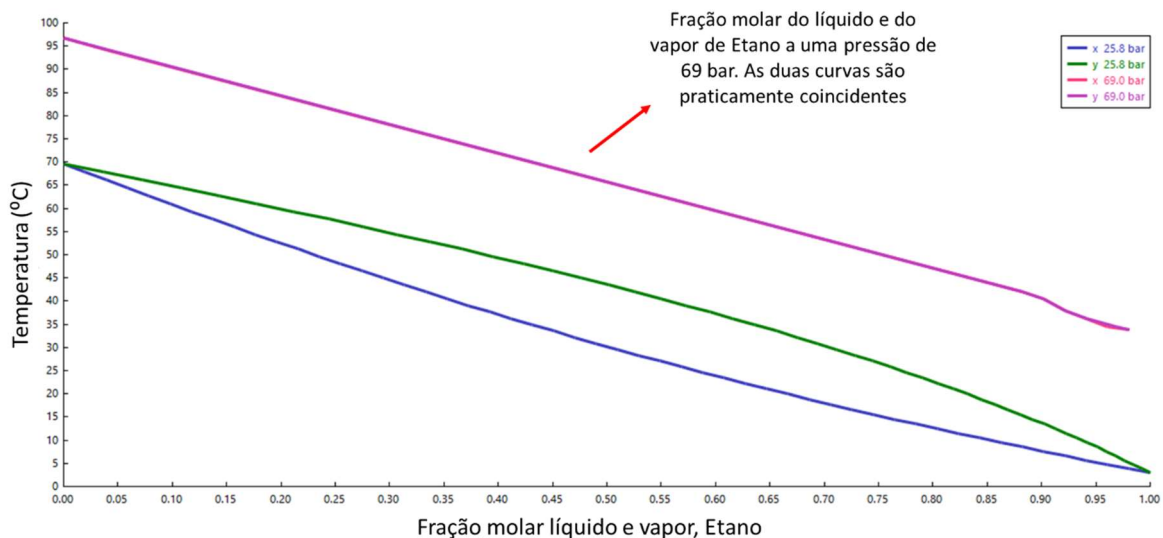


Figura 5.2: Diagrama de temperatura versus composição a pressão constante à 69 bar e à 25,82 bar

Portanto, o condicionamento do gás natural se faz necessário para se viabilizar a separação nas colunas de destilação. Salienta-se que a variação de temperatura

apresentada na coluna desetanizadora é maior do que a faixa de temperatura apresentada na Figura 5.2 na condição de 25,82 bar. Isto porque na figura estão representados apenas os componentes chaves da separação, etano e propano. Na coluna têm-se outros componentes mais leves e pesados que os chaves tornando a variação de temperatura maior.

O condicionamento do gás natural é realizado pelos equipamentos exibidos na Figura 5.3 (com exceção da coluna T-01).

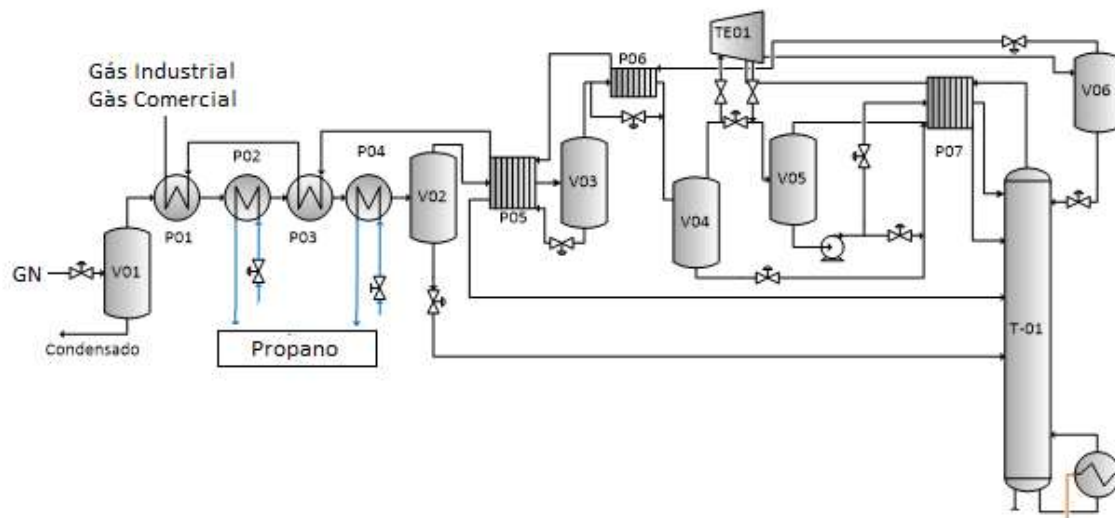


Figura 5.3: Fluxograma com os equipamentos que realizam o condicionamento do gás natural no estudo de caso avaliado nesta dissertação

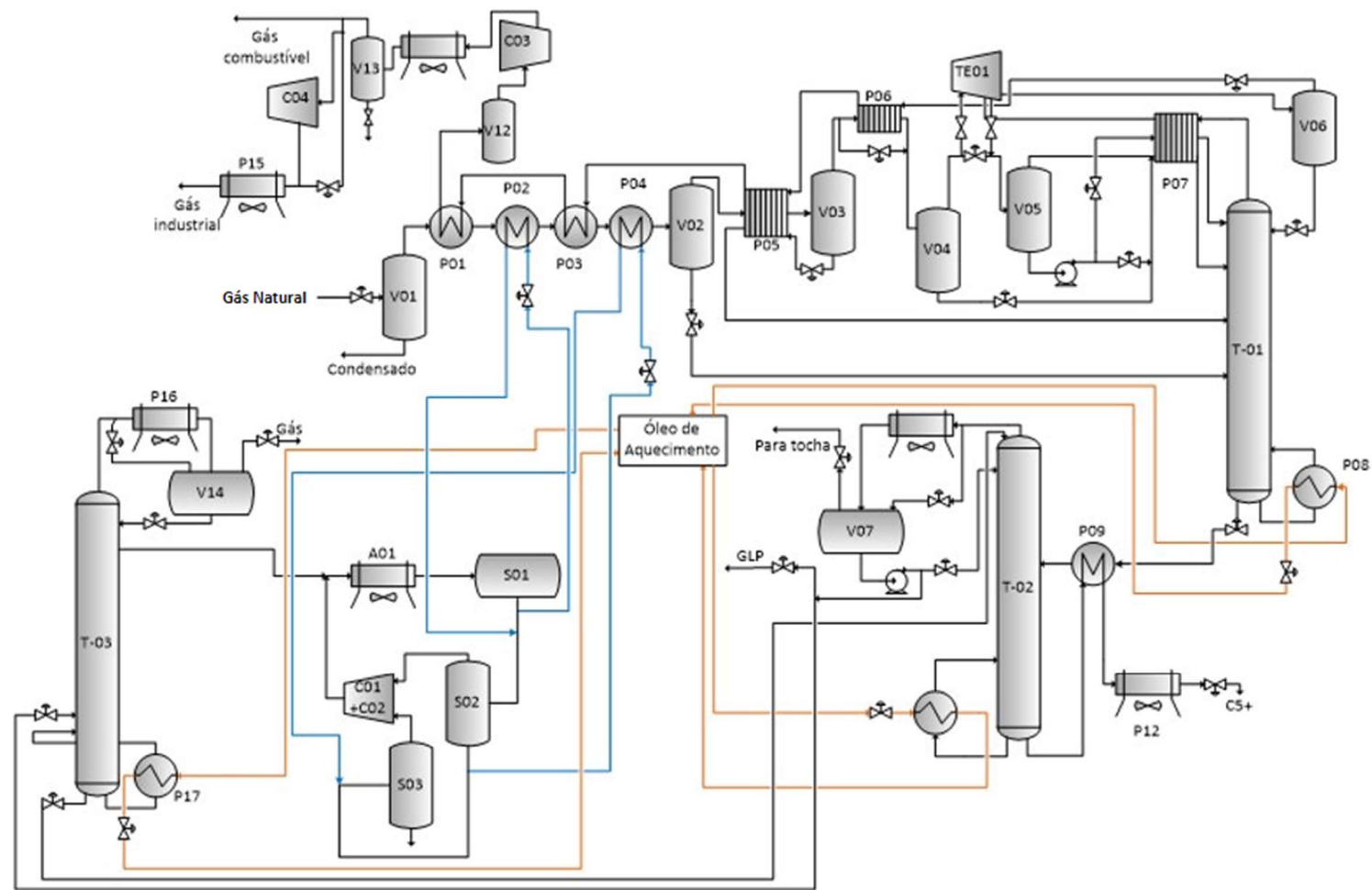


Figura 5.4: Esquema simplificado do processamento de gás natural e interligação com o sistema de refrigeração e óleo térmico (BARALDI, 2015)

Logo no início da UPGN, processo representado na Figura 5.4, o gás natural sofre uma diminuição drástica de temperatura através de uma série de quatro trocadores de calor (P01 a P04). Nestes quatro trocadores a corrente quente é o gás natural. A corrente fria nos trocadores P01 e P03 é o produto de topo da coluna T01. Já a corrente fria nos trocadores P02 e P04 é propano oriundo da Unidade de Refrigeração.

Nos vasos V02, V03 e V04 se realiza a remoção da fração pesada pelo fundo. Isto permite uma diminuição maior de pressão no turboexpansor sem que ocorra condensação no seu interior. Os produtos de fundo dos vasos são encaminhados em diferentes pratos da coluna T01 de acordo com a composição entre a corrente e a composição do prato. Entre os vasos têm-se dois trocadores de calor, P05 e P06, sendo que a corrente fria é o produto de topo da coluna T01. Estes trocadores condensam parcialmente o produto de topo dos vasos V02 e V03.

O turboexpansor TE01 efetua a redução de pressão do gás natural até uma pressão próxima da de operação da coluna T01. A temperatura do fluido na saída do turboexpansor é de aproximadamente $-77\text{ }^{\circ}\text{C}$, sendo inferior à temperatura no topo da coluna, $-60\text{ }^{\circ}\text{C}$. Assim, aproveita-se a temperatura baixa deste fluido após o turboexpansor, para condensar parcialmente o produto de topo da coluna T01 no trocador de calor P07.

O trabalho no turboexpansor realizado pela expansão do fluido é aproveitado para compressão do gás industrial/combustível no compressor C03.

5.2 Sistema de Separação de Produtos

O objetivo deste sistema é realizar a separação dos produtos de interesse comercial. Este sistema é constituído principalmente por colunas de destilação. A Figura 5.5 apresenta um fluxograma simplificado deste sistema com os principais equipamentos envolvidos e os produtos gerados.

Ao entrar no Sistema de Separação de Produtos, a corrente é alinhada para a coluna desetanizadora (T01), onde ocorre a separação do metano e do etano das frações mais pesadas. A corrente de topo é composta principalmente por metano e etano, e segue para os trocadores de calor P06, P05, P03 e P01, onde é aproveitada a baixa temperatura da corrente para resfriar a corrente que alimenta a UPGN. Após passar pelos trocadores, a corrente é comprimida e passa por um vaso de separação. Uma parte dessa corrente é destinada a utilização como gás combustível e a outra é comprimida novamente, ocorrendo recuperação de pressão para posterior utilização como gás industrial. Esses produtos são vendidos ou utilizados nos fornos da própria unidade.

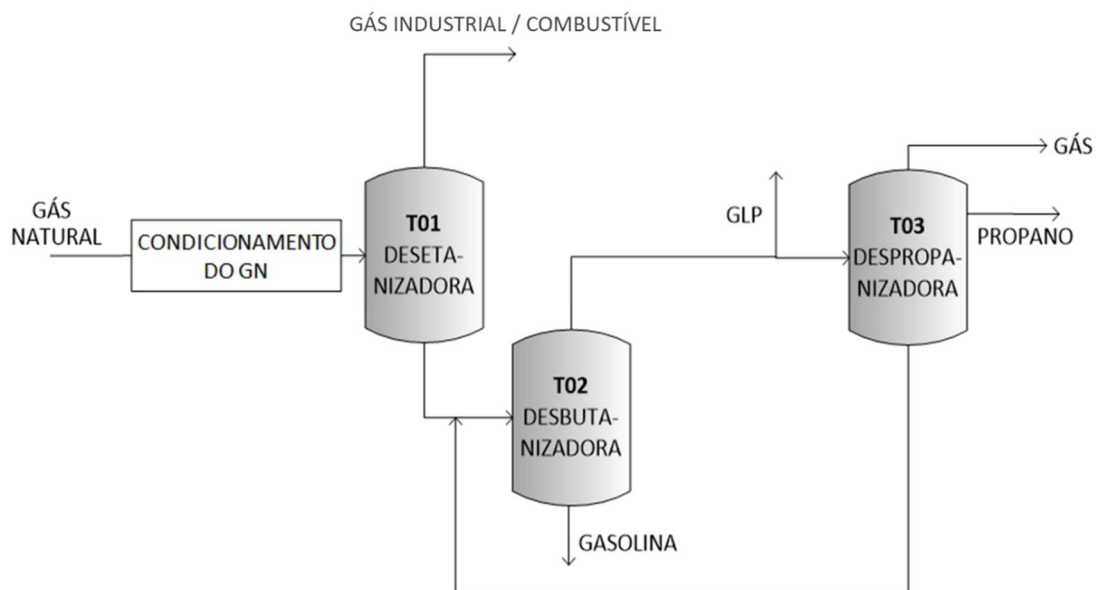


Figura 5.5: Fluxograma simplificado da UPGN (BARALDI, 2015)

A corrente de fundo da desetanizadora (T01) é enviada para a coluna desbutanizadora (T02) na qual se obtém o GLP na corrente de topo e na corrente de fundo obtém-se gasolina. O GLP resultante é enviado para a área de estocagem de GLP.

Uma parte da corrente de topo da desbutanizadora é alinhada para a coluna de despropanização (T03) para a produção de propano. O propano retirado é utilizado no ciclo de refrigeração. A carga térmica utilizada nas colunas desetanizadora, despropanizadora e desbutanizadora é fornecida pelo reboiler através da troca térmica com o óleo de aquecimento que passa por seus tubos.

5.3 Unidade de Refrigeração

A Unidade de Refrigeração se faz necessária para reduzir a temperatura no gás natural nos trocadores de calor P02 e P04 que fazem parte da etapa de condicionamento do gás. O fluido refrigerante utilizado no ciclo de refrigeração é o próprio propano removido na coluna T03.

A Figura 5.6 apresenta o fluxograma da Unidade de Refrigeração. A corrente em vermelho na figura representa o gás natural. As demais correntes do fluxograma são de propano. Na UPGN estudada o compressor utilizado apresenta dois estágios. Na figura estes estágios são representados por dois compressores.

- Vaso de separação S03: neste vaso ocorre a separação da corrente de topo para o primeiro estágio do compressor e a corrente líquida de fundo é retirada do ciclo de refrigeração;
- Primeiro estágio do compressor C01: realiza a compressão do produto de topo do vaso S03;
- Segundo estágio do compressor C02: realiza a compressão do produto de topo do vaso S02 e da corrente de saída do primeiro estágio do compressor (C01).

Através da Figura 5.6 é possível se observar que existem duas correntes de saída de propano do ciclo. Uma se encontra no topo do vaso de armazenamento S01 e a outra se encontra no fundo do vaso S03. A corrente no topo do vaso S02 é utilizada para o controle de pressão do vaso. Já no vaso S03 se realiza o alinhamento do propano para fora da Unidade de Refrigeração.

Esta unidade quando opera em estado estacionário, a vazão de propano oriunda da coluna T03 equivale a vazão removida através do fundo do vaso S03. Neste estudo se desprezou a vazão para controle de pressão no topo do vaso S02, pois para a simulação em estado estacionário não se realiza este controle sendo esta vazão nula.

6 Resultados e Discussão

Neste capítulo será aplicada a metodologia proposta no estudo de caso apresentado no capítulo anterior, sendo desta forma organizado em: a) construção do modelo da planta UPGN em Aspen Plus 8.8, b) a definição da função objetivo de otimização e suas variáveis de decisão, e por fim, c) os resultados obtidos através do algoritmo de otimização.

As otimizações realizadas se dividiram em dois casos, um sendo com o objetivo de minimizar o consumo de energia e o outro visando minimizar as perdas de exergia na planta. Para cada um desses casos utilizou-se duas composições de gás bruto, sendo uma chamada de composição original e a outra de composição rica em pesados. No total tem-se quatro otimizações que possibilitam a comparação entre a otimização energética e a de perda de exergia para a planta estudada.

6.1 Modelagem em Aspen Plus 8.8

O Aspen Plus é um software comercial que realiza modelagem de processos industriais. Neste estudo se realizou uma modelagem inicial de uma UPGN existente e, a partir desta modelagem foram alteradas condições de processo específicas para se adequar a modelagem com a função objetivo da otimização. A modelagem inicial serviu como referência para otimização realizada neste estudo, sendo os resultados obtidos comparados frente a mesma.

A Figura 6.1 apresenta o fluxograma de processo da modelagem construída no Aspen Plus para simular a UPGN. Nesta figura estão destacados os principais equipamentos e produtos.

Ao configurar o fluxograma no Aspen, houve a preocupação de que se obedecesse às especificações de qualidade dos produtos formados na saída de cada coluna de destilação do processo. Portanto, foi especificado a composição dos produtos formados na coluna e para se atender estas especificações variou-se a razão de *boilup* das colunas. Também, se manteve as vazões de produtos formados, pois assim, se comparou o consumo de energia e a perda de exergia da unidade em relação a uma vazão de produtos constante.

As otimizações foram realizadas para duas condições de alimentação da UPGN, sendo uma correspondente a composição original da UPGN e a outra uma composição mais rica nos componentes mais pesados do gás bruto. A Tabela 6.1 exhibe a composição da corrente de gás bruto da UPGN. O pacote termodinâmico utilizado na simulação foi o Peng-Robinson.

Primeiramente, simulou-se a UPGN para a condição de composição original, e para esta condição efetuou-se a otimização energética da unidade e, posteriormente, para esta mesma condição de alimentação, se realizou a otimização com base na perda de exergia da unidade. Por fim, se comparou os resultados obtidos das otimizações e a variação no perfil de perda de exergia da coluna desetanizadora. Este mesmo procedimento foi abordado para a condição de composição rica em pesados para a corrente de gás bruto.

A seguir serão detalhadas as configurações operacionais inseridas no Aspen Plus para os principais equipamentos da UPGN.

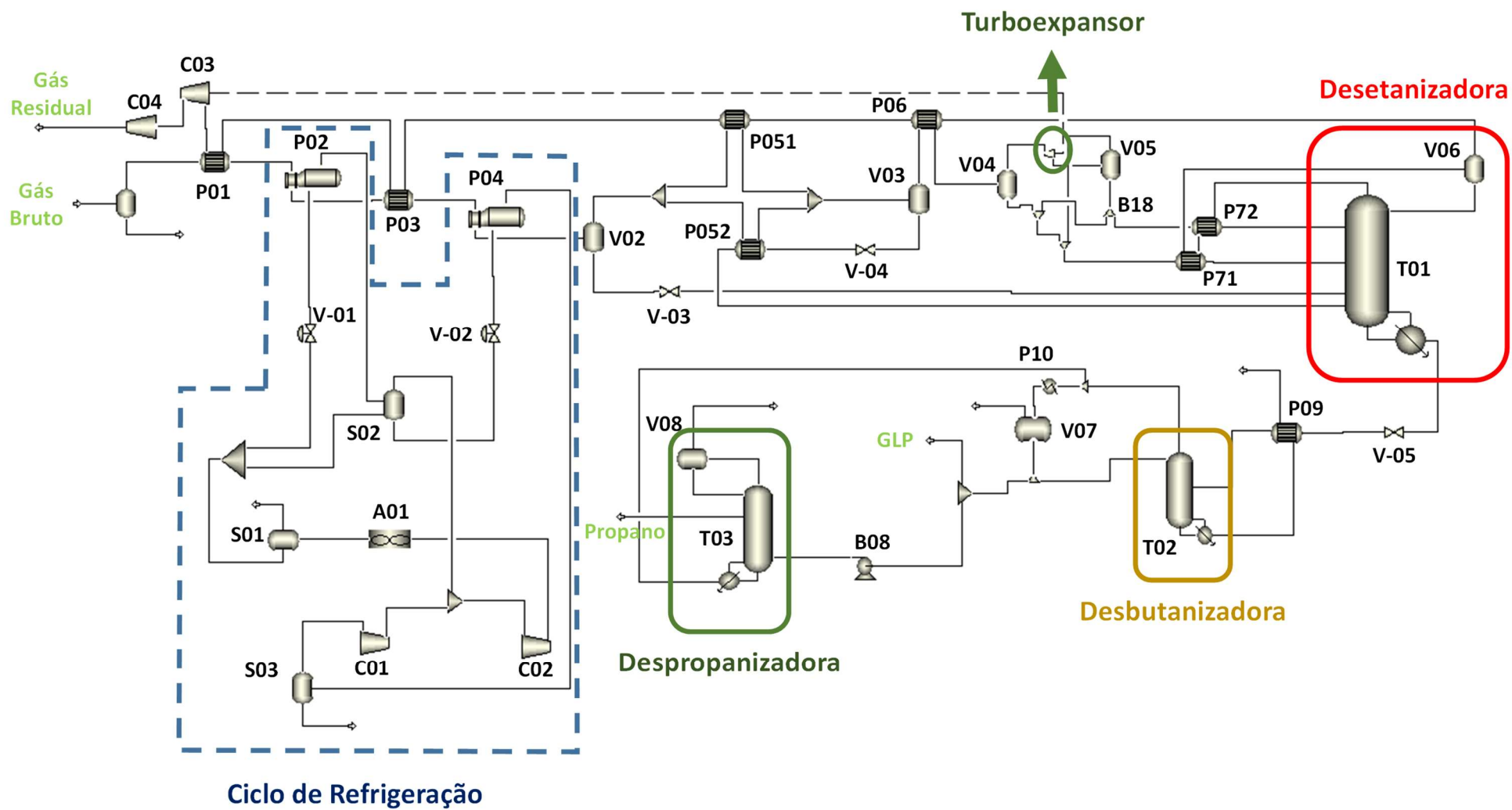


Figura 6.1: Diagrama de processo construído no Aspen Plus para a UPGN

Tabela 6.1: Composições consideradas para o gás bruto

	Composição Original		Composição Rica	
	Vazão Molar (kmol/h)	Fração Molar	Vazão Molar (kmol/h)	Fração Molar
Nitrogênio	64,45	0,0186	64,45	0,018599
CO ₂	37,77	0,0109	37,77	0,0109
Metano	2652,11	0,7654	2390	0,689717
Etano	382,19	0,1103	500	0,144292
Propano	175,33	0,0506	250	0,072146
Isobutano	37,42	0,010799	70	0,020201
n-Butano	55,44	0,016	60	0,017315
2-Metil-Butano	14,55	0,004199	30	0,008658
n-Pentano	21,14	0,006101	30	0,008658
n-Hexano	16,63	0,004799	25	0,007215
n-Heptano	6,58	0,001899	6,58	0,001899
n-Octano	1,04	0,0003	1,04	0,0003
n-Nonano	0,35	0,000101	0,35	0,000101
Total	3465		3465,19	

6.1.1 Ciclo de Refrigeração

O ciclo de refrigeração está representado detalhadamente na Figura 6.5. O objetivo deste ciclo é resfriar o gás bruto fazendo a temperatura da corrente 48 atingir 17 °C. Para isso, a vazão de propano no ciclo é manipulada para atingir o *setpoint* de temperatura desejado.

Os trocadores de calor do ciclo de refrigeração são do tipo *kettle*. Desta forma, se considerou na simulação que esses trocadores variam sua área de troca térmica de acordo com o nível de propano dentro do trocador.

Para a modelagem deste ciclo no Aspen Plus se necessitou da interface com o Python para o cálculo da vazão de propano no ciclo. Pela lógica de modelagem utilizada no Aspen, cada corrente ou bloco (equipamento ou instrumento), que possua alguma propriedade definida pelo usuário, deve ao final das especificações feitas pelo usuário ter grau de liberdade igual à zero. Este modo de modelagem se chama *Sequential Modular (SM)*.

No modo SM de modelagem, caso se necessite obter a temperatura da corrente 2 da Figura 6.2 é necessário ter-se grau de liberdade zero na corrente 1 e no trocador de calor. Como resultado todas as variáveis da corrente 2 serão calculadas após a simulação.

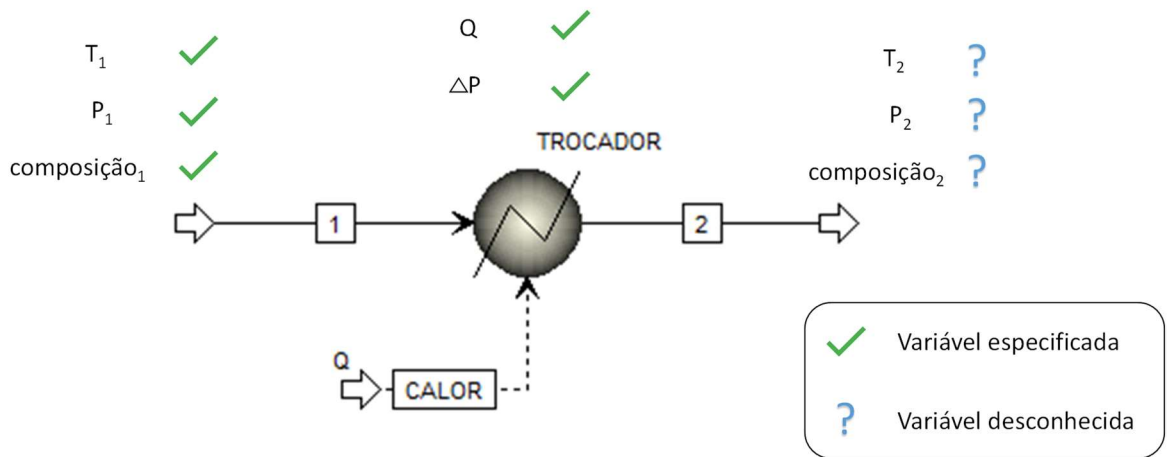


Figura 6.2: Especificação de variáveis para simulação no modo SM

Neste tipo de modelagem não é possível simular a condição apresentada na Figura 6.3, onde, apesar de se possuir grau de liberdade zero global para realizar a simulação, os blocos correspondentes ao trocador de calor e a corrente 2 não estão com grau de liberdade zero. Ou seja, na modelagem SM os blocos não podem ser parcialmente preenchidos.

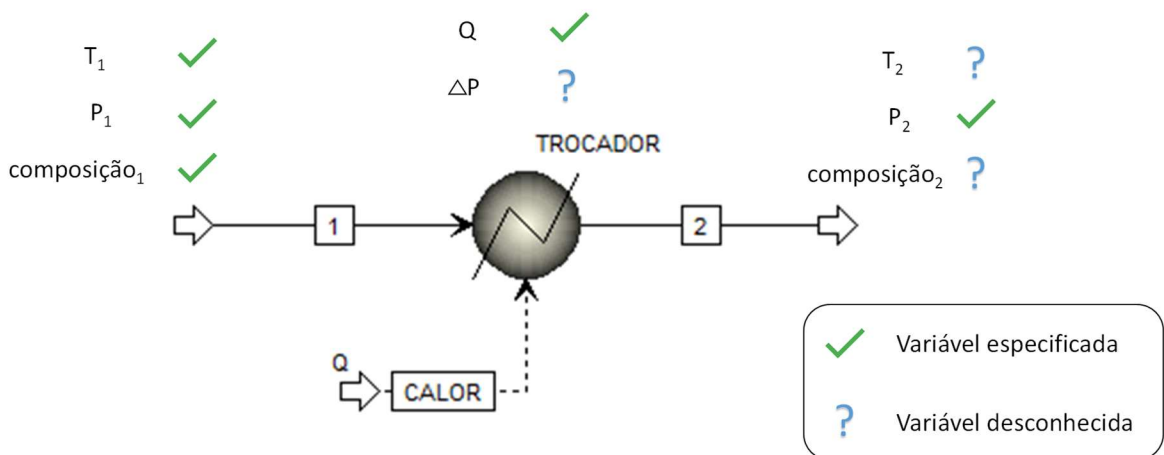


Figura 6.3: Situação não calculada no modo SM

Para resolver problemas conforme apresentado na Figura 6.3 o modo de cálculo da modelagem deve ser alterado no Aspen Plus de *Sequential Modular (SM)* para *Equation Oriented (EO)*.

Na modelagem desenvolvida nesta dissertação para o ciclo de refrigeração tem-se correntes com especificações parciais, conforme ilustra a Figura 6.4. Nesta figura, a vazão total de propano no ciclo é definida na corrente 3P. Esta vazão está em função da vazão necessária nos trocadores P02 e P04. Para executar a simulação no modo SM deve-se especificar a vazão na corrente 3P, ou seja, esta deve ser previamente conhecida. A solução presente no Aspen Plus para solucionar este tipo de problema seria alterar o modo de SM para EO. Porém, neste estudo, não se obteve êxito ao se

efetuar esta modificação. A solução encontrada foi elaborar através do Python um polinômio que calcula a vazão total de propano necessária no ciclo em função das pressões a jusante das válvulas V01, V02 e em função da temperatura da corrente de GN na saída do P04 (corrente 48 da Figura 6.4).

Ao se executar a otimização no Python, os valores assumidos para as variáveis de decisão são definidos pelo algoritmo genético. Entre essas variáveis estão a pressão no trocador P02, P04 e temperatura da corrente 48. Com os valores destas três variáveis, o Python calcula, através de um polinômio previamente obtido, qual a vazão total de propano necessário no ciclo e a distribuição desta vazão no Split B7 e insere esta informação no bloco da corrente 3P no Aspen Plus. Assim, o grau de liberdade no bloco da corrente e no bloco do split se iguala a zero tornando possível a simulação no modo SM.

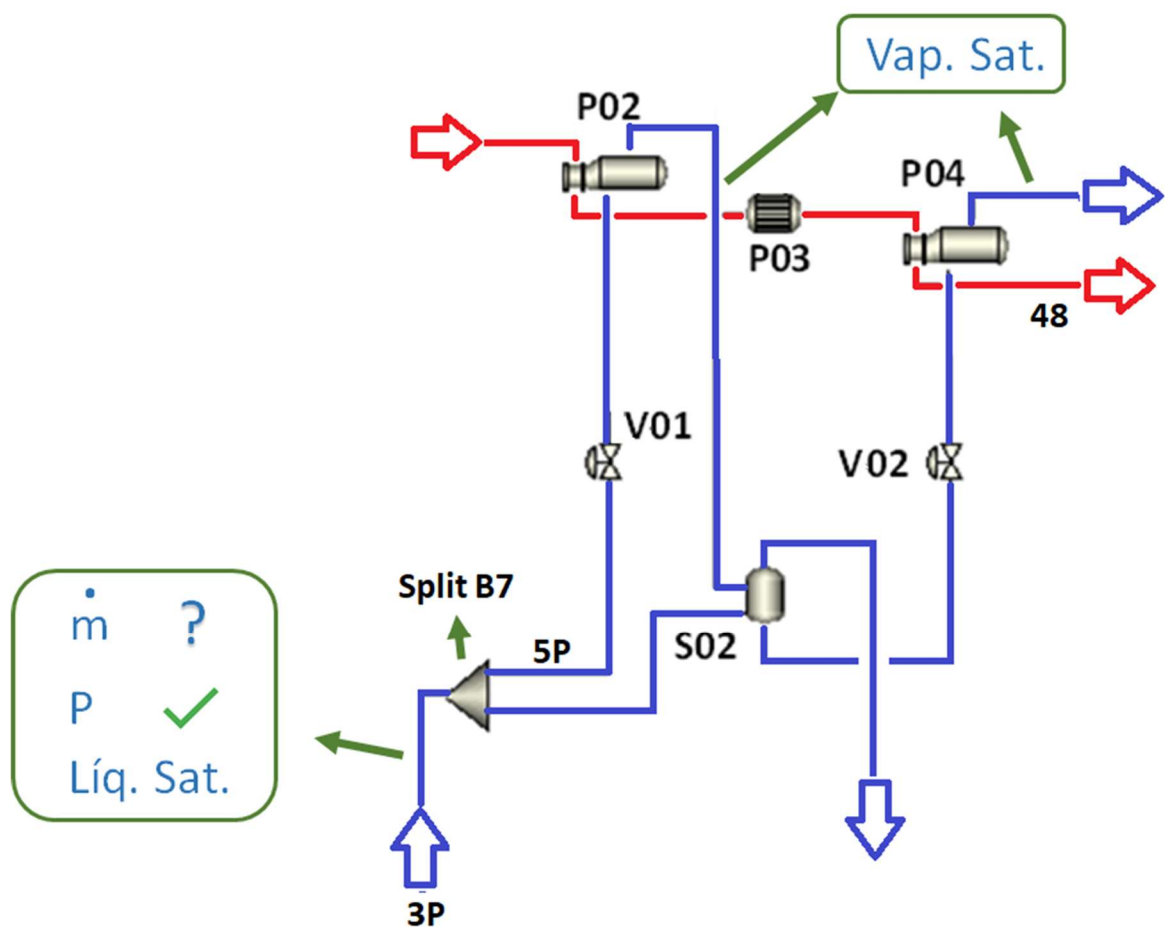


Figura 6.4: Representação parcial do ciclo de refrigeração

Para se obter o polinômio que calcula a vazão total de propano no ciclo e a vazão no Split B7 em função das pressões nos trocadores P02 e P04 e da temperatura da corrente 48 foram elaboradas tabelas com os pontos operacionais simulados. Primeiramente, elaborou-se uma tabela para se obter o polinômio que descreve a vazão de propano para o trocador P02 (corrente 5P) em função da pressão a jusante da válvula V01. Como restrição se estabeleceu para este trocador que a temperatura

do GN na saída fosse igual a 13 °C, valor original desta corrente, e que o propano na saída deste trocador estivesse próximo a condição de vapor saturado (foi considerado 98 % de vapor nesta corrente de propano para se evitar troca de calor sensível). Assim, se executou manualmente diversas simulações em Aspen Plus para se obter o valor da vazão da corrente 5P em função da pressão a jusante da válvula V01 que atendesse as restrições mencionadas. Com os valores desta tabela se fez um ajuste polinomial em Python, dado pela Equação (6.1), a qual descreve a vazão em *kg/h* na corrente 5P, m_{5P} , em função da pressão em *bar* a jusante de V01, P_{V-01} .

$$\dot{m}_{5P} = 0,2657 \cdot P_{V-01}^3 + 101,97 \cdot P_{V-01}^2 - 2100,55 \cdot P_{V-01} + 33029,22 \quad (6.1)$$

Procedimento semelhante foi realizado para se obter o polinômio que calcula a vazão de propano no ciclo, m_{3P} em *kg/h*, em função da pressão da válvula V02, P_{V-02} , em *bar*. Sendo que para este caso foi aplicado a restrição da temperatura da corrente 48 igual a - 17 °C e que o propano na saída do trocador P-04 estivesse no equilíbrio líquido-vapor com 98 % de vapor. A Equação (6.2) descreve o polinômio obtido.

$$\dot{m}_{3P} = -55,86 \cdot P_{V-02}^5 + 531,22 \cdot P_{V-02}^4 - 2235,06 \cdot P_{V-02}^3 + 5555,4 \cdot P_{V-02}^2 - 9554,8 \cdot P_{V-02} + 61609,9 \quad (6.2)$$

Um terceiro polinômio foi obtido para se conhecer a influência da temperatura da corrente 48, T_{48} em °C, na vazão total de propano no ciclo, m_{3P} em *kg/h*. Para se obter este polinômio se considerou constante a vazão no trocador P-02 e também as pressões a jusante das válvulas V01 e V02. Assim, variou-se manualmente no simulador Aspen Plus a vazão de 3P para se atingir diferentes temperaturas para a corrente 48, sendo mantido a fração de vapor da corrente de propano no P-04 em 98 % de vapor. O resultado deste procedimento gerou a Equação (6.3).

$$\dot{m}_{3P} = -0,347 \cdot T_{48}^3 - 9,97 \cdot T_{48}^2 - 1217,51 \cdot T_{48} - 19431,70 \quad (6.3)$$

Conhecendo-se esses polinômios é possível definir a vazão de propano total no ciclo, assim como, definir a vazão da corrente 5P. Com essas informações os blocos do Aspen Plus correspondentes à corrente 3P e ao Split B7 possuirão grau de liberdade igual a zero tornando possível a simulação no modo SM.

As Equações (6.1), (6.2) e (6.3) foram obtidas considerando que a composição do gás natural é a composição original da Tabela 6.1. Para a condição de composição rica apresentada nesta mesma tabela, foram obtidas as seguintes equações:

$$\dot{m}_{5P} = -81,42 \cdot P_{V-01}^3 + 1435,70 \cdot P_{V-01}^2 - 9943,34 \cdot P_{V-01} + 65626,13 \quad (6.4)$$

$$\dot{m}_{3P} = -574,38 \cdot P_{V-02}^3 + 3782,22 \cdot P_{V-02}^2 - 10556,27 \cdot P_{V-02} + 88340,81 \quad (6.5)$$

$$\dot{m}_{3P} = 0,347 \cdot T_{48}^3 + 26,04 \cdot T_{48}^2 - 886,06 \cdot T_{48} - 20916,52 \quad (6.6)$$

Durante a otimização se variou a pressão do propano na descarga da válvula V02 de 1 a 2,5 bar. O valor mínimo de 1 bar foi estabelecido para que não ocorresse operação em condição de vácuo, já o valor máximo de 2,5 bar foi determinado de acordo com o *approach* de temperatura mínimo necessário no trocador P04. Uma pressão superior a 2,5 bar à jusante da válvula V02 torna a temperatura do propano na corrente 13P muito próxima à temperatura da corrente 48 (17°C) inviabilizando a troca térmica no trocador. Já para a válvula V01 do ciclo de refrigeração, a variação de pressão à jusante da válvula foi de 3,5 a 6,5 bar.

As pressões de operação dos equipamentos S02, S03, C01 devem ser ajustadas de acordo com as pressões à jusante das válvulas V01 e V02.

A vazão de propano no ciclo varia de 52800 a 57700 kg/h para a condição do gás bruto com a composição original e de 76650 a 83900 para a condição de composição rica.

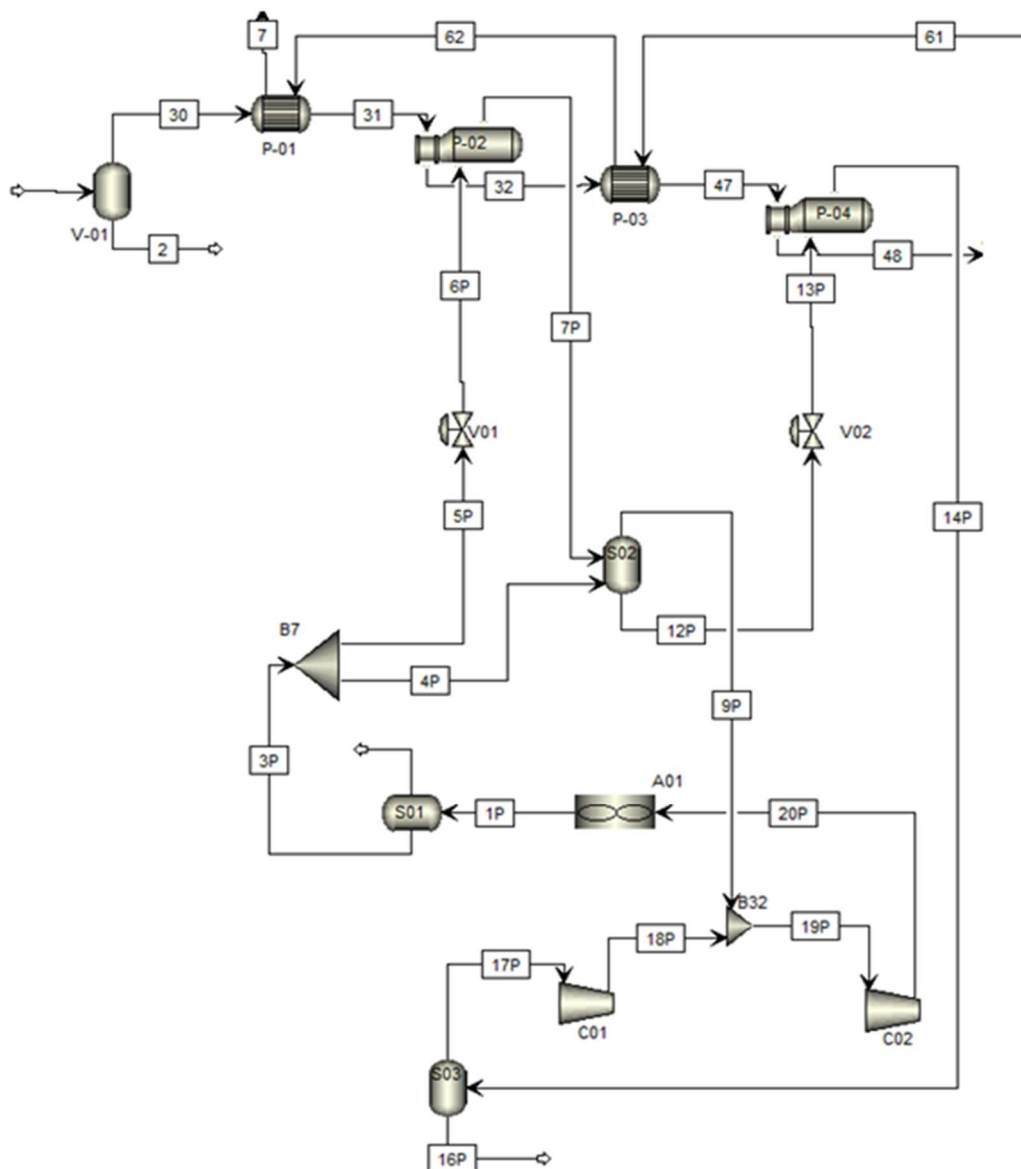


Figura 6.5: Ciclo de refrigeração modelado

6.1.2 Colunas de Destilação

As Tabela 6.2, Tabela 6.3 e Tabela 6.4, apresentam as condições operacionais das colunas T01, T02 e T03, respectivamente, tanto para a condição de composição original do gás bruto quanto para a composição rica.

Tabela 6.2: Dados operacionais da coluna T01

	Composição Original	Composição Rica
Pressão de Operação (bar)	25,82	25,82
Nº de estágios	45	45
Estágios de alimentação		
Corrente 76	8	8
Corrente 74	25	25
Corrente 58	31	31
Corrente 46	34	34
Vazões (kg/h)		
Corrente 76	1912,9	1572,8
Corrente 74	52183,9	42105
Corrente 58	1594,6	1851,6
Corrente 46	19775,8	37202
Corrente 21 (topo)	57520,3	53906,7
Corrente 16 (fundo)	17947,9	28823,4
Temperatura (°C)		
Corrente 76	-61,6	-58,3
Corrente 74	-66,3	-63,5
Corrente 58	-19,5	-21,2
Corrente 46	-36,0	-37,1
Corrente 21	-65,8	-62,9
Corrente 16	98,8	74,1

Tabela 6.3: Dados operacionais da coluna T02

	Composição Original	Composição Rica
Pressão de Operação (bar)	14,8	14,8
Nº de estágios	29	29
Estágios de alimentação		
Corrente 13	11	11
Corrente 42	1	1
Vazões (kg/h)		
Corrente 13	17947,9	28823,4
Corrente 42	12028,3	20847,6
Corrente 4 (topo)	24586,7	42469,9
Corrente 14 (fundo)	5389,4	7201,1
Temperatura (°C)		
Corrente 13	79,6	60,0
Corrente 42	54,3	31,4
Corrente 14	148,5	157,6
Corrente 4	64,5	55,7

Tabela 6.4: Dados operacionais da coluna T03

	Composição Original	Composição Rica
Pressão de Operação (bar)	21,8	21,8
Nº de estágios	30	30
Estágios		
Corrente 38	30	30
Corrente Propano	10	10
Vazões (kg/h)		
Corrente 38	2475,8	4291,1
Corrente Propano	505,9	496,0
Corrente 50 (topo)	1497,8	2832,6
Corrente 11 (fundo)	472,1	962,5
Temperatura (°C)		
Corrente 38	57,2	33,6
Corrente Propano	81,8	73,0
Corrente 50 (topo)	77,6	64,9
Corrente 11 (fundo)	81,9	73,1

6.1.3 Trocadores de calor

Não foi considerado perda de carga nos trocadores de calor da UPGN. As especificações de área de troca térmica utilizadas no modelo em Aspen estão descritas na Tabela 6.5.

Tabela 6.5: Área dos trocadores de calor

Trocador	Área (m ²)
P01	107
P03	98,5
P051	176
P052	6
P06	105
P71	85
P72	7,5
P09	8,2

Os trocadores de calor tipo *kettle*, P02 e P04, tiveram especificados a temperatura de saída da corrente fria em 13 e -17 °C, respectivamente. Já os trocadores P10 e A01 são condensadores e foram especificados fração de vapor nula nestes dois equipamentos.

Os vasos V06 e V08 realizam a função de condensadores parciais, sendo especificados nestes as frações de vapor de 0,9 e 0,67 respectivamente.

6.1.4 Compressores e turboexpansor

Estes equipamentos foram todos especificados como isentrópicos no simulador, sendo mantida a eficiência padrão de 0,72 do software. As pressões de descargas descritas na Tabela 6.6. O compressor C03 está ligado ao eixo do turboexpansor TE01. Logo, o trabalho gerado no TE01 é transferido para o compressor C03, assim, não se especifica pressão na descarga deste.

Tabela 6.6: Pressões de descarga nos compressores e no turboexpansor

Equipamento	Pressão (bar)
C03	---
C04	69
TE01	26,4
C01	6,0
C02	21,8

6.1.5 Vasos de separação

Os vasos de separação tiveram especificados as suas pressões de operação, de acordo com a Tabela 6.7, e as cargas térmica foram consideradas nula em todos os vasos.

Tabela 6.7: Pressão de operação dos vasos de separação

Vasos	Pressão (bar)
V02	68
V03	68
V04	67,7
V05	27
V06	25,9
V07	14,8
V08	21,8
S01	21,8
S02	6
S03	1,5

6.2 Função Objetivo das Otimizações

A função objetivo utilizada para minimizar a perda de exergia da planta está descrita na Equação (6.7). Nesta são somadas as perdas de exergia individuais de cada equipamento/instrumento da UPGN. Os cálculos para perda de exergia dos equipamentos são efetuados conforme a Tabela 2.1. Neste estudo, estes cálculos de perdas de exergias foram realizados através da interface Python e Aspen Plus, onde o primeiro realizou os cálculos necessários e o segundo informou as propriedades físico-químicas das correntes. Para as colunas do modelo, foi necessário apenas somar as perdas de exergia em cada estágio da coluna em questão, visto que o software Aspen Plus fornece a perda de exergia de cada estágio nos resultados da simulação.

$$f_{min} = \sum_{Equip} Ex_{loss} \quad (6.7)$$

Já para a otimização com objetivo de minimizar o consumo de energia foram considerados apenas os equipamentos que consomem energia de utilidades da planta, conforme descrito na Equação (6.8).

$$f_{min} = |En_{C04}| + |En_{V06}| + |En_{T01}| + |En_{T02}| + |En_{P10}| + |En_{B08}| + |En_{T03}| + |En_{B3}| + |En_{A01}| + |En_{S03}| + |En_{C01}| + |En_{C02}| \quad (6.8)$$

Os subíndices das variáveis da equação estão de acordo com os tags dos equipamentos mostrados na Figura 6.1.

O método de otimização selecionado foi o algoritmo genético, conforme o método utilizado por Shin et al. (SHIN; YOON; KIM, 2015). Para se executar este método foi adicionado no Python o pacote DEAP, conforme apresentado no capítulo 4.2.

A seleção das variáveis a serem modificadas pelo algoritmo genético durante a otimização foi realizada baseando-se no conhecimento operacional da planta UPGN em questão e no estudo elaborado por Shin et al. (SHIN; YOON; KIM, 2015). Neste estudo foi conduzida uma análise de sensibilidade para se concluir as variáveis que possuem impacto mais relevantes na planta de recuperação de gás natural.

As variáveis selecionadas são (os tags dos instrumentos/equipamentos podem ser verificados na Figura 6.5 e Figura 6.6):

- Pressão do propano à jusante da válvula V01 (propano que entra no trocador P02): modificando a pressão do propano se altera seu calor latente de vaporização o que acarreta variação na vazão de propano necessária para se realizar a troca térmica desejada;
- Pressão do propano à jusante da válvula V02 (propano que entra no trocador P04);
- Temperatura da corrente 48 (corrente de gás bruto na saída do trocador P04): alterando o *set* de temperatura desta corrente, modifica-se a vazão de propano necessária para a troca térmica. Esta modificação também influencia a separação no vaso de flash V02 e no processo subsequente;
- Pressão do turboexpansor TE01: variando esta pressão se altera também a pressão de operação da coluna T01;
- Fração no Split B18 que é encaminhada para a corrente 9: altera-se a quantidade de produto destinada para a corrente 73 e a 75, sendo que estas correntes alimentam a coluna T01 em estágios distintos.

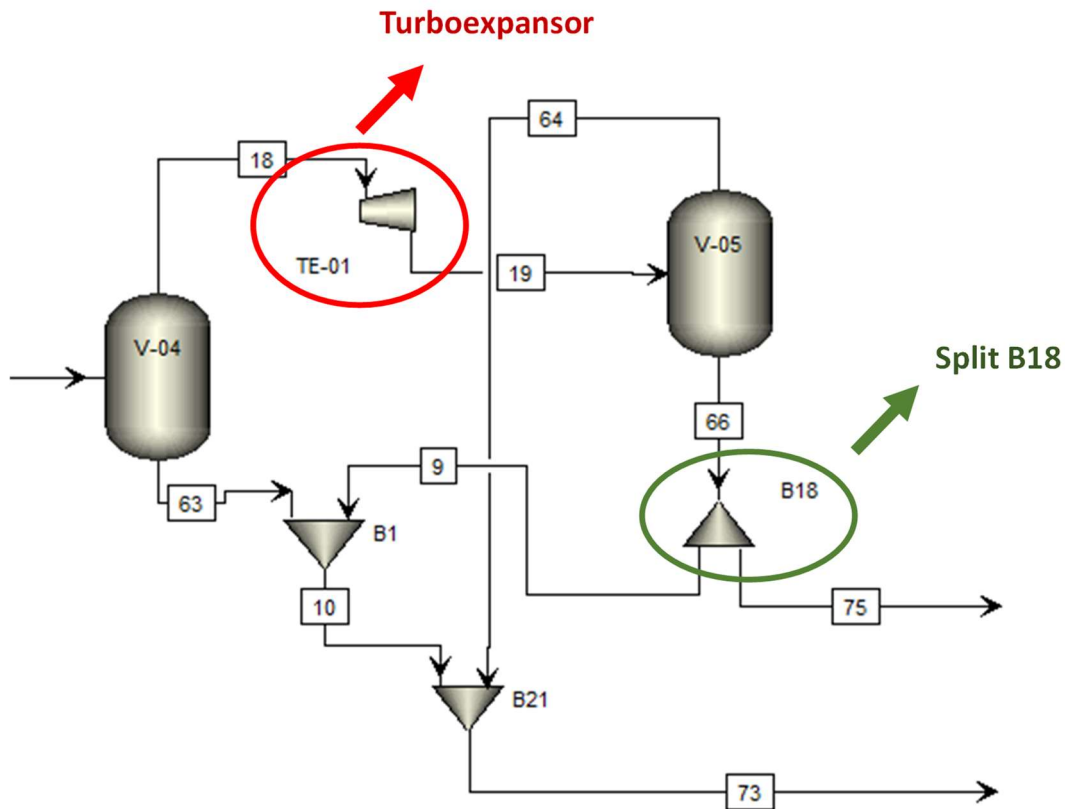


Figura 6.6: Fluxograma com o turboexpansor TE01 e o *split* B18

Deve-se atentar que a modificação na pressão do propano no ciclo acarreta modificação na pressão de outros equipamentos do ciclo, assim, como também, a alteração na pressão do turboexpansor acarreta alteração na pressão de operação da coluna T01, do vaso V05 e V06. Desta forma, a fim de se adequar as pressões em diferentes equipamentos do sistema, o código escrito no Python considera alterações nos parâmetros operacionais em equipamentos além daqueles que contém as variáveis de decisão da otimização.

Para executar o problema de otimização se consideraram as restrições nas variáveis de decisão presentes na Tabela 6.8. Também se restringiu a recuperação do propano no fundo da coluna T01. Esta recuperação deve ser superior a 99 % e para isso se ajusta a razão de *boilup* da coluna.

Tabela 6.8: Restrições nas variáveis de decisão

Variáveis	Limite mínimo	Limite máximo
Pressão propano no P02 (bar)	1,0	2,5
Pressão propano no P04 (bar)	3,5	6,5
Temperatura corrente 48 (°C)	-11	-23
Pressão no Turboexpansor (bar)	23	33
Split para corrente 09	0,02	0,98

No algoritmo genético criou-se o indivíduo com as cinco variáveis de decisão listadas na Tabela 6.8. A população do algoritmo é composta por 10 indivíduos. A formação da primeira população ocorre de forma randômica. Para a evolução do algoritmo considera-se o cruzamento entre os indivíduos da população. O método de cruzamento utilizado no pacote DEAP chama-se de *cxTwoPoint*. Este método realiza a permutação das variáveis entre dois indivíduos, mantendo a posição desta variável e o comprimento do indivíduo. A seleção de novos indivíduos ocorre através de torneios entre os indivíduos que compõem a população de uma geração específica. O resultado do torneio é a seleção do indivíduo mais apto, sendo a aptidão verificada observando qual indivíduo gera melhor resposta na função objetivo do problema de otimização. Com este resultado é criada uma geração. Após executar diversas otimizações observou-se que, para este estudo, o uso de 30 gerações é o suficiente para solucionar o problema, pois nas últimas gerações não tem variações significativas nos indivíduos ótimos obtidos.

Adicionou-se na modelagem trocadores de calor fictícios nas correntes de produtos da UPGN a fim de manter inalterado o estado termodinâmico destas correntes quando comparada a condição inicial de operação com a condição ótima. Por exemplo, caso a corrente de gás residual da Figura 6.1 eleve a sua temperatura após a otimização, o trocador fictício igualará esta temperatura com a desta mesma corrente no caso original simulado. Sendo assim, a função objetivo tanto para encontrar o mínimo de perda de exergia quanto a para encontrar o mínimo energético consideram a energia necessária a se adicionar ou remover para que os produtos mantenham as temperaturas constantes.

Salienta-se que os resultados gerados para o perfil de perda de exergia de colunas contemplam apenas a coluna T01. Isto porque, a seleção das variáveis de decisão do problema de otimização e as restrições aplicadas na coluna T01, tornaram irrelevantes alterações no perfil de perdas de exergia das demais colunas.

6.3 Otimização Energética

Dentro deste capítulo são apresentados os resultados para a otimização energética. Primeiramente, será visto a otimização para a composição original da UPGN e posteriormente para a composição rica.

6.3.1 Composição Original

A composição original está apresentada na Tabela 6.1. Para a otimização apresentada neste capítulo utilizou-se como condição inicial os dados operacionais usuais da planta estudada. A condição final de operação corresponde aos dados obtidos após a otimização energética.

A Tabela 6.9 mostra os valores das variáveis de decisão na condição original da planta e na condição final obtida após otimização energética.

Tabela 6.9: Valores das variáveis de decisão

Variáveis de decisão	Condição Original	Condição Final (Otimização Energética)
Pressão propano no P02 (bar)	6,0	5,71
Pressão propano no P04 (bar)	1,5	2,09
Temperatura corrente 48 (°C)	-17	-12,54
Pressão no Turboexpansor (bar)	26,4	26,0
Split para corrente 09	0,8	0,70

A Figura 6.7 apresenta os dados de consumo de energia para a planta na sua condição original de operação e para sua condição ótima. Nestas condições, houve uma diminuição no consumo total de energia de 17356,2 para 16518,5 kW, o que representa uma economia 4,8%.

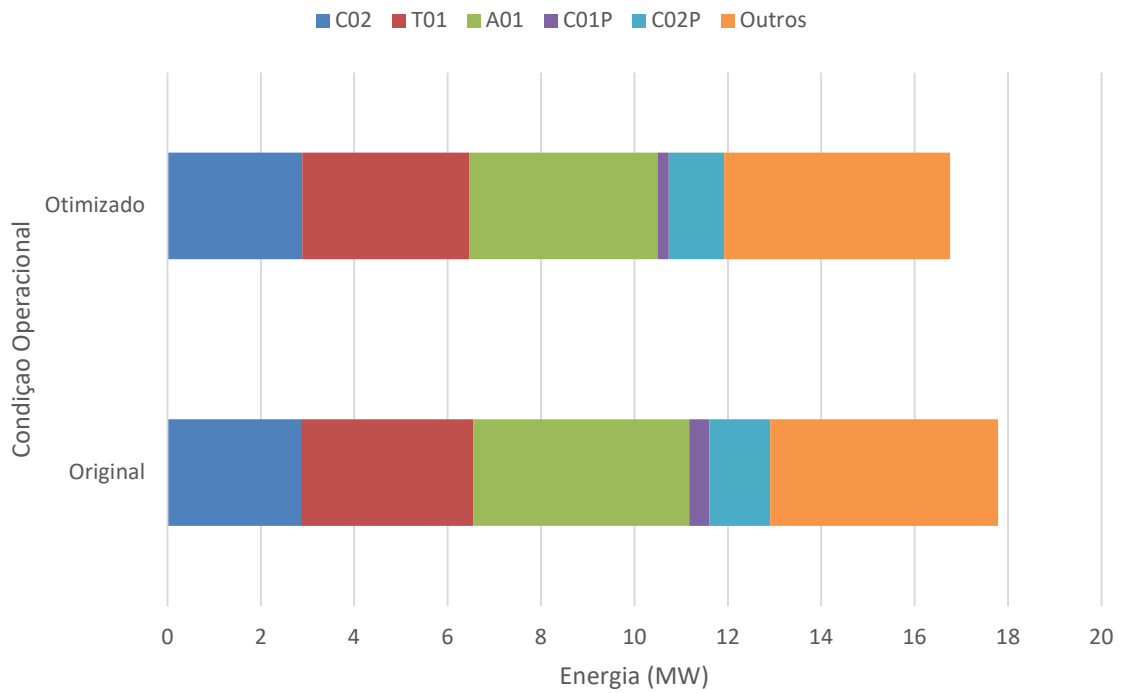


Figura 6.7: Consumo de energia da planta na condição original e ótima energética

A Figura 6.8 apresenta os dados de perdas de exergia para os equipamentos da UPGN em sua condição original de operação e na sua condição ótima energética obtida. Por esta figura, nota-se que houve uma diminuição na perda total de exergia de 4729,1 para 4477,8 kW, o que representa uma redução de 5,31 %.

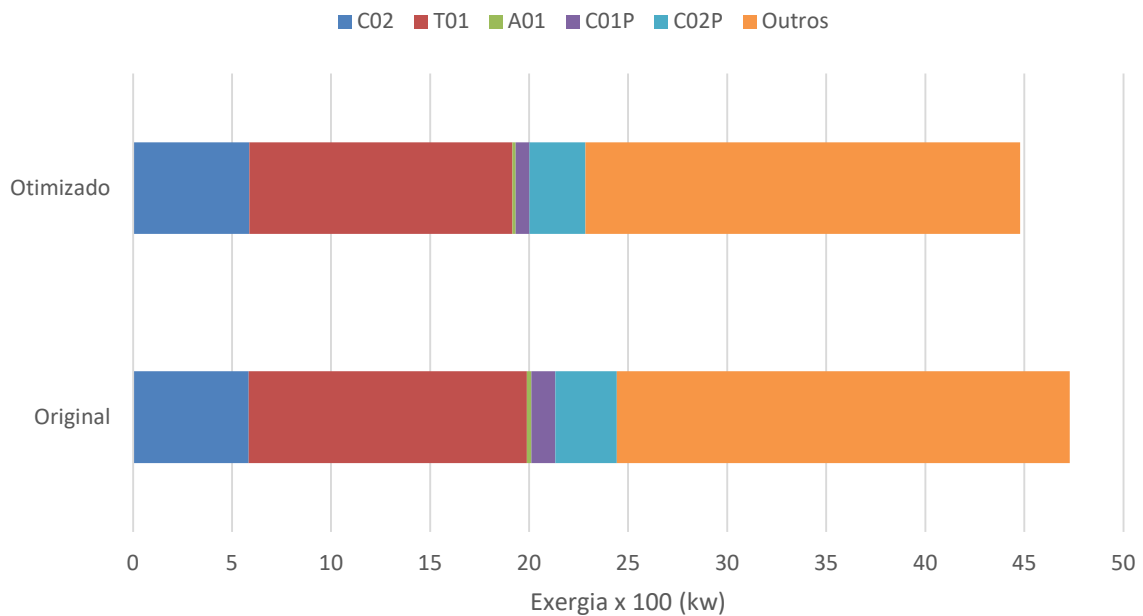


Figura 6.8: Perda de exergia na planta na condição original e ótima energética

A Figura 6.9 exibe o perfil de perda de exergia para a coluna T01 na (a) condição original de operação e na (b) condição ótima energética obtida. Por estes perfis, observa-se que a coluna possui perda de exergia mais elevada na região de esgotamento do que a de retificação. Conforme mencionado no capítulo 3.1.2, uma coluna com uma distribuição equilibrada no perfil de perda de exergia opera de forma mais eficiente termodinamicamente. Observa-se por esta figura que a coluna em sua condição ótima possui uma distribuição mais equilibrada na perda de exergia. A perda de exergia na coluna diminui de 1404,4 para 1328,7 kW, conforme apresentado na Figura 6.8. Portanto, as modificações realizadas a fim de se obter um mínimo consumo de energia acarretaram em um perfil de exergia mais equilibrado na coluna.

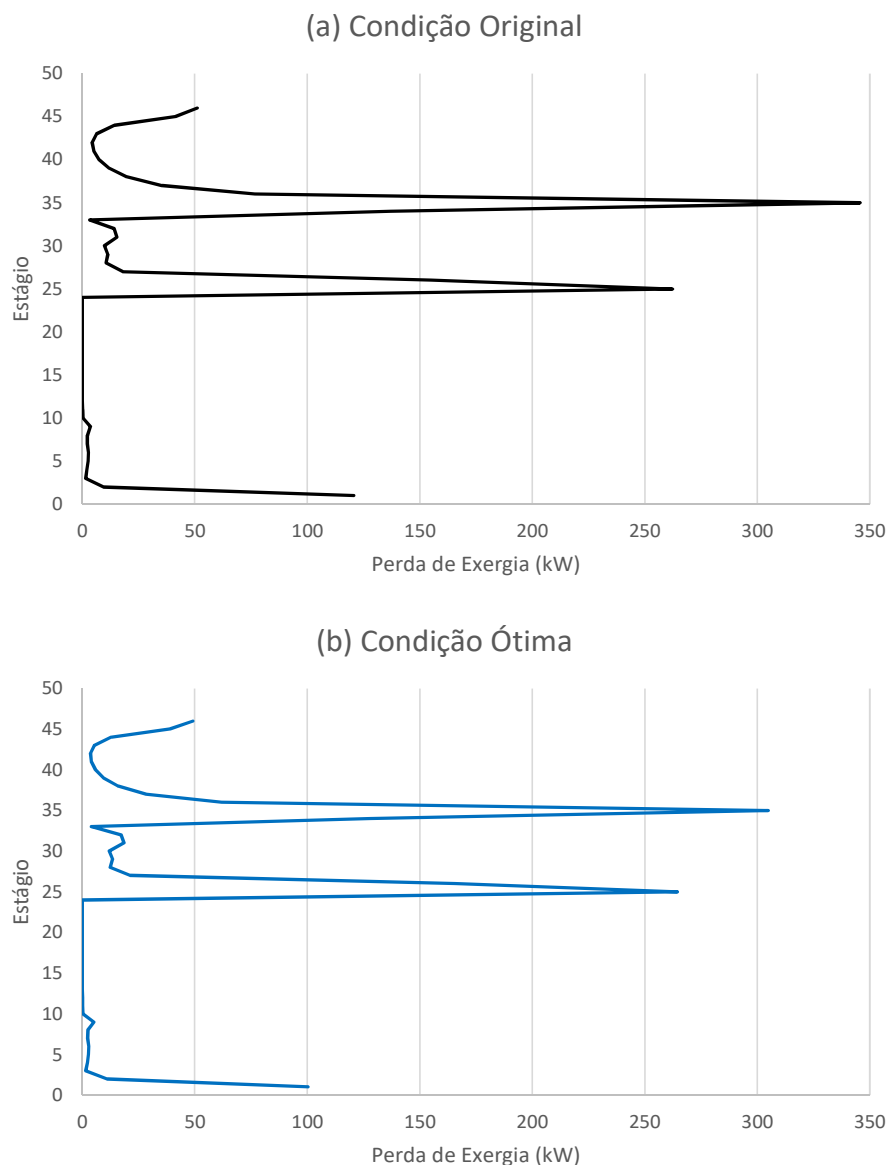


Figura 6.9: Perfil de perda de exergia para a coluna T01

O perfil de perda de exergia mais equilibrado acarretou a diminuição do consumo de energia no refeedor da coluna T01 de 3694,8 para 3569,7 kW.

A Tabela 6.10 apresenta a condição das correntes de entrada da coluna T01 para a condição original operacional e para a condição ótima energética obtida para a planta.

Tabela 6.10: Condição operacional da coluna T01

Corrente	76		74		58		46	
Estágio	9		26		32		35	
Condição	Original	Ótima	Original	Ótima	Original	Ótima	Original	Ótima
Temp. (°C)	-61,64	-63,57	-66,34	-65,69	-19,52	-15,52	-36,03	-30,89
Vazão (kmol/h)	74,33	106,12	2759,15	2807,43	52,60	56,25	578,92	495,21

Pelos dados apresentados para esta otimização percebe-se a relevante influência da elevação da temperatura da corrente 48 em diversos pontos:

- Diminuição da vazão da corrente 46: devido ao aumento da temperatura da corrente 48 (gás bruto após o ciclo de refrigeração) de -17 para -12,54 °C. Com isto, a vazão de líquido no fundo do vaso V02 diminuiu, assim, conseqüentemente, diminuiu-se a vazão da corrente 46 e a sua temperatura aumenta de -36,03 para -30,89 °C.
- Redução do pico de exergia observado no estágio 35 (correspondente a alimentação da corrente 46): Dois fatores influenciam esta redução no pico. A primeira é devido a diminuição de vazão da corrente 46 explicada no item anterior, visto que a exergia é uma propriedade extensiva. A segunda é devido a diminuição da diferença de temperatura existente entre a corrente 46 e a temperatura no estágio 35 (estágio à aproximadamente 45 °C). Esta elevação na temperatura tem efeito similar a alteração da condição térmica da alimentação explicada no capítulo 3.1.2.
- Menor consumo de energia no ciclo de refrigeração: a energia dos 4 equipamentos do ciclo de refrigeração (A01, S03, C01 e C02) consumiam um total de 6405,5 kW e após a otimização energética este valor reduziu para 5477,8 kW. Isto se deve a menor vazão de propano necessária no resfriamento do gás bruto.
- Menor perda de exergia no ciclo de refrigeração: a perda de exergia dos equipamentos do ciclo de refrigeração reduziu de 1038,1 para 837,1 kW.

6.3.2 Composição Rica em Pesados

A composição rica em pesados está apresentada na Tabela 6.1. Para a otimização apresentada neste capítulo utilizou-se como condição inicial os dados operacionais usuais da planta estudada. A condição final de operação corresponde aos dados obtidos após a otimização energética. A Tabela 6.11 mostra os valores das variáveis de decisão na condição original da planta e na condição final obtida após otimização.

Tabela 6.11: Valores das variáveis de decisão

Variáveis de decisão	Condição Original	Condição Final (Otimização Energética)
Pressão propano no P-02 (bar)	6,0	5,55
Pressão propano no P-04 (bar)	1,5	2,1
Temperatura corrente 48 (°C)	-17	-11,8
Pressão no Turboexpansor (bar)	26,4	26,15
Split para corrente 09	0,8	0,26

A Figura 6.10 apresenta os dados de consumo de energia para a planta na sua condição original de operação e para sua condição ótima. Por esta figura, nota-se que houve uma diminuição no consumo total de energia de 24625,39 para 22757,56 kW, o que representa uma economia de 7,6 %.

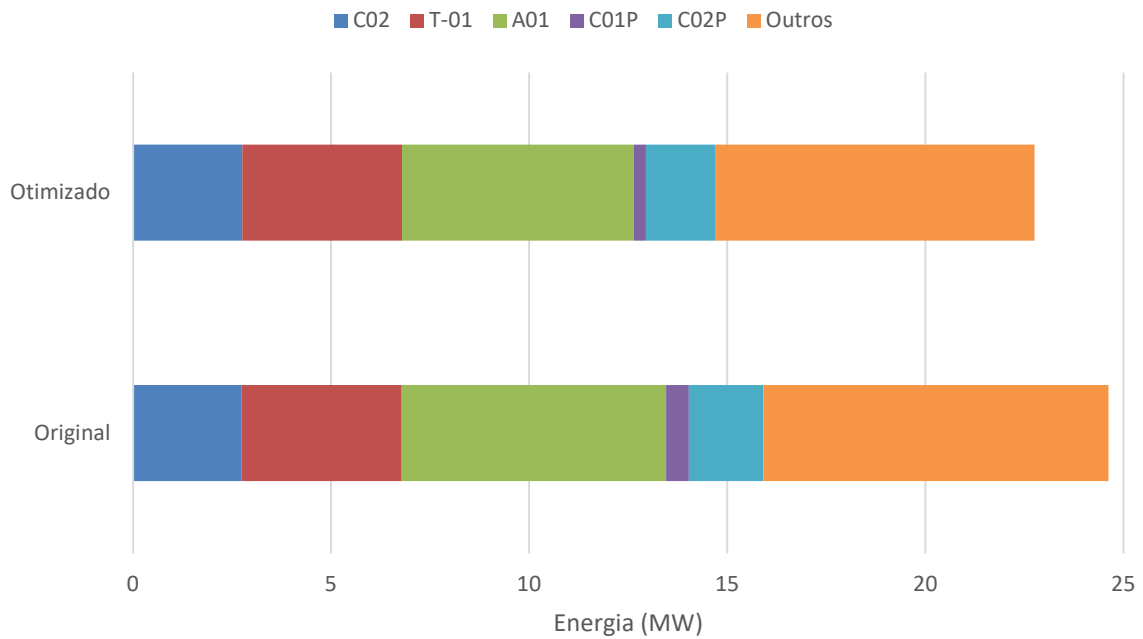


Figura 6.10: Consumo de energia da planta na condição original e ótima energética

A Figura 6.11 apresenta os dados de perdas de exergia para os equipamentos da UPGN em sua condição original de operação e sua condição ótima energética obtida. Por esta figura, nota-se que houve uma diminuição na perda total de exergia de 5697,17 para 5274,13 kW, o que representa uma redução de 7,4 %.

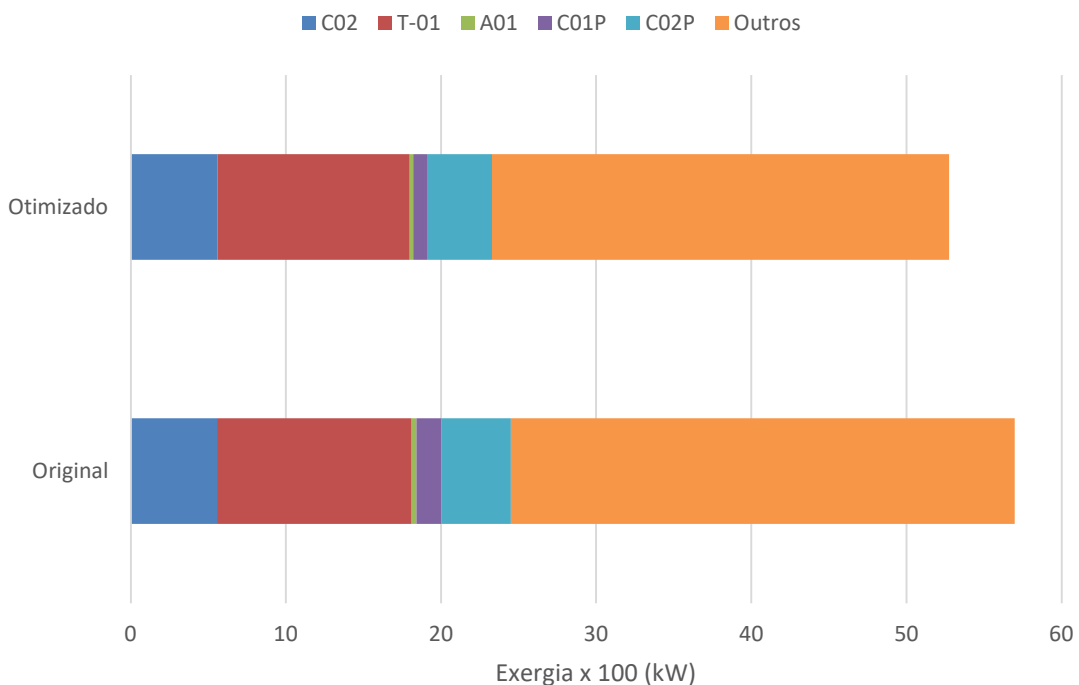


Figura 6.11: Perda de exergia na planta na condição original e ótima energética

A Figura 6.12 exibe o perfil de perda de exergia para a coluna T01 na (a) condição original de operação e na (b) condição ótima energética obtida. Observa-se por esta figura que a coluna em sua condição ótima possui uma distribuição mais equilibrada na perda de exergia. Portanto, as modificações realizadas a fim de se obter um mínimo consumo de energia acarretaram um perfil de exergia mais equilibrada na coluna para esta composição rica do gás bruto.

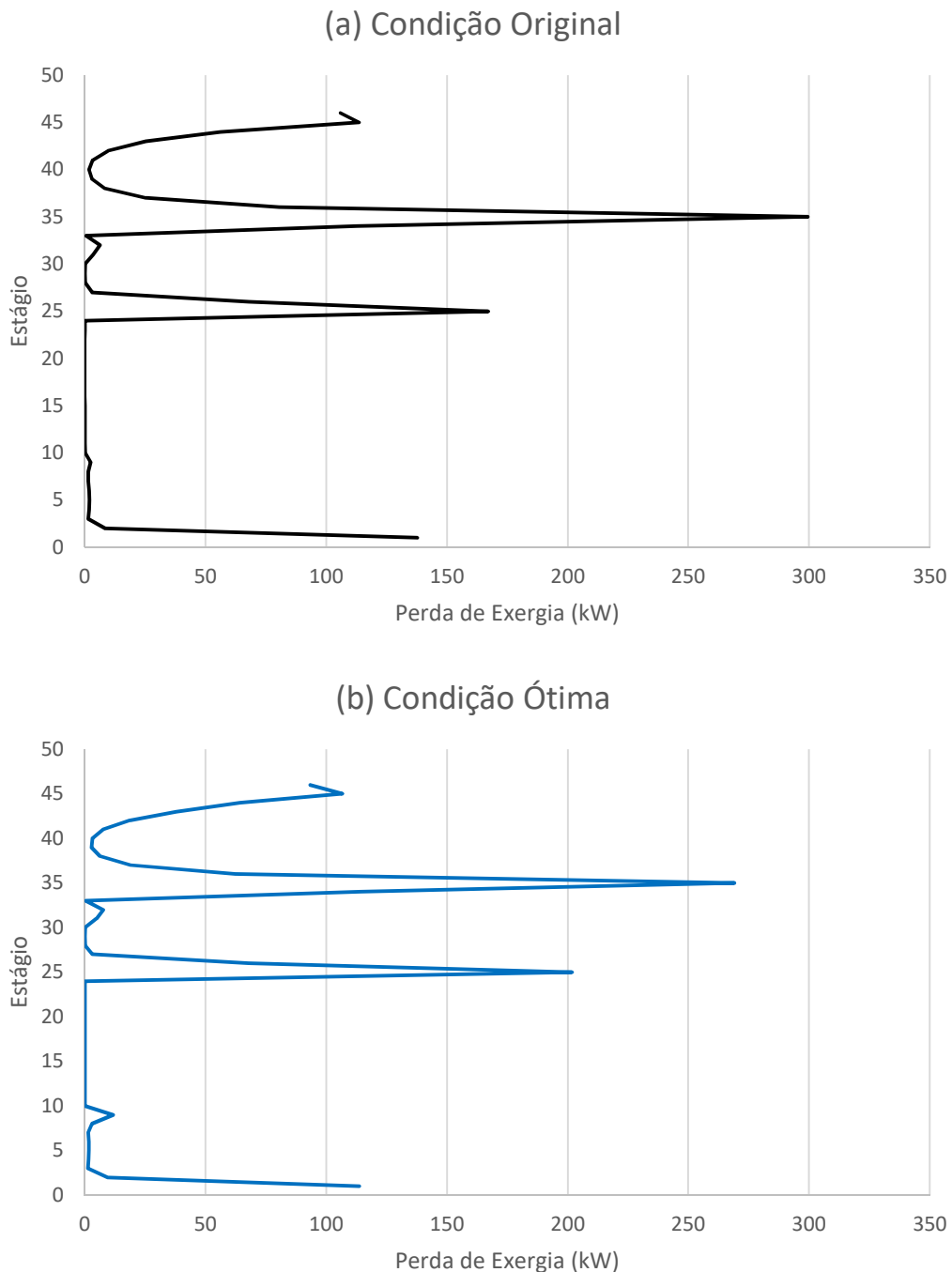


Figura 6.12: Perfil de perda de exergia para a coluna T01

A Tabela 6.12 apresenta a condição das correntes de entrada da coluna T01 para a condição original operacional e para a condição ótima energética obtida para a planta.

Tabela 6.12: Condição operacional da coluna T01

Corrente	76		74		58		46	
Estágio	9		26		32		35	
Condição	Original	Ótima	Original	Ótima	Original	Ótima	Original	Ótima
Temp. (°C)	-58,27	-65,31	-63,54	-60,60	-21,25	-16,82	-37,06	-31,22
Vazão (kmol/h)	62,00	222,33	2201,33	2181,13	63,71	69,21	1138,15	992,51

Pelos dados apresentados para esta otimização percebe-se a relevante influência da elevação da temperatura da corrente 48 de -17 para -11,8 °C em diversos pontos:

- Diminuição da vazão da corrente 46: a vazão diminuiu de 1138,15 para 992,51 kmol/h e a temperatura aumentou de -37,06 para -31,22 °C.
- Redução do pico de perda de exergia observado no estágio 35 (correspondente a alimentação da corrente 46);
- Menor consumo de energia no ciclo de refrigeração: a energia dos 4 equipamentos do ciclo de refrigeração (A01, S03, C01P e C02P) consumiam um total de 9246,3 kW e após a otimização energética este valor reduziu para 7977,7 kW.
- Menor perda de exergia no ciclo de refrigeração: a perda de exergia dos equipamentos do ciclo de refrigeração reduziu de 1516,8 para 1262,9 kW.

Observa-se que a otimização energética cumpriu seu objetivo atingindo um patamar menor de consumo de energia para os dois casos analisados. Os resultados obtidos comprovam a importância de se realizar a otimização a fim de se conhecer quais modificações operacionais auxiliam na economia de energia na planta. No primeiro caso estudado a redução no consumo de energia foi de 4,8 %, já no segundo foi de 7,6 %.

Na condição de gás bruto com composição rica em pesados percebeu-se o maior impacto da otimização, sendo que se obteve um maior percentual de economia. Isto se deve que neste estudo, a planta estava dimensionada para a condição original de composição do gás bruto. Quando se realiza a modificação na composição, se afasta

do ponto de operação de projeto e, assim, se perde eficiência nos equipamentos da planta. Deste modo, a otimização se torna importante para se determinar a condição de operação mais eficiente correspondente à nova composição de gás bruto.

Como consequência da otimização energética se obteve uma redução na perda de exergia na planta, o que indica que a planta está operando com melhor aproveitamento das fontes de energia. Percebe-se também, uma forte correlação entre a otimização energética com a geração de um perfil de perda de exergia mais equilibrada na coluna T01.

6.4 Otimização da Perda de Exergia

Dentro deste capítulo são apresentados os resultados para a otimização com objetivo de minimizar as perdas de exergia da planta. Primeiramente, será visto a otimização para a composição original da UPGN e posteriormente para a composição rica em pesados.

6.4.1 Composição Original

A composição original está apresentada na Tabela 6.1.

A Tabela 6.13 mostra os valores das variáveis de decisão na condição original da planta e na condição final obtida após otimização exergetica.

Tabela 6.13: Valores das variáveis de decisão

Variáveis de decisão	Condição Original	Condição Final (Otimização Exergetica)
Pressão propano no P-02 (bar)	6,0	6,06
Pressão propano no P-04 (bar)	1,5	2,08
Temperatura corrente 48 (°C)	-17	-14,9
Pressão no Turboexpansor (bar)	26,4	27,03
Split para corrente 09	0,8	0,65

As Figura 6.13 e Figura 6.14 apresentam respectivamente os resultados da otimização exergetica frente à condição original de operação para o consumo de

energia e para a perda de exergia na planta. Pela Figura 6.13 nota-se que houve uma diminuição no consumo total de energia de 17356,2 para 16695,3 kW, o que representa uma economia de 3,8%. Já a Figura 6.14 exibe os dados de perdas de exergia para os equipamentos da UPGN em sua condição original de operação e sua condição ótima exergética obtida. Por esta figura, nota-se que houve uma diminuição na perda total de exergia de 4729,1 para 4462,6 kW, representando uma redução de 5,6%.

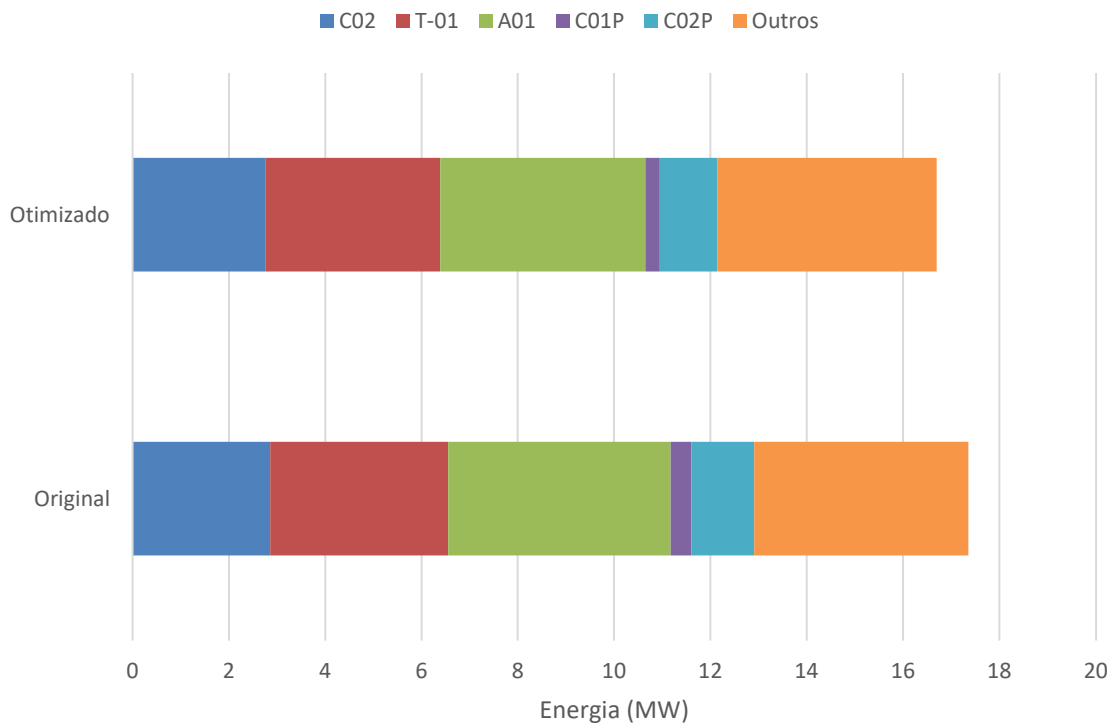


Figura 6.13: Consumo de energia da planta na condição original e ótima exergética

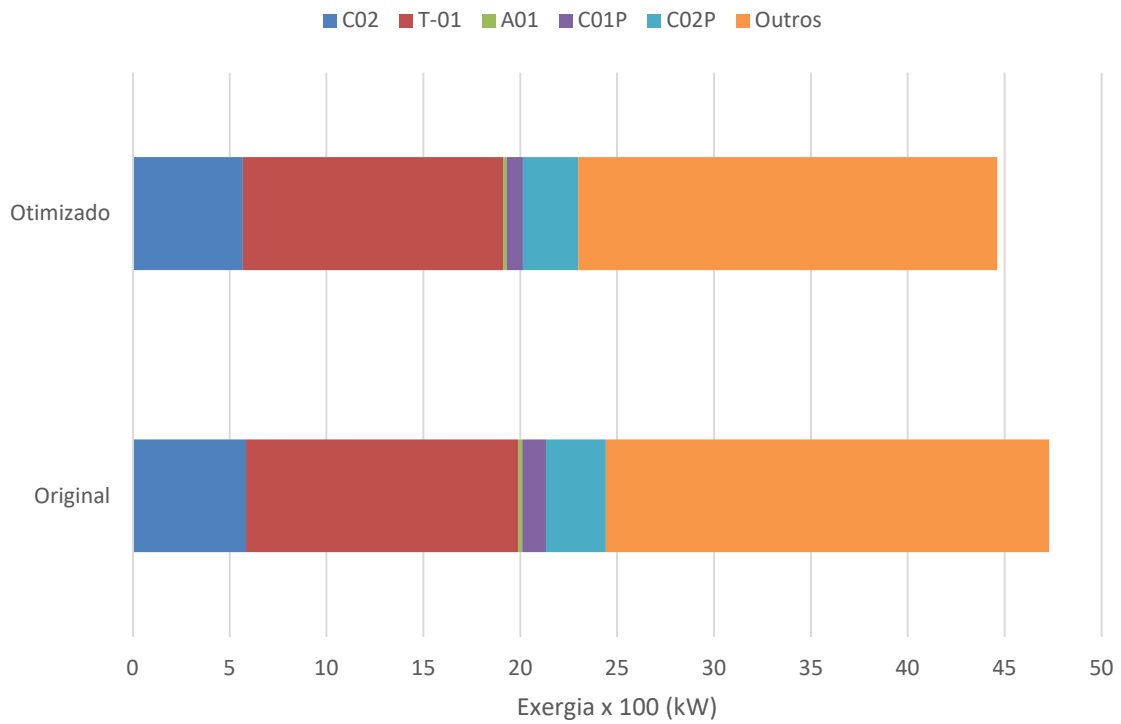


Figura 6.14: Perda de exergia na planta na condição original e ótima exergética

A Figura 6.15 exibe o perfil de perda de exergia para a coluna T01 na (a) condição original de operação e na (b) condição ótima, correspondente à mínima perda de exergia obtida. Observa-se também, que a coluna em sua condição ótima possui uma distribuição mais equilibrada no perfil de perda de exergia.

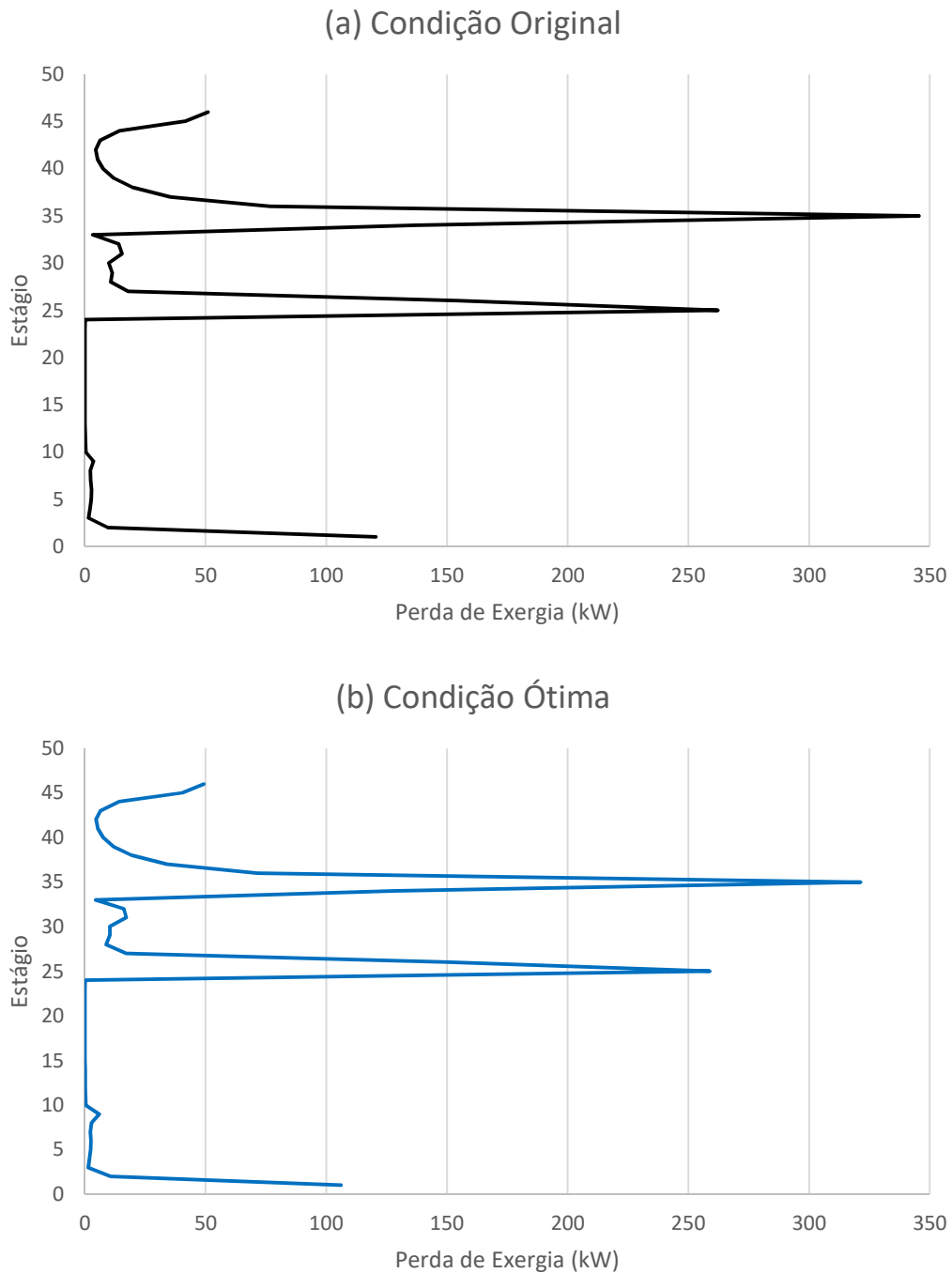


Figura 6.15: Perfil de perda de exergia para a coluna T01

Salienta-se que o perfil de perda de exergia obtido na otimização energética é mais equilibrada do que o obtido pela otimização com base na perda de exergia. No primeiro se atingiu um total de perda de exergia de 1328,7 kW, segundo Figura 6.8. Já no segundo, obteve-se 1344,2 kW, segundo Figura 6.14.

A Tabela 6.14 apresenta a condição das correntes de entrada da coluna T01 para a condição original operacional e para a condição ótima com base na perda de exergia obtida para a planta.

Tabela 6.14: Condição operacional da coluna T01

Corrente	76		74		58		46	
Estágio	8		25		31		34	
Condição	Original	Ótima	Original	Ótima	Original	Ótima	Original	Ótima
Temp. (°C)	-61,64	-64,29	-66,34	-65,20	-19,52	-17,62	-36,03	-33,09
Vazão (kmol/h)	74,33	125,65	2759,15	2746,43	52,60	54,74	578,92	538,18

Pelos dados apresentados para esta otimização percebe-se a relevante influência da elevação da temperatura da corrente 48 de -17 para -14,9 °C em diversos pontos:

- Diminuição da vazão da corrente 46: a vazão diminuiu de 578,92 para 538,18 kmol/h e a temperatura aumentou de -36,03 para -33,09 °C.
- Redução do pico de perda de exergia observado no estágio 35 (correspondente a alimentação da corrente 46):
- Menor consumo de energia no ciclo de refrigeração: a energia dos 4 equipamentos do ciclo de refrigeração (A01, S03, C01 e C02) consumiam um total de 6405,5 kW e após a otimização energética este valor reduziu para 5808,3 kW.
- Menor perda de exergia no ciclo de refrigeração: a perda de exergia dos equipamentos do ciclo de refrigeração reduziu de 1038,1 para 863,1 kW.

6.4.2 Composição Rica em Pesados

A composição rica em pesados está apresentada na Tabela 6.1.

A Tabela 6.15 mostra os valores das variáveis de decisão na condição original da planta e na condição final obtida através da otimização exergética.

Tabela 6.15: Valores das variáveis de decisão

Variáveis de decisão	Condição Original	Condição Final (Otimização Exergética)
Pressão propano no P-02 (bar)	6,0	5,8
Pressão propano no P-04 (bar)	1,5	2,36
Temperatura corrente 48 (°C)	-17	-11,96
Pressão no Turboexpansor (bar)	26,4	28,06
Split para corrente 09	0,8	0,25

A Figura 6.16 apresenta os dados de consumo de energia para a planta na sua condição original de operação e para sua condição ótima. Por esta figura, nota-se que houve uma diminuição no consumo total de energia de 24625,39 para 23291,04 kW, o que representa uma economia 5,4 %.

A Figura 6.17 apresenta os dados de perdas de exergia para os equipamentos da UPGN em sua condição original de operação e sua condição ótima energética obtida. Por esta tabela, nota-se que houve uma diminuição na perda total de exergia de 5699,5 para 5119,88 kW, o que representa uma redução de 10,2 %.

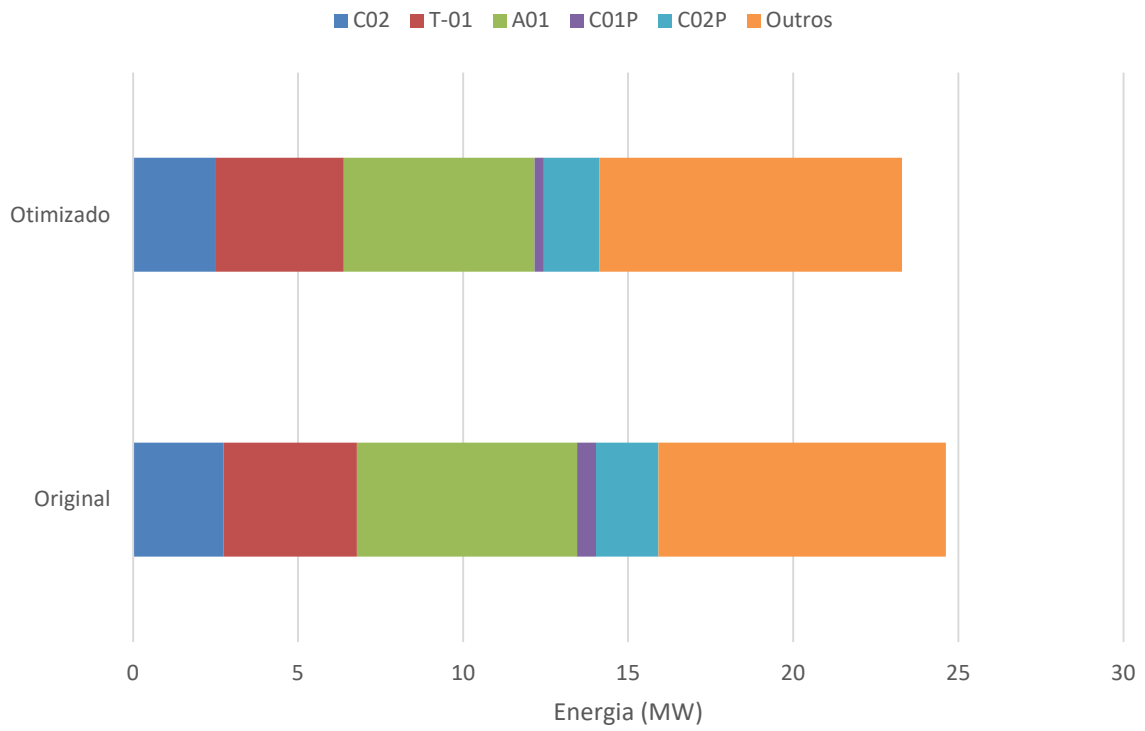


Figura 6.16: Consumo de energia da planta na condição original e ótima exergetica

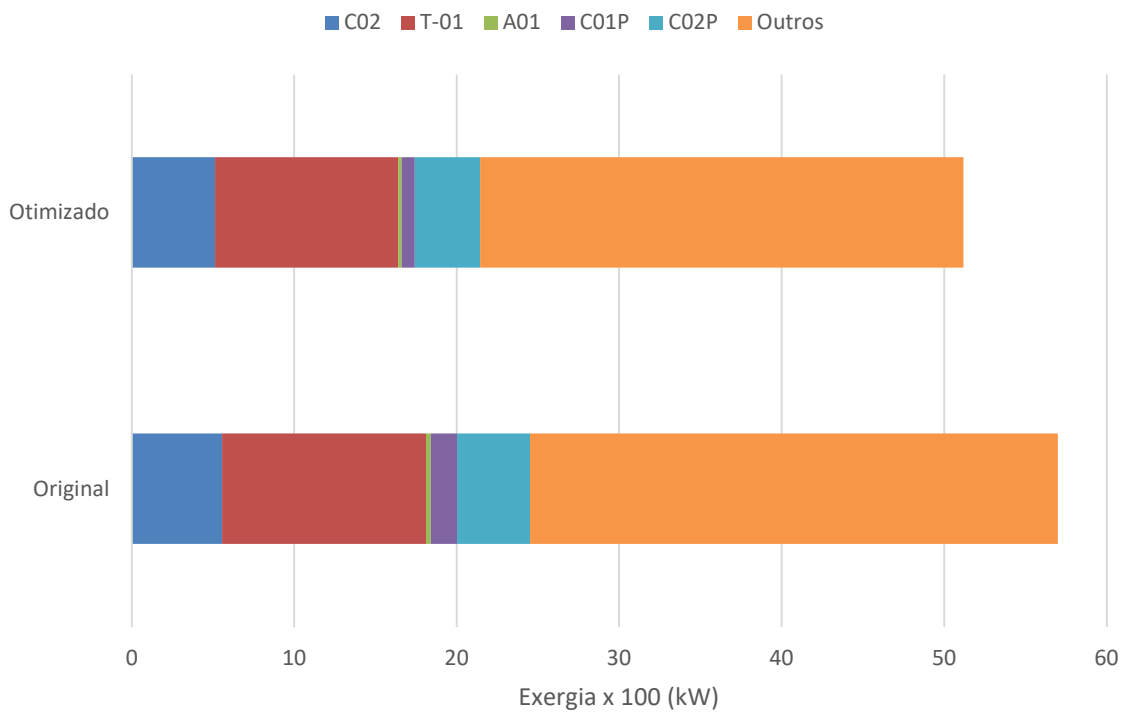


Figura 6.17: Perda de exergia na planta na condição original e ótima exergetica

A Figura 6.18 exibe o perfil de perda de exergia para a coluna T01 na (a) condição original de operação e na (b) condição ótima, correspondente à mínima perda de exergia obtida. Observa-se também, que a coluna em sua condição ótima possui uma distribuição mais equilibrada no perfil de perda de exergia.

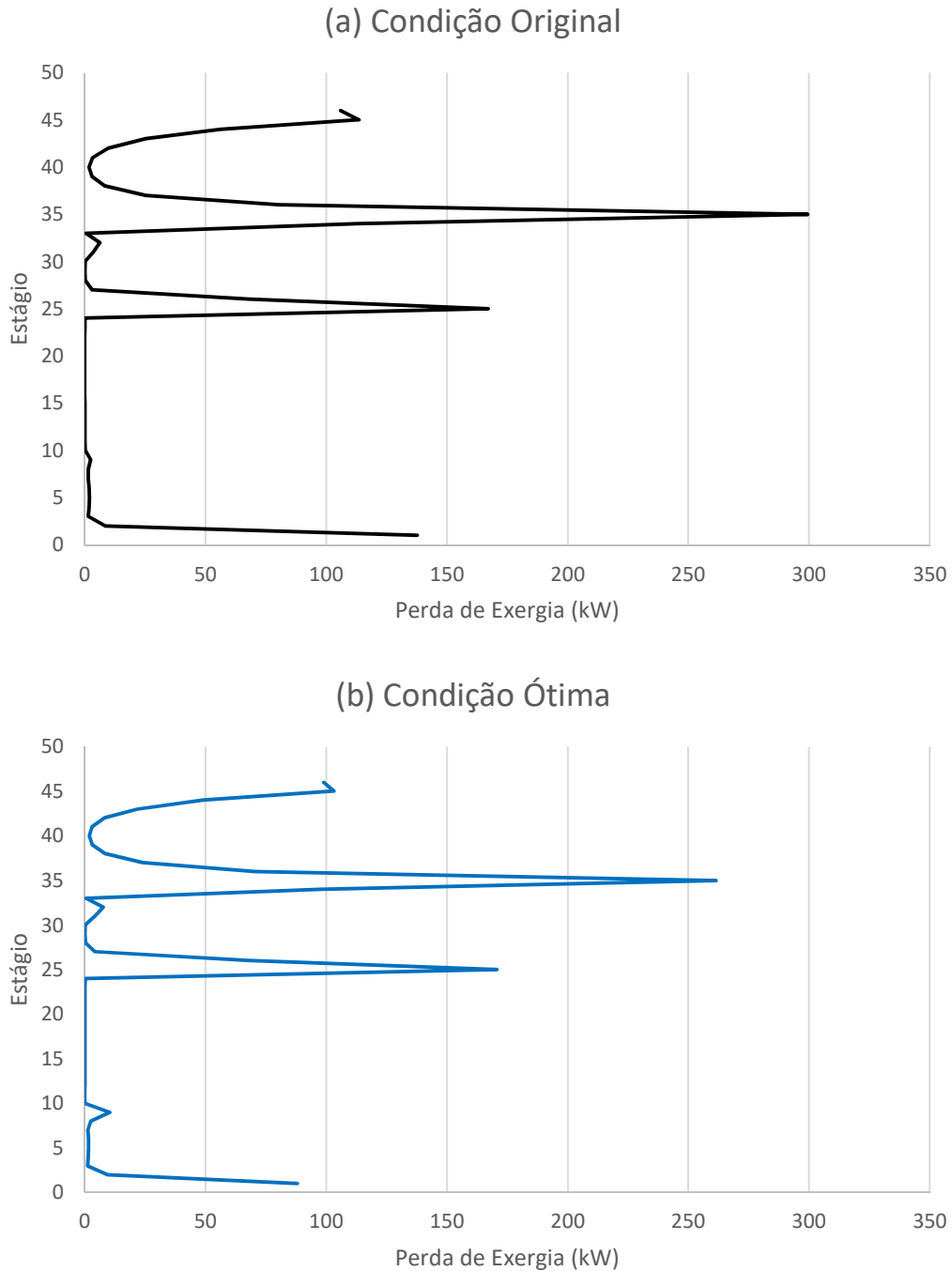


Figura 6.18: Perfil de perda de exergia para a coluna T01

A Tabela 6.16 apresenta a condição das correntes de entrada da coluna T01 para a condição original operacional e para a condição ótima com base na perda de exergia obtida para a planta.

Tabela 6.16: Condição operacional da coluna T01

Corrente	76		74		58		46	
Estágio	9		26		32		35	
Condição	Original	Ótima	Original	Ótima	Original	Ótima	Original	Ótima
Temp. (°C)	-58,27	-64,43	-63,54	-60,44	-21,25	-16,58	-37,06	-30,12
Vazão (kmol/h)	62,00	220,55	2201,33	2180,06	63,71	68,35	1138,15	996,23

Pelos dados apresentados percebe-se a relevante influência da elevação da temperatura da corrente 48 de -17 para -11,96 °C em diversos pontos:

- Diminuição da vazão da corrente 46: a vazão diminuiu de 1138,15 para 996,23 kmol/h e a temperatura aumentou de -37,06 para -30,12 °C.
- Redução do pico de perda de exergia observado no estágio 35 (correspondente a alimentação da corrente 46);
- Menor consumo de energia no ciclo de refrigeração: a energia dos 4 equipamentos do ciclo de refrigeração (A01, S03, C01 e C02) consumiam um total de 9246,3 kW e após a otimização energética este valor reduziu para 7802,6 kW.
- Menor perda de exergia no ciclo de refrigeração: a perda de exergia dos equipamentos do ciclo de refrigeração reduziu de 1516,8 para 1175,2 kW.

Percebe-se que a otimização com base na redução das perdas de exergia nos equipamentos cumpriu seu objetivo atingindo um patamar menor de perdas para os dois casos analisados. Estas duas últimas otimizações levaram a planta operar com um menor consumo de energia também.

Os resultados obtidos mostram que a otimização com foco na redução das perdas de exergia conduzem a planta em uma condição operacional mais eficiente e, como consequência, faz com que a planta consuma menos energia. Nestes casos verificados, a análise exérgica da planta se mostrou como uma boa métrica para avaliação do desempenho operacional da planta.

Da mesma forma que observado na otimização energética, também na exérgica se percebeu o maior impacto da otimização utilizando a condição de gás

bruto com composição rica em pesados. Obteve-se um maior percentual na redução de perdas de exergia para esta composição.

Através da Tabela 6.17, nota-se, como se esperava, que as perdas de exergia totais obtidas nestas duas últimas otimizações são menores que as perdas de exergia obtidas nas otimizações energéticas e os consumos de energias obtidos são maiores que os obtidos que os das otimizações energéticas.

Tabela 6.17: Comparativo entre resultados obtidos para as diferentes otimizações

	Otimização Energética		Otimização Perda de Exergia	
	Composição Original	Composição Rica	Composição Original	Composição Rica
Consumo Energia Total (kW)	16518,5 Redução 4,8 %	22757,6 Redução 7,6 %	16695,3 Redução 3,8 %	23291,0 Redução 5,4 %
Perdas Exergia Total (kW)	4477,8 Redução 5,3 %	5274,1 Redução 7,4 %	4462,6 Redução 5,6 %	5119,9 Redução 10,2 %

O entendimento do perfil de exergia da coluna T01 auxilia a se obter conclusões referentes a possíveis modificações operacionais na planta, tornando a operação mais eficiente, mesmo sem a realização de um algoritmo de otimização.

Como exemplo, observa-se que em todos os 4 casos de otimização estudados o resultado ótimo considerou um aumento na temperatura da corrente 48 e uma diminuição na vazão correspondente a corrente 9 no Split. Conforme mencionado, o perfil de perdas de exergia da coluna T01 em todas as situações analisadas apresenta uma maior perda de exergia na região de esgotamento da coluna. Assim, para se tornar esse perfil mais equilibrado se tem que aumentar as perdas de exergia na região de retificação e diminuir na de esgotamento.

A Figura 6.19 mostra no perfil de perdas de exergia, já exibida na Figura 6.12, os picos de perdas de exergia correspondente às correntes 46 e 76 que alimentam a coluna. Segundo sugerido, tem-se que tornar o perfil mais equilibrado aumentando as perdas na região de retificação e diminuindo na de esgotamento. Portanto, aumento a vazão da corrente 76, a qual se encontra na região de retificação, aumenta-se o pico de perda de exergia nesta região. Isto é atingido reduzindo-se a vazão da corrente 9 no Split, conforme é confirmado através dos resultados das otimizações. Outra modificação fácil de se observar através da Figura 6.19, é o benefício da diminuição do pico correspondente à corrente 46. Uma forma de redução deste pico é realizar o condicionamento da corrente, ou seja, já que a corrente se encontra na região de esgotamento, sugere-se pré-aquecer a carga da

entrada na coluna e, em consequência, a energia necessária no refeedor da coluna diminui. Esse pré-aquecimento é atingido ao se elevar a temperatura da corrente 48 (corrente de gás bruto após ser resfriada pelo ciclo de refrigeração). Esta condição também é observada em todos os resultados das otimizações.

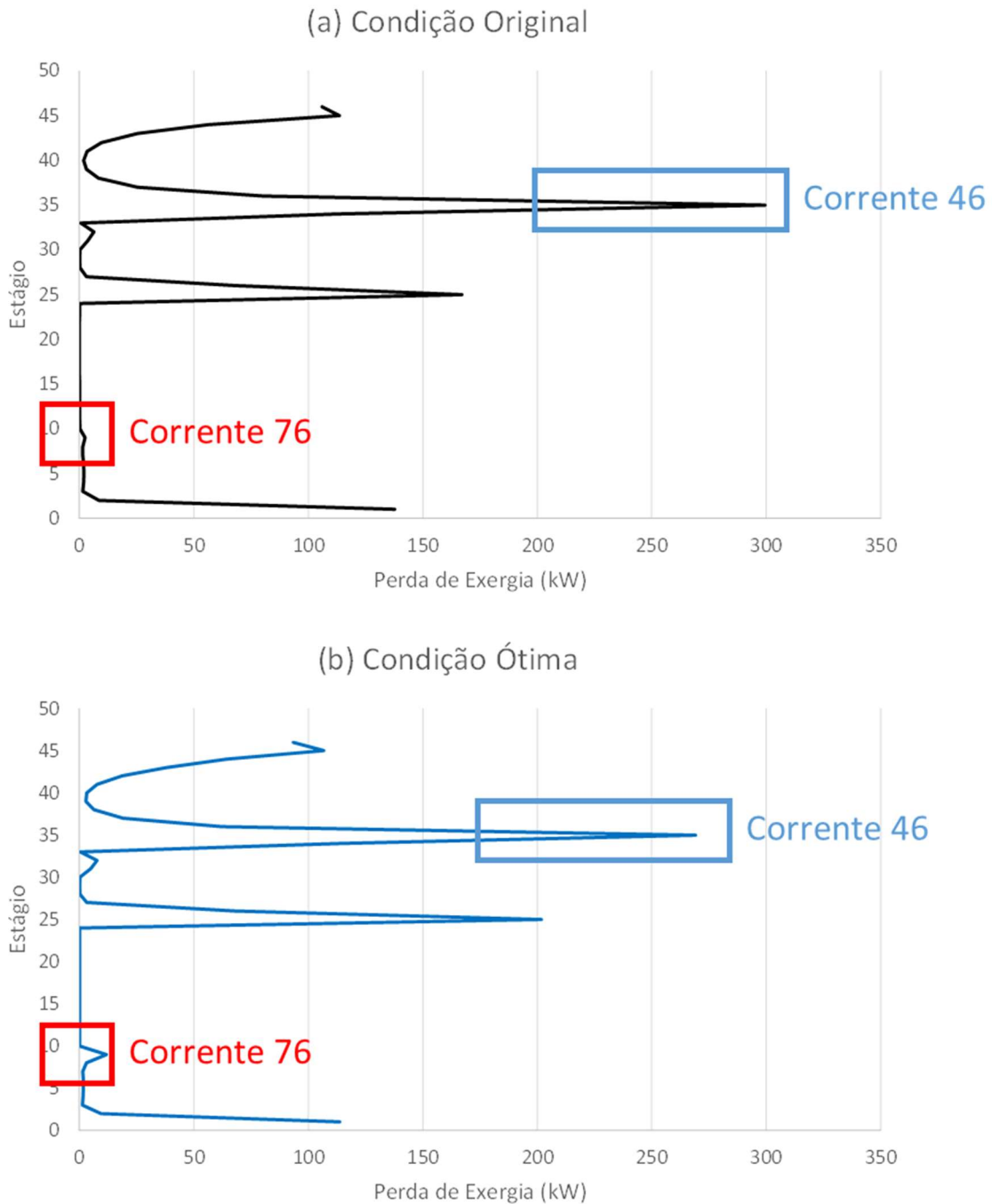


Figura 6.19: Análise dos perfis de perdas de exergias obtidos na otimização energética com a composição rica do gás bruto

7 Conclusões e Sugestões para Trabalhos Futuros

Neste capítulo são apresentadas as conclusões que foram obtidas através da realização deste trabalho. Posteriormente neste capítulo são sugeridos tópicos para trabalhos futuros.

7.1 Conclusões

Com os resultados que se obteve com este estudo, se conclui que a otimização com base na redução de perdas de exergia é uma boa métrica para avaliação da eficiência operacional da planta. Pois, para as duas situações analisadas, o ponto ótimo obtido através da análise de exergia resultou em pontos operacionais com melhores desempenhos no consumo de energia. Mas, como pode ser percebido através dos resultados, se o objetivo da otimização é obter maior economia de energia, o procedimento a se adotar é a realização de otimização energética.

A vantagem do uso da análise de exergia é a maior informação que esta variável traz sobre o desempenho da planta. A análise energética exibe os consumos de energia dos equipamentos. Mas não se tem diretrizes, com esta informação, o quanto se pode melhorar a operação da planta. Já a análise de exergia, por ser uma medida de irreversibilidade, se tem indícios para poder se melhorar o processo. Em um processo reversível, ou seja, a melhor eficiência possível, a perda de exergia é

zero. Portanto, quanto mais afastados do zero a perda de exergia de um processo estiver, mais irreversível é este processo. Em suma, cabe à avaliação exérgica determinar os pontos de maiores irreversibilidades que se deve atuar para se alcançar uma melhor operação da planta do ponto de vista energético.

Em casos em que não se pode realizar uma otimização, os dados de perda de exergia são bastante importantes pois mostram onde se é mais relevante atuar para melhorar a eficiência da planta. É possível criar variáveis para a perda de exergia de equipamentos em uma planta e criar uma malha de controle para esta variável.

Como visto no capítulo 2.1, o cálculo necessário para se obter a perda de exergia de um equipamento é simples, desde que se possua as propriedades termodinâmicas das correntes envolvidas. Estas propriedades podem ser obtidas por simuladores de processo e até por *plugins* a serem adicionadas em softwares de planilhas eletrônicas. Assim, se torna possível realizar um mapeamento das perdas de exergias de um processo e, a partir disto, se estabelecer *setpoint* para variáveis criadas a partir deste mapeamento. Conhecendo-se as variáveis de perdas de exergia que possuem maior correlação com custos operacionais em uma planta, cria-se malhas de controles mais focadas em pontos estratégicos da planta.

O estabelecimento de malhas de controle conforme discutido neste capítulo se assemelha com a metodologia SOC (*Self-Optimizing Control*) discutida no capítulo 3.2.2. Portanto, é possível criar, a partir da análise de perdas de exergia da planta, esta metodologia de controle que visa manter a operação da planta próxima ao seu ponto ótimo, mantendo constante a variável para perda de exergia em um valor previamente estabelecido de *setpoint*. Assim, após um distúrbio na planta, a malha de controle atua para manter constante o *setpoint* da variável de perda de exergia. A implementação da metodologia SOC possui como vantagem um custo menor que a metodologia RTO (*Real-Time Optimization*).

Segundo os pré-requisitos para um SOC, faz-se necessário a determinação de um *setpoint* constante na variável que atuará a malha de controle. Uma variável estabelecida a partir da perda de exergia para ser o *setpoint* constante do SOC apresenta vantagem em relação a uma variável estabelecida a partir do consumo de energia. Independente da condição operacional de uma planta, a perda de exergia possui o valor constante de zero correspondente ao seu ponto ótimo. Já para o consumo de energia existem diferentes valores de mínimo consumo em função de uma nova condição operacional da planta. Desta forma, um *setpoint* constante estabelecido para o consumo ótimo de energia em uma condição operacional não será mais ótimo em uma nova condição após um distúrbio.

Este trabalho evidencia a importância na atuação em variáveis específicas que levam a coluna de destilação operar mais eficientemente. Observou-se que nas duas otimizações energéticas obteve-se como resultado perfis de perdas de exergia mais

equilibrados na coluna T01. Inclusive, no caso da otimização energética para o gás bruto na condição original, capítulo 6.3.1, a perda de exergia atingida na coluna T01, Figura 6.8, foi menor que a perda de exergia alcançada para esta mesma coluna, Figura 6.14, após realizada a otimização das perdas de exergia da UPGN.

7.2 Sugestões para Trabalhos Futuros

De acordo com o fato observado que a otimização energética atingiu uma menor perda de exergia na coluna T01 que a otimização das perdas de exergia, sugere-se a realização de um novo estudo onde a otimização com base nas perdas de exergia seja mais focada nos equipamentos mais relevantes sob ponto de vista de consumo energético.

Este novo estudo analisaria se esta nova otimização de perdas de exergia focada nos equipamentos mais relevantes, ou seja, aqueles com maior consumo de energia, resultaria em um consumo de energia mais próximo ao da otimização energética. Em caso positivo, uma análise de perdas de exergia focada nos equipamentos mais relevantes seria uma melhor métrica para a avaliação do desempenho operacional da planta.

Outra sugestão para novos estudos é o de realizar a avaliação da análise de perdas de exergia frente ao custo econômico operacional. Pois conforme os resultados de consumo de energia, se observa que o equipamento com maior consumo é o *Air Cooler A01*. Normalmente, este tipo de equipamento, não apresenta um custo econômico operacional tão elevado sendo mais relevante que a otimização foque em reduzir o consumo energético em outros equipamentos. Com uma otimização econômica, pode-se comparar as perdas de exergia dos equipamentos frente ao seu custo financeiro operacional e estabelecer diferentes pesos às perdas de exergia dos equipamentos de acordo com estes custos, ou seja, se criaria um KPI exergético ponderado, onde o fator de ponderação estaria vinculado ao custo operacional.

Referências

ALHAJJI, M.; DEMIREL, Y. Energy intensity and environmental impact metrics of the back-end separation of ethylene plant by thermodynamic analysis. **International Journal of Energy and Environmental Engineering**, v. 7, n. 1, p. 45–59, 2016.

ALKAYA, D.; VASAMTHARAJAN, S.; BIEGLER, L. T. Successive quadratic programming: applications in the process industry. In: FLOUDAS, C. A.; PARDALOS, P. M. (Eds.). . **Encyclopedia of Optimization**. [s.l.] Springer, 2008. p. 3853–3864.

BARALDI, A. **Análise, Modelagem e Otimização do Ciclo de Refrigeração de uma Unidade de Processamento de Gás Natural**. [s.l.] Universidade Federal do Rio Grande do Sul, 2015.

BEJAN, A. **Entropy Generation Through Heat and Fluid Flow**. First Edit ed. New York: Wiley, 1982.

DARBY, M. L. et al. RTO: An overview and assessment of current practice. **Journal of Process Control**, v. 21, n. 6, p. 874–884, 2011.

DHOLE, V. .; LINNHOFF, B. Distillation Column Targets. **Computers & Chemical Engineering**, v. 17, p. 549–560, 1993.

EDGAR, T. F.; HIMMELBLAU, D. M.; LASDON, L. S. **Optimization of Chemical Processes**. Second ed. New York: McGraw-Hill, 2001.

- ENGELL, S. Feedback control for optimal process operation. **Journal of Process Control**, v. 17, n. 3, p. 203–219, 2007.
- FLECK, T. D. **Nova Metodologia para Desenvolvimento de Inferências Baseadas em Dados**. [s.l.] Universidade Federal do Rio Grande do Sul, 2012.
- GUO, B.; GHALAMBOR, A. **Natural Gas Engineering Handbook**. first ed. Houston: Gulf Publishing Company, 2005.
- KEMP, I. C. **Pinch Analysis and Process Integration**. Second ed. [s.l.] Butterworth-Heinemann, 2007.
- KORETSKY, M. D. **Engineering and Chemical Thermodynamics**. Second ed. [s.l.] Wiley, 2013.
- MIZOGUSHI, A. ; MARLINS, T. E.; HRYMAK, A. N. Distillation Process. **the Canadian Journal of Chemical Engineering**, v. 73, 1995.
- MOKHTAB, S.; POE, W. A.; MAK, J. Y. **Handbook of Natural Gas Transmission and Processing**. [s.l: s.n.]. v. 1
- MORAN, M. J.; SHAPIRO, H. N. **Fundamentals of Engineering Thermodynamics**. [s.l: s.n.].
- MOUSSA, L. S. **Análise Termodinâmica de Colunas de Destilação Visando à Otimização Energética**. [s.l.] Universidade Estadual de Campinas, 2001.
- NGUYEN, N.; DEMIREL, Y. Retrofit of distillation columns in biodiesel production plants. **Energy**, v. 35, n. 4, p. 1625–1632, 2010.
- OFFICE OF ENERGY EFFICIENCY & RENEWABLE ENERGY. **AMO Success Story: New Software Will Enable Chemical Manufacturers to Optimize Distillation Column Configuration and Save Energy**. Disponível em: <<https://energy.gov/eere/amo/articles/amo-success-story-new-software-will-enable-chemical-manufacturers-optimize>>. Acesso em: 9 mar. 2017.
- PINTO, F. S. et al. Thermodynamic optimisation of distillation columns. **Chemical Engineering Science**, v. 66, n. 13, p. 2920–2934, 2011.
- SCHULTZ, E. DOS S. **A importância do ponto de operação nas técnicas de Self-optimizing Control**. [s.l.] Universidade Federal do Rio Grande do Sul, 2015.

SEQUEIRA S.E.; GRAELLS, M. . P. L. Real-Time Evolution for On-line Optimization of Continuous Processes. **Industrial and Chemical Engineering Research**, v. 41, p. 1815–1825, 2002.

SHIN, J.; YOON, S.; KIM, J. K. Application of exergy analysis for improving energy efficiency of natural gas liquids recovery processes. **Applied Thermal Engineering**, v. 75, p. 967–977, 2015.

SKOGESTAD, S. Plantwide control: The search for the self-optimizing control structure. **Journal of Process Control**, v. 10, n. 5, p. 487–507, 2000a.

SKOGESTAD, S. Self-optimizing control: the missing link between steady-state optimization and control. **Computers & Chemical Engineering**, v. 24, n. 2–7, p. 569–575, 2000b.

TRIERWEILER, J. O. **Real-Time Optimization of Industrial Processes**. London: Springer London, 2014.

VAZ, C. E. M.; MAIA, J. L. P.; DOS SANTOS, W. G. **Tecnologia da Indústria do Gás Natural**. 1. ed. [s.l.] Edgard Blucher, 2008.

ZEMP, R. J.; DE FARIA, S. H. B. **Improving thermal efficiency of distillation columns using exergy analysis**. Iberian Latin American Conference on Computational Methods for Engineering. **Anais...Curitiba**: 1995

ZEMP, R. J.; DE FARIA, S. H. B.; OLIVEIRA MAIA, M. D. L. Driving force distribution and exergy loss in the thermodynamic analysis of distillation columns. **Computers & Chemical Engineering**, v. 21, n. Figure 1, p. S523–S528, 1997.

APÊNDICES – Scripts Escritos em Python

Código em Python para realizar a interface com Aspen Plus

```
C:\Users\Severo\OneDrive\Profissional\Mestrado\Tese\4. UPGN-Python\14. New UPGN - python ciclo\init.py Friday, June 01, 2018 14:58  
import os  
import win32com.client as win32  
  
#import importlib; importlib.reload(init) #comando para recarregar modulo caso haja alteracao  
  
aspen = win32.Dispatch('Apwn.Document')  
  
if __name__ == '__main__':  
    aspen.InitFromArchive2(os.path.abspath('C:\\Users\\Severo\\OneDrive\\Profissional\\Mestrado\\Tese\\4. UPGN-Python\\14. New UPGN - python ciclo\\0. Aspen\\New UPGN 8.bkp'))
```

Código para obter o perfil de exergia da coluna T-01

C:\Users\Severo\OneDrive\Profissional\Mestrado\Tese4. UPGN-Python\14. New UPGN - python cicloExergy.py

Friday, June 01, 2018 15:20

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 27 11:40:12 2016

@author: Severo
"""

import numpy as np
import pandas as pd
import scipy.integrate
import importlib; importlib.reload(Exergy) #comando para recarregar modulo caso haja alteracao

#blk = "T-01A" #block name
#blk = "T-01" #block name
#blk = "B1" #block name

def exergia(aspern,blk):

    stages_number =
    aspen.Tree.FindNode("\\Data\\Blocks\\{0}\\Input\\{1}".format(blk,'NSTAGE')).Value

    Ex = []

    # P-71
    streams_P71 = [73, 74, 69, 71]
    exergy_P71 = []
    for i in streams_P71:
        C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
        exergy_P71.append(C)
    exergy_P71 = np.asarray(exergy_P71)
    Exloss_P71 = exergy_P71[0]+exergy_P71[2]-exergy_P71[1]-exergy_P71[3]
    Exloss_P71 = Exloss_P71 * (1/0.0041868)

    # P-72
    streams_P72 = [75, 76, 84, 69]
    exergy_P72 = []
    for i in streams_P72:
        C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
        exergy_P72.append(C)
    exergy_P72 = np.asarray(exergy_P72)
    Exloss_P72 = exergy_P72[0]+exergy_P72[2]-exergy_P72[1]-exergy_P72[3]
    Exloss_P72 = Exloss_P72 * (1/0.0041868)

    # V-06
    streams_V06 = [71,21,36]
    exergy_V06 = []
    for i in streams_V06:
        C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
        exergy_V06.append(C)
    Tc = aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Output\\B_TEMP").Value + 273.15
    C = aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Output\\QCALC").Value * 0.0041868
    if Tc > 298.15:
        Qc = C*(1-(298.15/Tc))
    else:
```

```
Qc = C*(1-(Tc/298.15))
exergy_V06.append(Qc)
exergy_V06 = np.asarray(exergy_V06)
Exloss_V06 = exergy_V06[0]-exergy_V06[2]-exergy_V06[1]-exergy_V06[3]
Exloss_V06 = Exloss_V06 * (1/0.0041868)

Ex_cond = Exloss_P71 + Exloss_P72 + Exloss_V06
Ex.append(Ex_cond)

for i in range (1,stages_number+1,1):
    C = aspen.Tree.FindNode("\\Data\\Blocks\\{0}\\Output\\EXERGY\\{1}".format(blk, i)).Value
    Ex.append(C)

#Ex = np.asarray(Ex)/1000 # converter para kcal/sec
Ex = np.asarray(Ex) * 0.0041868 #conversao de unidade de cal/sec para kW
#Ex = np.asarray(Ex) * 14.286/1000000 #conversao de unidade de cal/sec para MMBtu/hr
#Ex = pd.Series(Ex)

A_Ex = scipy.integrate.simps(Ex)

return Ex, A_Ex
```

Código para obter o consumo de energia dos equipamentos da UPGN

```
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 21 16:02:10 2017

@author: Severo
"""

import numpy as np

#import importlib; importlib.reload(Energy) #comando para recarregar modulo caso haja alteracao

def Energy(aspern):

    # turbo expensor TE01
    energy_TE01 = aspern.Tree.FindNode("\\Data\\Blocks\\TE-01\\Output\\WNET").Value

    # compressor C-01
    energy_C01 = aspern.Tree.FindNode("\\Data\\Blocks\\C-01\\Output\\IND_POWER").Value

    # compressor C-02
    energy_C02 = aspern.Tree.FindNode("\\Data\\Blocks\\C-02\\Output\\IND_POWER").Value

    # V-06
    energy_V06 = aspern.Tree.FindNode("\\Data\\Blocks\\V-06\\Output\\QCALC").Value * 0.0041868

    #Coluna T-01
    energy_T01 = aspern.Tree.FindNode("\\Data\\Blocks\\T-01\\Output\\REB_DUTY").Value *
    0.0041868

    #Coluna T-02
    energy_T02 = aspern.Tree.FindNode("\\Data\\Blocks\\T-02\\Output\\REB_DUTY").Value *
    0.0041868

    # P-10
    energy_P10 = aspern.Tree.FindNode("\\Data\\Blocks\\P-10\\Output\\QCALC").Value * 0.0041868

    # bomba B-08
```

```

energy_B08 = aspen.Tree.FindNode("\\Data\\Blocks\\B8\\Output\\WNET").Value

#Coluna T-03
energy_T03 = aspen.Tree.FindNode("\\Data\\Blocks\\T-03\\Output\\REB_DUTY").Value *
0.0041868

# B3
energy_B3 = aspen.Tree.FindNode("\\Data\\Blocks\\B3\\Output\\QCALC").Value * 0.0041868

# condensador A-01
energy_A01 = aspen.Tree.FindNode("\\Data\\Blocks\\A01\\Output\\QCALC").Value * 0.0041868

# vaso S03
energy_S03 = aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Output\\QCALC").Value * 0.0041868

# compressor C-01P ciclo de refrigeracao
energy_C01P = aspen.Tree.FindNode("\\Data\\Blocks\\C01\\Output\\IND_POWER").Value

# compressor C-02P ciclo de refrigeracao
energy_C02P = aspen.Tree.FindNode("\\Data\\Blocks\\C02\\Output\\IND_POWER").Value

#trocadores para ajuste de temperatura da corrente de saida da UPGN
energy_B11 = aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Output\\QCALC").Value * 0.0041868
energy_B12 = aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Output\\QCALC").Value * 0.0041868
energy_B13 = aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Output\\QCALC").Value * 0.0041868
energy_B14 = aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Output\\QCALC").Value * 0.0041868

energy_tot = (abs(energy_C02) + abs(energy_V06) + abs(energy_T01) +
abs(energy_T02) + abs(energy_P10) + abs(energy_B08) + abs(energy_T03) +
abs(energy_B3) + abs(energy_A01) + abs(energy_S03) + abs(energy_C01P) +
abs(energy_C02P) + abs(energy_B11)+ abs(energy_B12)+ abs(energy_B13)+
abs(energy_B14))

```



```
return (energy_TE01, energy_C01, energy_C02, energy_V06, energy_T01, energy_T02,  
        energy_P10, energy_B08, energy_T03, energy_B3, energy_A01,  
        energy_S03, energy_C01P, energy_C02P, energy_B11, energy_B12,  
        energy_B13, energy_B14, energy_tot)
```

Código para calcular as perdas de exergia dos equipamentos da UPGN

```
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 14 15:50:26 2017

@author: Senaro
"""

import numpy as np
import Exergy
#import importlib; importlib.reload(Ex_loss) #comando para recarregar modulo caso haja alteracao

def Exloss(aspern):

    # P-01
    streams = [30,31,7,62]

    exergy_P01 = []

    for i in streams:
        C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
at(i)).Value
        exergy_P01.append(C)

    exergy_P01 = np.asarray(exergy_P01)

    Exloss_P01 = exergy_P01[0]+exergy_P01[3]-exergy_P01[1]-exergy_P01[2]

    # P-02
    streams_P02 = [31,32,'6P','7P']

    exergy_P02 = []

    for i in streams_P02:
        C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
at(i)).Value
        exergy_P02.append(C)

    exergy_P02 = np.asarray(exergy_P02)

    Exloss_P02 = exergy_P02[0]+exergy_P02[2]-exergy_P02[1]-exergy_P02[3]

    # P-03
    streams_P03 = [32,47,61,62]

    exergy_P03 = []

    for i in streams_P03:
        C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
at(i)).Value
        exergy_P03.append(C)

    exergy_P03 = np.asarray(exergy_P03)

    Exloss_P03 = exergy_P03[0]+exergy_P03[2]-exergy_P03[1]-exergy_P03[3]
```

```

# P-04
streams_P04 = [47,48,'13P','14P']

exergy_P04 = []

for i in streams_P04:
    C =
    aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".format(i)).Value
    exergy_P04.append(C)

exergy_P04 = np.asarray(exergy_P04)
Exloss_P04 = exergy_P04[0]+exergy_P04[2]-exergy_P04[1]-exergy_P04[3]

# V-02
streams_V02 = [48,49,15]

exergy_V02 = []

for i in streams_V02:
    C =
    aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".format(i)).Value
    exergy_V02.append(C)

exergy_V02 = np.asarray(exergy_V02)
Exloss_V02 = exergy_V02[0]-exergy_V02[2]-exergy_V02[1]

# Valvula B4
streams_B4 = [15, 46]

exergy_B4 = []

for i in streams_B4:
    C =
    aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".format(i)).Value
    exergy_B4.append(C)

exergy_B4 = np.asarray(exergy_B4)
Exloss_B4 = exergy_B4[0]-exergy_B4[1]

# P052
streams_P052 = [52,53,6,58]

```

```

exergy_P52 = []

for i in streams_P52:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
    exergy_P52.append(C)

exergy_P52 = np.asarray(exergy_P52)

Exloss_P52 = exergy_P52[0]+exergy_P52[2]-exergy_P52[1]-exergy_P52[3]

# P051
streams_P51 = [51,54,60,61]

exergy_P51 = []

for i in streams_P51:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
    exergy_P51.append(C)

exergy_P51 = np.asarray(exergy_P51)

Exloss_P51 = exergy_P51[0]+exergy_P51[2]-exergy_P51[1]-exergy_P51[3]

# V-03
streams_V03 = [55,56,59]

exergy_V03 = []

for i in streams_V03:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
    exergy_V03.append(C)

exergy_V03 = np.asarray(exergy_V03)

Exloss_V03 = exergy_V03[0]-exergy_V03[2]-exergy_V03[1]

# Valvula B10
streams_B10 = [56, 6]

exergy_B10 = []

for i in streams_B10:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form

```

```

        at(i).Value
        exergy_B10.append(C)
exergy_B10 = np.asarray(exergy_B10)
Exloss_B10 = exergy_B10[0]-exergy_B10[1]

# P-06
streams_P06 = [59, 17, 21, 60]
exergy_P06 = []
for i in streams_P06:
    C =
    aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
    at(i).Value
    exergy_P06.append(C)
exergy_P06 = np.asarray(exergy_P06)
Exloss_P06 = exergy_P06[0]+exergy_P06[2]-exergy_P06[1]-exergy_P06[3]

# V-04
streams_V04 = [17,18,63]
exergy_V04 = []
for i in streams_V04:
    C =
    aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
    at(i).Value
    exergy_V04.append(C)
exergy_V04 = np.asarray(exergy_V04)
Exloss_V04 = exergy_V04[0]-exergy_V04[2]-exergy_V04[1]

# turbo expensor TE01
streams_TE01 = [18, 19]
exergy_TE01 = []
for i in streams_TE01:
    C =
    aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
    at(i).Value
    exergy_TE01.append(C)

```

```

C = aspen.Tree.FindNode("\\Data\\Blocks\\TE-01\\Output\\WNET").Value
exergy_TE01.append(C)
exergy_TE01 = np.asarray(exergy_TE01)

Exloss_TE01 = exergy_TE01[0]-exergy_TE01[1]+exergy_TE01[2]

# V-05
streams_V05 = [19, 64, 66]

exergy_V05 = []

for i in streams_V05:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".format(i)).Value
    exergy_V05.append(C)

exergy_V05 = np.asarray(exergy_V05)

Exloss_V05 = exergy_V05[0]-exergy_V05[2]-exergy_V05[1]

# P-71
streams_P71 = [73, 74, 69, 71]

exergy_P71 = []

for i in streams_P71:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".format(i)).Value
    exergy_P71.append(C)

exergy_P71 = np.asarray(exergy_P71)

Exloss_P71 = exergy_P71[0]+exergy_P71[2]-exergy_P71[1]-exergy_P71[3]

# P-72
streams_P72 = [75, 76, 34, 69]

exergy_P72 = []

for i in streams_P72:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".format(i)).Value
    exergy_P72.append(C)

```

```

exergy_P72 = np.asarray(exergy_P72)
Exloss_P72 = exergy_P72[0]+exergy_P72[2]-exergy_P72[1]-exergy_P72[3]

# compressor C-01
streams_C01 = [7, 8]
exergy_C01 = []
for i in streams_C01:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
            at(i)).Value
        exergy_C01.append(C)
C = aspen.Tree.FindNode("\\Data\\Blocks\\C-01\\Output\\IND_POWER").Value
exergy_C01.append(C)
exergy_C01 = np.asarray(exergy_C01)
Exloss_C01 = exergy_C01[0]-exergy_C01[1]+exergy_C01[2]

# compressor C-02
streams_C02 = [8, 'GAS-RES']
exergy_C02 = []
for i in streams_C02:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
            at(i)).Value
        exergy_C02.append(C)
C = aspen.Tree.FindNode("\\Data\\Blocks\\C-02\\Output\\IND_POWER").Value
exergy_C02.append(C)
exergy_C02 = np.asarray(exergy_C02)
Exloss_C02 = exergy_C02[0]-exergy_C02[1]+exergy_C02[2]

# V-06
streams_V06 = [71,21,36]
exergy_V06 = []

```

```

for i in streams_V06:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".format(i)).Value
    exergy_V06.append(C)

Tc = aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Output\\B_TEMP").Value + 273.15
C = aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Output\\QCALC").Value * 0.0041868

if Tc > 298.15:
    Qc = C*(1-(298.15/Tc))
else:
    Qc = C*(1-(Tc/298.15))

exergy_V06.append(Qc)
exergy_V06 = np.asarray(exergy_V06)

Exloss_V06 = exergy_V06[0]-exergy_V06[2]-exergy_V06[1]-exergy_V06[3]

#print (exergy_V06,Tc,C,Qc, Exloss_V06)
#print (exergy_V06)

#Columna T-01
Exloss_T01 = np.sum(Exergy.exergia(aspen,'T-01')[0])

# Valvula B2
streams_B2 = [16, 22]

exergy_B2 = []

for i in streams_B2:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".format(i)).Value
    exergy_B2.append(C)

exergy_B2 = np.asarray(exergy_B2)

Exloss_B2 = exergy_B2[0]-exergy_B2[1]

# P-09
streams_P09 = [22, 13, 14, 'GASOLINA']

exergy_P09 = []

for i in streams_P09:
    C =

```



```

    aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".format(i)).Value
    exergy_P09.append(C)

exergy_P09 = np.asarray(exergy_P09)

Exloss_P09 = exergy_P09[0]+exergy_P09[2]-exergy_P09[1]-exergy_P09[3]

#Coluna T-02
Exloss_T02 = np.sum(Exergy.exergia(aspen, 'T-02')[0])

# condensador P-10
streams_P10 = [3,28]

exergy_P10 = []

for i in streams_P10:
    C =
    aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".format(i)).Value
    exergy_P10.append(C)

Tc = aspen.Tree.FindNode("\\Data\\Blocks\\P-10\\Output\\B_TEMP").Value + 273.15
C = aspen.Tree.FindNode("\\Data\\Blocks\\P-10\\Output\\QCALC").Value * 0.0041668
if Tc > 298.15:
    Qc = C*(1-(298.15/Tc))
else:
    Qc = C*(1-(Tc/298.15))

exergy_P10.append(Qc)

exergy_P10 = np.asarray(exergy_P10)

Exloss_P10 = exergy_P10[0]-exergy_P10[1]+exergy_P10[2]

#print (exergy_P10,Tc,C, Exloss_P10)

# V-07
streams_V07 = [28,41] #33 corrente sem vazao

exergy_V07 = []

for i in streams_V07:

```

```

C =
    aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
    at(i)).Value
    exergy_V07.append(C)

exergy_V07 = np.asarray(exergy_V07)

Exloss_V07 = exergy_V07[0]-exergy_V07[1] #-exergy_V07[2]

# bomba B-08
streams_B08 = [43, 38]

exergy_B08 = []

for i in streams_B08:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
        exergy_B08.append(C)

C = aspen.Tree.FindNode("\\Data\\Blocks\\B8\\Output\\WNET").Value
exergy_B08.append(C)
exergy_B08 = np.asarray(exergy_B08)

Exloss_B08 = exergy_B08[0]-exergy_B08[1]+exergy_B08[2]

#Coluna T-03
Exloss_T03 = np.sum(Exergy.exergia(aspen,'T-03')[0])

# Vaso B3
streams_B3 = [39,50,44]

exergy_B3 = []

for i in streams_B3:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
        exergy_B3.append(C)

Tc = aspen.Tree.FindNode("\\Data\\Blocks\\B3\\Output\\B_TEMP").Value + 273.15
C = aspen.Tree.FindNode("\\Data\\Blocks\\B3\\Output\\QCALC").Value * 0.0041868
if Tc > 298.15:
    Qc = C*(1-(298.15/Tc))
else:
    Qc = C*(1-(Tc/298.15))

```

```

exergy_B3.append(Qc)
exergy_B3 = np.asarray(exergy_B3)

Exloss_B3 = exergy_B3[0]-exergy_B3[2]-exergy_B3[4]+exergy_B3[3]

#print (exergy_B3,Tc,C)

# condensador A-01
streams_A01 = ['20P','1P']

exergy_A01 = []

for i in streams_A01:
    C =
    aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL" .form
    at(i)).Value
    exergy_A01.append(C)

Tc = aspen.Tree.FindNode("\\Data\\Blocks\\A01\\Output\\B_TEMP").Value + 273.15
C = aspen.Tree.FindNode("\\Data\\Blocks\\A01\\Output\\QCALC").Value * 0.0041868
if Tc > 298.15:
    Qc = C*(1-(298.15/Tc))
else:
    Qc = C*(1-(Tc/298.15))
exergy_A01.append(Qc)

exergy_A01 = np.asarray(exergy_A01)

Exloss_A01 = exergy_A01[0]-exergy_A01[1]+exergy_A01[2]

#print (exergy_A01,Tc,C)

# Valvula V01P ciclo de refrigeracao
streams_V01P = ['5P', '6P']

exergy_V01P = []

for i in streams_V01P:
    C =
    aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL" .form
    at(i)).Value
    exergy_V01P.append(C)

exergy_V01P = np.asarray(exergy_V01P)

Exloss_V01P = exergy_V01P[0]-exergy_V01P[1]

```

```

# Vaso S02
streams_S02 = ['4P', '9P', '7P', '12P']

exergy_S02 = []

for i in streams_S02:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
at(i)).Value
        exergy_S02.append(C)

exergy_S02 = np.asarray(exergy_S02)

Exloss_S02 = exergy_S02[0]+exergy_S02[2]-exergy_S02[1]-exergy_S02[3]

# Valvula V02P ciclo de refrigeracao
streams_V02P = ['12P', '13P']

exergy_V02P = []

for i in streams_V02P:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
at(i)).Value
        exergy_V02P.append(C)

exergy_V02P = np.asarray(exergy_V02P)

Exloss_V02P = exergy_V02P[0]-exergy_V02P[1]

# vaso S03
streams_S03 = ['14P', '17P']

exergy_S03 = []

for i in streams_S03:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
at(i)).Value
        exergy_S03.append(C)

Tc = aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Output\\B_TEMP").Value + 273.15
C = aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Output\\QCALC").Value * 0.0041868
if Tc > 298.15:
    Qc = C*(1-(298.15/Tc))
else:
    Qc = C*(1-(Tc/298.15))
exergy_S03.append(Qc)

```

```

exergy_S03 = np.asarray(exergy_S03)

Exloss_S03 = exergy_S03[0]-exergy_S03[1]+exergy_S03[2]

#print (exergy_S03,Tc,C, Exloss_S03)

# compressor C-01P ciclo de refrigeracao
streams_C01P = ['17P', '18P']

exergy_C01P = []

for i in streams_C01P:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
    exergy_C01P.append(C)

C = aspen.Tree.FindNode("\\Data\\Blocks\\C01\\Output\\IND_POWER").Value
exergy_C01P.append(C)
exergy_C01P = np.asarray(exergy_C01P)

Exloss_C01P = exergy_C01P[0]-exergy_C01P[1]+exergy_C01P[2]

# compressor C-02P ciclo de refrigeracao
streams_C02P = ['19P', '20P']

exergy_C02P = []

for i in streams_C02P:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
    exergy_C02P.append(C)

C = aspen.Tree.FindNode("\\Data\\Blocks\\C02\\Output\\IND_POWER").Value
exergy_C02P.append(C)
exergy_C02P = np.asarray(exergy_C02P)

Exloss_C02P = exergy_C02P[0]-exergy_C02P[1]+exergy_C02P[2]

# trocadores para ajuste de temperatura das correntes de saidas da UFGN
streams_B11 = ['GAS-RES', 'GAS-RES2']
exergy_B11 = []
for i in streams_B11:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form

```

```

        at(i)).Value
        exergy_B11.append(C)
Tc = aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Output\\B_TEMP").Value + 273.15
C = aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Output\\QCALC").Value * 0.0041868
if Tc > 298.15:
    Qc = C*(1-(298.15/Tc))
else:
    Qc = C*(1-(Tc/298.15))
exergy_B11.append(Qc)
exergy_B11 = np.asarray(exergy_B11)
Exloss_B11 = exergy_B11[0]-exergy_B11[1]+exergy_B11[2]
#print (exergy_B11,Tc,C,Exloss_B11)

streams_B12 = ['GASOLINA','GAS-2']
exergy_B12 = []
for i in streams_B12:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
        exergy_B12.append(C)
Tc = aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Output\\B_TEMP").Value + 273.15
C = aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Output\\QCALC").Value * 0.0041868
if Tc > 298.15:
    Qc = C*(1-(298.15/Tc))
else:
    Qc = C*(1-(Tc/298.15))
exergy_B12.append(Qc)
exergy_B12 = np.asarray(exergy_B12)
Exloss_B12 = exergy_B12[0]-exergy_B12[1]+exergy_B12[2]
#print (exergy_B12,Tc,C,Exloss_B12)

streams_B13 = ['GLP','GLP2']
exergy_B13 = []
for i in streams_B13:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
        exergy_B13.append(C)
Tc = aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Output\\B_TEMP").Value + 273.15
C = aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Output\\QCALC").Value * 0.0041868
if Tc > 298.15:
    Qc = C*(1-(298.15/Tc))
else:
    Qc = C*(1-(Tc/298.15))
exergy_B13.append(Qc)
exergy_B13 = np.asarray(exergy_B13)
Exloss_B13 = exergy_B13[0]-exergy_B13[1]+exergy_B13[2]
#print (exergy_B13,Tc,C,Exloss_B13)

streams_B14 = ['GLP','GLP2']
exergy_B14 = []
for i in streams_B14:
    C =
        aspen.Tree.FindNode("\\Data\\Streams\\{0}\\Output\\STRM_UPP\\EXERGYFL\\MIXED\\TOTAL".form
        at(i)).Value
        exergy_B14.append(C)
Tc = aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Output\\B_TEMP").Value + 273.15
C = aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Output\\QCALC").Value * 0.0041868

```

```

if Tc > 298.15:
    Qc = C*(1-(298.15/Tc))
else:
    Qc = C*(1-(Tc/298.15))
exergy_B14.append(Qc)
exergy_B14 = np.asarray(exergy_B14)
Exloss_B14 = exergy_B14[0]-exergy_B14[1]+exergy_B14[2]
#print (exergy_B14,Tc,C,Exloss_B14)

#Exloss_P71, Exloss_P72 e Exloss_V06 foram retirados da soma abaixo porque
#seus valores de exergia estao considerados na exergia da coluna (esses equipamentos
#sao o condensador da coluna
Exloss_Tot = (Exloss_P01 + Exloss_P02 + Exloss_P03 + Exloss_P04 +
Exloss_V02 + Exloss_B4 + Exloss_P52 + Exloss_P51 + Exloss_V03 +
Exloss_B10 + Exloss_P06 + Exloss_V04 + Exloss_TE01 + Exloss_V05 +
Exloss_C01 + Exloss_C02 +
Exloss_T01 + Exloss_B2 + Exloss_P09 + Exloss_T02 + Exloss_P10 +
Exloss_V07 + Exloss_B08 + Exloss_T03 + Exloss_B3 + Exloss_A01 +
Exloss_V01P + Exloss_S02 + Exloss_V02P + Exloss_S03 + Exloss_C01P +
Exloss_C02P + Exloss_B11 + Exloss_B12 + Exloss_B13 + Exloss_B14)

return (Exloss_P01, Exloss_P02, Exloss_P03, Exloss_P04, Exloss_V02, Exloss_B4,
Exloss_P52, Exloss_P51, Exloss_V03, Exloss_B10, Exloss_P06, Exloss_V04,
Exloss_TE01, Exloss_V05, Exloss_P71, Exloss_P72, Exloss_C01, Exloss_C02,
Exloss_V06, Exloss_T01, Exloss_B2, Exloss_P09, Exloss_T02, Exloss_P10,
Exloss_V07, Exloss_B08, Exloss_T03, Exloss_B3, Exloss_A01, Exloss_V01P,
Exloss_S02, Exloss_V02P, Exloss_S03, Exloss_C01P, Exloss_C02P, Exloss_B11,
Exloss_B12, Exloss_B13, Exloss_B14, Exloss_Tot)

```

Script em Python (jupyter notebook) da otimização energética

```
In [ ]: run init
```

```
In [ ]: aspen
```

```
In [ ]: import CGCC
import Exergy
import aspen_error
import Ex_loss
import Energy

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import scipy.integrate
from IPython.display import Image

import random #pacote usado pelo pacote de otimizacao deap
from deap import algorithms, base, creator, tools
```

Caso Original

```
In [ ]: comp_list = []

for i in np.arange(0,1000,1):
    try:
        comp = aspen.Tree.FindNode("\\Data\\Components\\Comp-Lists\\GLOBAL\\Input\\CID\\#{0}".format(i)).Value
        comp_list.append(comp)

    except AttributeError:
        break

comp_list = np.asarray(comp_list) # Converte a Lista em vetor

#Output em molar frac e molar flow para os componentes da corrente de entrada
flow_1 = {}
flow_1['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\1\\Output\\MOLEFLOW\\MIXED\\#{0}".format(i)).Value
    flow_1['molar_flow'].append(F)

flow_1['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\1\\Output\\MOLEFRAC\\MIXED\\#{0}".format(i)).Value
    flow_1['molar_frac'].append(F)

flow_1 = pd.DataFrame.from_dict(flow_1, orient='columns', dtype=None)
flow_1.index = comp_list

flow_1
```



```

In [ ]: #Output em molar frac e molar flow para os componentes da corrente 16
flow_16 = {}
flow_16['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_16['molar_flow'].append(F)

flow_16['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_16['molar_frac'].append(F)

flow_16 = pd.DataFrame.from_dict(flow_16, orient='columns', dtype=None)
flow_16.index = comp_list

print (flow_16)

#Output em molar frac e molar flow para os componentes da corrente 21
flow_21 = {}
flow_21['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_21['molar_flow'].append(F)

flow_21['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_21['molar_frac'].append(F)

flow_21 = pd.DataFrame.from_dict(flow_21, orient='columns', dtype=None)
flow_21.index = comp_list

print (flow_21)

In [ ]: En_orig = Energy.Energy(aspen)[-1]
Ex_loss_orig = Ex_loss.Exloss(aspen)[-1]
str_16 = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MASSFLOW\\MIXED").Value
str_21 = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MASSFLOW\\MIXED").Value

print ('Energia Consumida - ',En_orig,' kW')
print ('Energia Perdida - ',Ex_loss_orig,' kW')
print ('str 16 ',str_16)
print ('str 21 ',str_21)

In [ ]: T_B11 = aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Output\\B_TEMP").Value
T_B12 = aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Output\\B_TEMP").Value
T_B13 = aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Output\\B_TEMP").Value
T_B14 = aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Output\\B_TEMP").Value
print (T_B11,T_B12,T_B13,T_B14)

In [ ]: aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Input\\SPEC_OPT").Value = 'TP'
aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Input\\TEMP").Value = T_B11
aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Input\\TEMP").Value = T_B12
aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Input\\TEMP").Value = T_B13
aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Input\\TEMP").Value = T_B14

In [ ]: # obtencao do polinomio para o calculo da vazao de C3 em funcao da pressao de C3 no trocador P-84 e P-82
Pressao_C3 = [1., 1.2, 1.3, 1.5, 1.7, 1.8, 2., 2.3, 2.5]
vazao_C3 = [55850., 55250., 54970., 54500., 54085., 53900., 53550., 53100., 52817.]
z = np.polyfit(Pressao_C3, vazao_C3, 5)# ,full = True)
vazao = np.poly1d(z)

Pressao_V01 = [3.5, 3.8, 4., 4.3, 4.5, 4.8, 5., 5.3, 5.5, 5.8, 6., 6.3, 6.5]
vazao_5P_1 = [26960., 26510., 26260., 25910., 25660., 25360., 25160., 24760., 24560., 24360., 24160., 23910., 23760]
x1 = np.polyfit(Pressao_V01, vazao_5P_1, 3)# ,full = True)
vazao_5P = np.poly1d(x1)

#polinomio para variacao de temperatura em 48
Temp_48 = [-11., -13., -15., -17., -19., -21., -23]
vazao_T = [-6800., -4500., -2200., 0., 2500., 5000., 7500.]
t_48 = np.polyfit(Temp_48, vazao_T, 3)# ,full = True)
vazao_t48 = np.poly1d(t_48)

```

```

In [ ]: dados = [aspen.Tree.FindNode("\\Data\\Results Summary\\Run-Status\\Output\\PER_ERROR").Value,
                aspen.Tree.FindNode("\\Data\\Blocks\\TE-01\\Input\\PRES").Value,
                aspen.Tree.FindNode("\\Data\\Blocks\\V-05\\Input\\PRES").Value,
                aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Input\\PRES").Value,
                aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES1").Value,
                aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Input\\PRES").Value,
                Energy.Energy(aspen)[-1], ((Energy.Energy(aspen)[-1]-En_orig)/En_orig)*100]

dados_2 = [1, 2, 3, 4, 5, 6, 7, 8]

Data = pd.DataFrame([pd.Series(dados_2), pd.Series(dados),
                    pd.Series(CGCC.stages_T(aspen, 'T-01')), pd.Series(CGCC.h_CGCC(aspen, 'T-01')[-1]),
                    pd.Series(Energy.exergia(aspen, 'T-01')[0]), pd.Series(np.arange(1,46,1))])

Data = Data.T
Data.columns = ['i', 'Dados', 'Stages_T', 'H_CGCC', 'Exergy', 'Stages']
Data = pd.DataFrame(Data)

fig, axes = plt.subplots(nrows=1, ncols=3)

a = pd.Series.as_matrix(Data['i'][:8])
c = pd.Series.as_matrix(Data['Dados'][:8])
clust_data = np.vstack((a, c))
clust_data = clust_data.T

collabel = ('i', 'Dados')
rowlabel=('erro', 'TE-01', 'V-05', 'V-06', 'T-01', 'S-03', 'energia', 'en %')
axes[0].axis('tight')
axes[0].axis('off')

the_table = axes[0].table(cellText=clust_data,
                          rowLabels=rowlabel,
                          colLabels=collabel,
                          colwidths=None,#[0.1,0.2,0.2],
                          loc='center left')#'.center'

the_table.set_fontsize(36)
the_table.scale(1., 2.)

Data.plot(x='H_CGCC', y = 'Stages_T', xlim = (0, 20), ax=axes[1], figsize=(24, 5))
Data.plot(x='Exergy', y = 'Stages_T', xlim = (0, 400.), ax =axes[2], figsize=(24, 5))

```

```

In [ ]: plt.plot(Energy.exergia(aspen, 'T-01')[0], np.arange(1,47,1))

```

```

In [ ]: fig, axes = plt.subplots(nrows=1, ncols=2)

Data = pd.DataFrame([pd.Series(CGCC.stages_T(aspen, 'T-01')), pd.Series(CGCC.h_CGCC(aspen, 'T-01')[-1]),
                    pd.Series(Energy.exergia(aspen, 'T-01')[0]), pd.Series(np.arange(1,47,1))])

Data = Data.T
Data.columns = ['Stages_T', 'H_CGCC', 'Exergy', 'Stages']
Data = pd.DataFrame(Data)

Data.plot(x='Exergy', y = 'Stages_T', xlim = (0, 400), ax=axes[0], figsize=(24, 5))
Data.plot(x='Exergy', y = 'Stages_T', xlim = (0, 400.), ax =axes[1], figsize=(24, 5))

```

Caso Otimizado - Composicao original

```
In [ ]: creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

FLT_MIN, FLT_MAX = 0.02, 0.98 #range da split B18
FLT_MIN2, FLT_MAX2 = 23.0, 30.0 #range de pressao do turbo expanson a coluna T01 (valor inicial TE-01 = 26.05477424)
FLT_MIN3, FLT_MAX3 = 1.0, 2.5 #range de pressao do propano no segundo trocador
FLT_MIN4, FLT_MAX4 = 3.5, 6.5 #range de pressao do propano no primeiro trocador
FLT_MIN5, FLT_MAX5 = -11., -23. #range de temperatura da corrente 48

N_CYCLES = 1

toolbox = base.Toolbox()

toolbox.register("attr_flt", random.uniform, FLT_MIN, FLT_MAX)
toolbox.register("attr_flt2", random.uniform, FLT_MIN2, FLT_MAX2)
toolbox.register("attr_flt3", random.uniform, FLT_MIN3, FLT_MAX3)
toolbox.register("attr_flt4", random.uniform, FLT_MIN4, FLT_MAX4)
toolbox.register("attr_flt5", random.uniform, FLT_MIN5, FLT_MAX5)
toolbox.register("aspen", aspen)

func_seq = [toolbox.attr_flt, toolbox.attr_flt2, toolbox.attr_flt3, toolbox.attr_flt4, toolbox.attr_flt5, toolbox.aspen]
print (func_seq)

In [ ]: toolbox.register("individual", tools.initCycle, creator.Individual, func_seq, n=N_CYCLES)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

pop = toolbox.population(n=10)
```



```

In [ ]: def evalOneMin(individual):

    #variaveis da otimizacao
    aspen.Tree.FindNode("\\Data\\Blocks\\B18\\Input\\FRAC\\9").Value = individual[0] #B18 (split corrente 9) = 0.8
    aspen.Tree.FindNode("\\Data\\Blocks\\TE-01\\Input\\PRES").Value = individual[1] #TE-01 = 26.05477424
    aspen.Tree.FindNode("\\Data\\Blocks\\V02\\Input\\P_OUT").Value = individual[2] #P_out = 1.5 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\V01\\Input\\P_OUT").Value = individual[3] #P_out = 6 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\P-04\\Input\\VALUE").Value = individual[4] #Temp original = -17

    # pressoes da V-05, T-01, V-06, S-03, valvula B10 e valvula B4
    aspen.Tree.FindNode("\\Data\\Blocks\\V-05\\Input\\PRES").Value = individual[1] #26.6469282
    aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Input\\PRES").Value = individual[1] - 0.5 #25.56131261
    aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES1").Value = individual[1] - 0.6 #25.48235875
    aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Input\\PRES").Value = individual[2] # original 1.5 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\B4\\Input\\P_OUT").Value = individual[1] #26.6469282
    aspen.Tree.FindNode("\\Data\\Blocks\\B10\\Input\\P_OUT").Value = individual[1] #26.6469282
    aspen.Tree.FindNode("\\Data\\Streams\\3P\\Input\\TOTFLOW\\MIXED").Value = vazao(individual[2])+vazao_t48(individual[4]
    aspen.Tree.FindNode("\\Data\\Blocks\\B7\\Input\\BASIS_FLOW\\5P").Value = vazao_5P(individual[3])
    aspen.Tree.FindNode("\\Data\\Blocks\\S02\\Input\\PRES").Value = individual[3] # original 6 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\C01\\Input\\PRES").Value = individual[3] # original 6 bar

    aspen.Engine.Run2()

    print (aspen.Tree.FindNode("\\Data\\Results Summary\\Run-Status\\Output\\PER_ERROR").Value)
    print ('TE-01', aspen.Tree.FindNode("\\Data\\Blocks\\TE-01\\Input\\PRES").Value)
    print ('split_B18', aspen.Tree.FindNode("\\Data\\Blocks\\B18\\Input\\FRAC\\9").Value)
    print ('pres_prop', aspen.Tree.FindNode("\\Data\\Blocks\\V02\\Input\\P_OUT").Value)
    print ('V-05', aspen.Tree.FindNode("\\Data\\Blocks\\V-05\\Input\\PRES").Value)
    print ('V-06', aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Input\\PRES").Value)
    print ('T-01', aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES1").Value)
    print ('S-03', aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Input\\PRES").Value)
    print ('S-02', aspen.Tree.FindNode("\\Data\\Blocks\\S02\\Input\\PRES").Value)
    print ('vazao_3P', aspen.Tree.FindNode("\\Data\\Streams\\3P\\Output\\MASSFLOW\\MIXED").Value)
    print ('vazao_5P', aspen.Tree.FindNode("\\Data\\Streams\\5P\\Output\\MASSFLOW\\MIXED").Value)
    print ('Temp 48', aspen.Tree.FindNode("\\Data\\Streams\\48\\Output\\TEMP_OUT\\MIXED").Value)
    print ('energia', Energy.Energy(aspen)[-1], ((Energy.Energy(aspen)[-1]-En_orig)/En_orig)*100)
    print ('exergia', Ex_loss.Exloss(aspen)[-1], ((Ex_loss.Exloss(aspen)[-1]-Exloss_orig)/Exloss_orig)*100)
    #print (En_orig)
    #print (ExLoss_orig)
    #print (plt.plot(Exergy.exergia(aspen, 'T-01')[0], np.arange(1,47,1)))

    print ()
    #exergy_loss = Ex_loss.Exloss(aspen)[-1] #exergia total kW
    energy_loss = Energy.Energy(aspen)[-1] #energia consumida total kW

    #return exergy_loss, #esta virgula no final eh essencial
    return energy_loss, #esta virgula no final eh essencial

In [ ]: toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.0) # passei de 0.05 para 0.0
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", evalOneMin)

In [ ]: ind1 = toolbox.individual()
print (ind1)
print ()
print (ind1.fitness.valid)

In [ ]: toolbox.individual()[1]

In [ ]: ind1.fitness.values = evalOneMin(ind1)
print (ind1.fitness.valid)
print (ind1.fitness)
print (ind1)

In [ ]: result = algorithms.eaSimple(pop, toolbox, cxpb=0.5, mutpb=0., ngen=30, verbose=False)
print('Current best fitness:', evalOneMin(tools.selBest(pop, k=1)[0]))
print (tools.selBest(pop, k=1)[0])

In [ ]: Ex_loss.Exloss(aspen)

In [ ]: Energy.Energy(aspen)

In [ ]: pd.Series(Exergy.exergia(aspen, 'T-01')[0])

```

```
In [ ]: print ('Energia Consumida - ', Energy.Energy(aspern)[-1], ' kW')
print ('Energia Perdida - ', Ex_loss.Exloss(aspern)[-1], ' kW')

print ('str 16 ', aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MASSFLMX\\MIXED").Value)
print ('str 21 ', aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MASSFLMX\\MIXED").Value)
```

```
In [ ]: #Output em molar frac e molar flow para os componentes da corrente 16
flow_16 = {}
flow_16['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_16['molar_flow'].append(F)

flow_16['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_16['molar_frac'].append(F)

flow_16 = pd.DataFrame.from_dict(flow_16, orient='columns', dtype=None)
flow_16.index = comp_list

flow_16
```

```
In [ ]: #Output em molar frac e molar flow para os componentes da corrente 21
flow_21 = {}
flow_21['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_21['molar_flow'].append(F)

flow_21['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_21['molar_frac'].append(F)

flow_21 = pd.DataFrame.from_dict(flow_21, orient='columns', dtype=None)
flow_21.index = comp_list

flow_21
```

```
In [ ]: T_B11 = aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Output\\B_TEMP").Value
T_B12 = aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Output\\B_TEMP").Value
T_B13 = aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Output\\B_TEMP").Value
T_B14 = aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Output\\B_TEMP").Value
print (T_B11, T_B12, T_B13, T_B14)
```

```
In [ ]: plt.plot(CGCC.h.CGCC(aspern, 'T-01')[-1], CGCC.stages_T(aspern, 'T-01'))
```

```
In [ ]: fig, axes = plt.subplots(nrows=1, ncols=2)

Data = pd.DataFrame([pd.Series(CGCC.stages_T(aspern, 'T-01')), pd.Series(CGCC.h.CGCC(aspern, 'T-01')[-1]),
                    pd.Series(Exergy.exergia(aspern, 'T-01')[0]), pd.Series(np.arange(1,47,1))])

Data = Data.T
Data.columns = ['Stages_T', 'H.CGCC', 'Exergy', 'Stages']
Data = pd.DataFrame(Data)

Data.plot(x='Exergy', y='Stages', xlim = (0, 400), ax=axes[0], figsize=(24, 5))
Data.plot(x='Exergy', y='Stages_T', xlim = (0, 400.), ax = axes[1], figsize=(24, 5))
```

Caso Otimizado - Composicao rica

```
In [ ]: aspen.Close()
```

```
In [ ]: run init
```

```
In [ ]: import CGCC
import Exergy
import aspen_error
import Ex_loss
import Energy

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import scipy.integrate
from IPython.display import Image

import random #pacote usado pelo pacote de otimizacao deap
from deap import algorithms, base, creator, tools
```

```
In [ ]: comp_list = []

for i in np.arange(0,1000,1):
    try:
        comp = aspen.Tree.FindNode("\\Data\\Components\\Comp-Lists\\GLOBAL\\Input\\CID\\#{0}".format(i)).Value
        comp_list.append(comp)

    except AttributeError:
        break

comp_list = np.asarray(comp_list) # Converte a lista em vetor

# Dados para confirmar que a simulacao retornou a condicao original
print ('Temp_1 ', aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\TEMP\\MIXED").Value)
print ('TE-01 ', aspen.Tree.FindNode("\\Data\\Blocks\\TE-01\\Input\\PRES").Value)
print ('split_B18 ', aspen.Tree.FindNode("\\Data\\Blocks\\B18\\Input\\FRAC\\9").Value)
print ('pres_prop ', aspen.Tree.FindNode("\\Data\\Blocks\\V02\\Input\\P_OUT").Value)
print ()
#Output em molar frac e molar flow para os componentes da corrente de entrada
flow_1 = {}
flow_1['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\I\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_1['molar_flow'].append(F)

flow_1['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\I\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_1['molar_frac'].append(F)

flow_1 = pd.DataFrame.from_dict(flow_1, orient='columns', dtype=None)
flow_1.index = comp_list

print (flow_1)
```

```
In [ ]: aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\N2").Value = 64.45
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\CO2").Value = 37.77
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\CH4").Value = 2390.0
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\ETHANE").Value = 500.
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\PROPANE").Value = 250.
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\ISOBU-01").Value = 70.
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\N-BUT-01").Value = 60.
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\2-MET-01").Value = 30.
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\N-PEN-01").Value = 30.
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\N-HEX-01").Value = 25.
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\N-HEP-01").Value = 6.58
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\N-OCT-01").Value = 1.04
aspen.Tree.FindNode("\\Data\\Streams\\I\\Input\\FLOW\\MIXED\\N-NON-01").Value = 0.35

aspen.Tree.FindNode("\\Data\\Blocks\\87\\Input\\BASIS_FLOW\\5P").Value = 40000
aspen.Tree.FindNode("\\Data\\Streams\\3P\\Input\\TOTFLOW\\MIXED").Value = 79000

aspen.Engine.Run2()
print (aspen.Tree.FindNode("\\Data\\Results Summary\\Run-Status\\Output\\PER_ERROR").Value)
```



```
In [ ]: #Output em molar frac e molar flow para os componentes da corrente de entrada
flow_1 = {}
flow_1['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\1\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_1['molar_flow'].append(F)

flow_1['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\1\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_1['molar_frac'].append(F)

flow_1 = pd.DataFrame.from_dict(flow_1, orient='columns', dtype=None)
flow_1.index = comp_list

flow_1
```

```
In [ ]: #Output em molar frac e molar flow para os componentes da corrente 16
flow_16 = {}
flow_16['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_16['molar_flow'].append(F)

flow_16['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_16['molar_frac'].append(F)

flow_16 = pd.DataFrame.from_dict(flow_16, orient='columns', dtype=None)
flow_16.index = comp_list

print (flow_16)
print ()

#Output em molar frac e molar flow para os componentes da corrente 21
flow_21 = {}
flow_21['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_21['molar_flow'].append(F)

flow_21['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_21['molar_frac'].append(F)

flow_21 = pd.DataFrame.from_dict(flow_21, orient='columns', dtype=None)
flow_21.index = comp_list

print (flow_21)
```

```
In [ ]: En_orig = Energy.Energy(aspern)[-1]
Ex_loss_orig = Ex_loss.Exloss(aspern)[-1]

print ('Energia Consumida - ',En_orig,' kW')
print ('Energia Perdida - ',Ex_loss_orig,' kW')
print ('str 16 ',aspern.Tree.FindNode("\\Data\\Streams\\16\\Output\\MASSFLM\\MIXED").Value)
print ('str 21 ',aspern.Tree.FindNode("\\Data\\Streams\\21\\Output\\MASSFLM\\MIXED").Value)
```

```
In [ ]: T_B11 = aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Output\\B_TEMP").Value
T_B12 = aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Output\\B_TEMP").Value
T_B13 = aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Output\\B_TEMP").Value
T_B14 = aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Output\\B_TEMP").Value
print (T_B11,T_B12,T_B13,T_B14)
```

```
In [ ]: aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Input\\SPEC_OPT").Value = 'TP'
aspern.Tree.FindNode("\\Data\\Blocks\\B11\\Input\\TEMP").Value = T_B11
aspern.Tree.FindNode("\\Data\\Blocks\\B12\\Input\\TEMP").Value = T_B12
aspern.Tree.FindNode("\\Data\\Blocks\\B13\\Input\\TEMP").Value = T_B13
aspern.Tree.FindNode("\\Data\\Blocks\\B14\\Input\\TEMP").Value = T_B14
```

```

In [ ]: fig, axes = plt.subplots(nrows=1, ncols=2)

Data = pd.DataFrame([pd.Series(CGCC.stages_T(aspens, 'T-01')), pd.Series(CGCC.h_CGCC(aspens, 'T-01')[-1]),
                    pd.Series(Exergy.exergia(aspens, 'T-01')[0]), pd.Series(np.arange(1,47,1))])

Data = Data.T
Data.columns = ['Stages_T', 'H_CGCC', 'Exergy', 'Stages']
Data = pd.DataFrame(Data)

Data.plot(x='Exergy', y = 'Stages', xlim = (0, 400), ax=axes[0], figsize=(24, 5))
Data.plot(x='Exergy', y = 'Stages_T', xlim = (0, 400.), ax =axes[1],figsize=(24, 5))

In [ ]: # obtencao do polinomio para o calculo da vazao de C3 em funcao da pressao do C3 no trocador P-04 e P-02
Pressao_C3_rico = [1., 1.2, 1.3, 1.5, 1.7, 1.8, 2., 2.3, 2.5]
vazao_C3_rico = [81000., 80100., 79800., 79000., 78500., 78300., 77800., 77000., 76650.]
y = np.polyfit(Pressao_C3_rico, vazao_C3_rico, 3)# ,full = True)
vazao_rico = np.poly1d(y)

Pressao_V01_rico = [3.5, 3.8, 4., 4.3, 4.5, 4.8, 5., 5.3, 5.5, 5.8, 6., 6.3, 6.5]
vazao_SP_1_rico = [44900., 44100., 43700., 42900., 42500., 42000., 41600., 41100., 40850., 40450., 40000., 39600., 39300.]
x2 = np.polyfit(Pressao_V01_rico, vazao_SP_1_rico, 3)# ,full = True)
vazao_SP_1_rico = np.poly1d(x2)

#polinomio para variacao de temperatura em 48
Temp_48_rico = [-11., -13., -15., -17., -19., -21., -23]
vazao_T_rico = [-8500., -5700., -3000., 0., 2900., 6000., 9000.]
t_48_rico = np.polyfit(Temp_48_rico, vazao_T_rico, 3)# ,full = True)
vazao_t48_rico = np.poly1d(t_48_rico)

In [ ]: creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

FLT_MIN, FLT_MAX = 0.02, 0.98 #range do split B18
FLT_MIN2, FLT_MAX2 = 23.0, 30.0 #range de pressao do turbo expensor a coluna T01 (valor inicial TE-01 = 26.05477424)
FLT_MIN3, FLT_MAX3 = 1.0, 2.5 #range de pressao do propano no segundo trocador
FLT_MIN4, FLT_MAX4 = 3.5, 6.5 #range de pressao do propano no primeiro trocador
FLT_MIN5, FLT_MAX5 = -11., -23. #range de temperatura da corrente 48

N_CYCLES = 1

toolbox = base.Toolbox()

toolbox.register("attr_flt", random.uniform, FLT_MIN, FLT_MAX)
toolbox.register("attr_flt2", random.uniform, FLT_MIN2, FLT_MAX2)
toolbox.register("attr_flt3", random.uniform, FLT_MIN3, FLT_MAX3)
toolbox.register("attr_flt4", random.uniform, FLT_MIN4, FLT_MAX4)
toolbox.register("attr_flt5", random.uniform, FLT_MIN5, FLT_MAX5)
toolbox.register("aspens", aspens)

func_seq = [toolbox.attr_flt, toolbox.attr_flt2, toolbox.attr_flt3, toolbox.attr_flt4, toolbox.attr_flt5, toolbox.aspens]

print (func_seq)

In [ ]: toolbox.register("individual", tools.initCycle, creator.Individual, func_seq, n=N_CYCLES)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

pop = toolbox.population(n=10)

```



```

In [ ]: def evalOneMin(individual):

    #variaveis da otimizacao
    aspen.Tree.FindNode("\\Data\\Blocks\\B18\\Input\\FRAC\\9").Value = individual[0] #B18 (split corrente 9) = 0.8
    aspen.Tree.FindNode("\\Data\\Blocks\\TE-01\\Input\\PRES").Value = individual[1] #TE-01 = 26.05477424
    aspen.Tree.FindNode("\\Data\\Blocks\\V02\\Input\\P_OUT").Value = individual[2] #P_out = 1.5 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\V02\\Input\\P_OUT").Value = individual[3] #P_out = 6 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\P-04\\Input\\VALUE").Value = individual[4] #Temp original = -17

    # pressoes da V-05, T-01, V-06, S-03, valvula B10 e valvula B4
    aspen.Tree.FindNode("\\Data\\Blocks\\V-05\\Input\\PRES").Value = individual[1] #26.6469282
    aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Input\\PRES").Value = individual[1] - 0.5 #25.56131261
    aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES1").Value = individual[1] - 0.6 #25.48235875
    aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Input\\PRES").Value = individual[2] # original 1.5 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\B4\\Input\\P_OUT").Value = individual[1] #26.6469282
    aspen.Tree.FindNode("\\Data\\Blocks\\B10\\Input\\P_OUT").Value = individual[1] #26.6469282
    aspen.Tree.FindNode("\\Data\\Streams\\3P\\Input\\TOTFLOW\\MIXED").Value = vazao_rico(individual[2])+vazao_t48_rico(individual[1])
    aspen.Tree.FindNode("\\Data\\Blocks\\B7\\Input\\BASIS_FLOW\\5P").Value = vazao_5P_rico(individual[3])
    aspen.Tree.FindNode("\\Data\\Blocks\\S02\\Input\\PRES").Value = individual[3] # original 6 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\C02\\Input\\PRES").Value = individual[3] # original 6 bar

    aspen.Engine.Run2()

    print (aspen.Tree.FindNode("\\Data\\Results Summary\\Run-Status\\Output\\PER_ERROR").Value)
    print ('TE-01 ', aspen.Tree.FindNode("\\Data\\Blocks\\TE-01\\Input\\PRES").Value)
    print ('split B18 ', aspen.Tree.FindNode("\\Data\\Blocks\\B18\\Input\\FRAC\\9").Value)
    print ('pres_prop ', aspen.Tree.FindNode("\\Data\\Blocks\\V02\\Input\\P_OUT").Value)
    print ('V-05 ', aspen.Tree.FindNode("\\Data\\Blocks\\V-05\\Input\\PRES").Value)
    print ('V-06 ', aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Input\\PRES").Value)
    print ('T-01 ', aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES1").Value)
    print ('S-03 ', aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Input\\PRES").Value)
    print ('S-02 ', aspen.Tree.FindNode("\\Data\\Blocks\\S02\\Input\\PRES").Value)
    print ('vazao_3P ', aspen.Tree.FindNode("\\Data\\Streams\\3P\\Output\\MASSFLM\\MIXED").Value)
    print ('vazao_5P ', aspen.Tree.FindNode("\\Data\\Streams\\5P\\Output\\MASSFLM\\MIXED").Value)
    print ('Temp 48 ', aspen.Tree.FindNode("\\Data\\Streams\\48\\Output\\TEMP_OUT\\MIXED").Value)
    print ('energia ', Energy.Energy(aspen)[-1], ((Energy.Energy(aspen)[-1]-En_orig)/En_orig)*100)
    print ('exergia ', Ex_loss.Exloss(aspen)[-1], ((Ex_loss.Exloss(aspen)[-1]-Ex_loss_orig)/Ex_loss_orig)*100)
    #print (En_orig)
    #print (ExLoss_orig)
    #print (plt.plot(Energy.exergia(aspen, 'T-01')[0], np.arange(1,47,1)))

    print ()
    #exergy_loss = Ex_loss.Exloss(aspen)[-1] #exergia total kW
    energy_loss = Energy.Energy(aspen)[-1] #energia consumida total kW

    #return exergy_loss, #esta virgula no final eh essencial
    return energy_loss, #esta virgula no final eh essencial

```

```

In [ ]: toolbox.register("mate", tools.cxTwoPoint)
        toolbox.register("mutate", tools.mutFlipBit, indpb=0.0) # passei de 0.05 para 0.0
        toolbox.register("select", tools.selTournament, tournsize=3)
        toolbox.register("evaluate", evalOneMin)

```

```

In [ ]: ind1 = toolbox.individual()
        print (ind1)
        print ()
        print (ind1.fitness.valid)

```

```

In [ ]: ind1.fitness.values = evalOneMin(ind1)
        print (ind1.fitness.valid)
        print (ind1.fitness)
        print (ind1)

```

```

In [ ]: result = algorithms.eaSimple(pop, toolbox, cxpb=0.5, mutpb=0., ngen=30, verbose=False)
        print('Current best fitness:', evalOneMin(tools.selBest(pop, k=1)[0]))
        print (tools.selBest(pop, k=1)[0])

```

```

In [ ]: print ('Energia Consumida - ', Energy.Energy(aspen)[-1], ' kW')
        print ('Exergia Perdida - ', Ex_loss.Exloss(aspen)[-1], ' kW')
        print ('str 16 ', aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MASSFLM\\MIXED").Value)
        print ('str 21 ', aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MASSFLM\\MIXED").Value)

```

```

In [ ]: Ex_loss.Exloss(aspen)

```

```
In [ ]: #Output em molar frac e molar flow para os componentes da corrente 16
flow_16 = {}
flow_16['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_16['molar_flow'].append(F)

flow_16['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_16['molar_frac'].append(F)

flow_16 = pd.DataFrame.from_dict(flow_16, orient='columns', dtype=None)
flow_16.index = comp_list

flow_16
```

```
In [ ]: #Output em molar frac e molar flow para os componentes da corrente 21
flow_21 = {}
flow_21['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_21['molar_flow'].append(F)

flow_21['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_21['molar_frac'].append(F)

flow_21 = pd.DataFrame.from_dict(flow_21, orient='columns', dtype=None)
flow_21.index = comp_list

flow_21
```

```
In [ ]: T_B11 = aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Output\\B_TEMP").Value
T_B12 = aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Output\\B_TEMP").Value
T_B13 = aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Output\\B_TEMP").Value
T_B14 = aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Output\\B_TEMP").Value
print (T_B11,T_B12,T_B13,T_B14)
```

```
In [ ]: Energy.Energy(aspern)
```

```
In [ ]: plt.plot(CGCC.h.CGCC(aspern,'T-01')[-1],CGCC.stages_T(aspern,'T-01'))
```

```
In [ ]: fig, axes = plt.subplots(nrows=1, ncols=2)

Data = pd.DataFrame([pd.Series(CGCC.stages_T(aspern,'T-01')), pd.Series(CGCC.h.CGCC(aspern,'T-01')[-1]),
                    pd.Series(Exergy.exergia(aspern,'T-01')[0]), pd.Series(np.arange(1,47,1))])

Data = Data.T
Data.columns = ['Stages_T','H.CGCC', 'Exergy', 'Stages']
Data = pd.DataFrame(Data)

Data.plot(x='Exergy', y = 'Stages',xlim = (0, 400), ax=axes[0],figsize=(24, 5))
Data.plot(x='Exergy', y = 'Stages_T',xlim = (0, 400.), ax =axes[1],figsize=(24, 5))
```

```
In [ ]: Exergy.exergia(aspern,'T-01')[0]
```

Script em Python (jupyter notebook) da otimização exergética

```
In [ ]: run init
```

```
In [ ]: aspen
```

```
In [ ]: import CGCC
import Exergy
import aspen_error
import Ex_loss
import Energy

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import scipy.integrate
from IPython.display import Image

import random #pacote usado pelo pacote de otimizacao deap
from deap import algorithms, base, creator, tools
```

Caso Original

```
In [ ]: comp_list = []

for i in np.arange(0,1000,1):
    try:
        comp = aspen.Tree.FindNode("\\Data\\Components\\Comp-Lists\\GLOBAL\\Input\\CID\\#{0}".format(i)).Value
        comp_list.append(comp)

    except AttributeError:
        break

comp_list = np.asarray(comp_list) # Converte a lista em vetor

#Output em molar frac e molar flow para os componentes da corrente de entrada
flow_1 = {}
flow_1['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\1\\Output\\MOLEFLOW\\MIXED\\#{0}".format(i)).Value
    flow_1['molar_flow'].append(F)

flow_1['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\1\\Output\\MOLEFRAC\\MIXED\\#{0}".format(i)).Value
    flow_1['molar_frac'].append(F)

flow_1 = pd.DataFrame.from_dict(flow_1, orient='columns', dtype=None)
flow_1.index = comp_list

flow_1
```

```

In [ ]: #Output em molar frac e molar flow para os componentes da corrente 16
flow_16 = {}
flow_16['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_16['molar_flow'].append(F)

flow_16['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_16['molar_frac'].append(F)

flow_16 = pd.DataFrame.from_dict(flow_16, orient='columns', dtype=None)
flow_16.index = comp_list

print (flow_16)

#Output em molar frac e molar flow para os componentes da corrente 21
flow_21 = {}
flow_21['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_21['molar_flow'].append(F)

flow_21['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_21['molar_frac'].append(F)

flow_21 = pd.DataFrame.from_dict(flow_21, orient='columns', dtype=None)
flow_21.index = comp_list

print (flow_21)

In [ ]: En_orig = Energy.Energy(aspern)[-1]
Exloss_orig = Ex_loss.Exloss(aspern)[-1]
str_16 = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MASSFLMX\\MIXED").Value
str_21 = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MASSFLMX\\MIXED").Value

print ('Energia Consumida - ',En_orig,' kw')
print ('Exergia Perdida - ',Exloss_orig,' kw')
print ('str 16 ',str_16)
print ('str 21 ',str_21)

In [ ]: T_B11 = aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Output\\B_TEMP").Value
T_B12 = aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Output\\B_TEMP").Value
T_B13 = aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Output\\B_TEMP").Value
T_B14 = aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Output\\B_TEMP").Value
print (T_B11,T_B12,T_B13,T_B14)

In [ ]: aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Input\\SPEC_OPT").Value = 'TP'
aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Input\\TEMP").Value = T_B11
aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Input\\TEMP").Value = T_B12
aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Input\\TEMP").Value = T_B13
aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Input\\TEMP").Value = T_B14

In [ ]: # obtencao do polinomio para o calculo da vazao de C3 em funcao da pressao do C3 no trocador P-04 e P-02
Pressao_C3 = [1., 1.2, 1.3, 1.5, 1.7, 1.8, 2., 2.3, 2.5]
vazao_C3 = [55850., 55250., 54970., 54500., 54085., 53900., 53550., 53100., 52817.]
z = np.polyfit(Pressao_C3, vazao_C3, 5)# ,full = True)
vazao = np.poly1d(z)

Pressao_V01 = [3.5, 3.8, 4., 4.3, 4.5, 4.8, 5., 5.3, 5.5, 5.8, 6., 6.3, 6.5]
vazao_SP_1 = [26960., 26510., 26260., 25910., 25660., 25360., 25160., 24760., 24560., 24360., 24160., 23910., 23760]
x1 = np.polyfit(Pressao_V01, vazao_SP_1, 3)# ,full = True)
vazao_SP = np.poly1d(x1)

#polinomio para variacao de temperatura em 48
Temp_48 = [-11., -13., -15., -17., -19., -21., -23]
vazao_T = [-6800., -4500., -2200., 0., 2500., 5000., 7500.]
t_48 = np.polyfit(Temp_48, vazao_T, 3)# ,full = True)
vazao_t48 = np.poly1d(t_48)

```



```

In [ ]: dados = [aspen.Tree.FindNode("\\Data\\Results Summary\\Run-Status\\Output\\PER_ERROR").Value,
                aspen.Tree.FindNode("\\Data\\Blocks\\TE-01\\Input\\PRES").Value,
                aspen.Tree.FindNode("\\Data\\Blocks\\V-05\\Input\\PRES").Value,
                aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Input\\PRES").Value,
                aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES1").Value,
                aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Input\\PRES").Value,
                Energy.Energy(aspen)[-1], ((Energy.Energy(aspen)[-1]-En_orig)/En_orig)*100]

dados_2 = [1, 2, 3, 4, 5, 6, 7, 8]

Data = pd.DataFrame([pd.Series(dados_2), pd.Series(dados),
                    pd.Series(CGCC.stages_T(aspen, 'T-01')), pd.Series(CGCC.h_CGCC(aspen, 'T-01')[-1]),
                    pd.Series(Exergy.exergia(aspen, 'T-01')[0]), pd.Series(np.arange(1,46,1))])

Data = Data.T
Data.columns = ['i', 'Dados', 'Stages_T', 'H_CGCC', 'Exergy', 'Stages']
Data = pd.DataFrame(Data)

fig, axes = plt.subplots(nrows=1, ncols=3)

a = pd.Series.as_matrix(Data['i'][:8])
c = pd.Series.as_matrix(Data['Dados'][:8])
clust_data = np.vstack((a, c))
clust_data = clust_data.T

collabel = ('i', 'Dados')
rowlabel = ('erro', 'TE-01', 'V-05', 'V-06', 'T-01', 'S-03', 'energia', 'en X')
axes[0].axis('tight')
axes[0].axis('off')

the_table = axes[0].table(cellText=clust_data,
                        rowLabels=rowlabel,
                        colLabels=collabel,
                        colwidths=None, # [0.1, 0.2, 0.2],
                        loc='center left')# 'center'

the_table.set_fontsize(36)
the_table.scale(1., 2.)

Data.plot(x='H_CGCC', y = 'Stages_T', xlim = (0, 20), ax=axes[1], figsize=(24, 5))
Data.plot(x='Exergy', y = 'Stages_T', xlim = (0, 400.), ax =axes[2], figsize=(24, 5))

```

```

In [ ]: plt.plot(Exergy.exergia(aspen, 'T-01')[0], np.arange(1,47,1))

```

```

In [ ]: fig, axes = plt.subplots(nrows=1, ncols=2)

Data = pd.DataFrame([pd.Series(CGCC.stages_T(aspen, 'T-01')), pd.Series(CGCC.h_CGCC(aspen, 'T-01')[-1]),
                    pd.Series(Exergy.exergia(aspen, 'T-01')[0]), pd.Series(np.arange(1,47,1))])

Data = Data.T
Data.columns = ['Stages_T', 'H_CGCC', 'Exergy', 'Stages']
Data = pd.DataFrame(Data)

Data.plot(x='Exergy', y = 'Stages', xlim = (0, 400), ax=axes[0], figsize=(24, 5))
Data.plot(x='Exergy', y = 'Stages_T', xlim = (0, 400.), ax =axes[1], figsize=(24, 5))

```

Caso Otimizado - Composicao original

```
In [ ]: creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

FLT_MIN, FLT_MAX = 0.02, 0.98 #range do split B18
FLT_MIN2, FLT_MAX2 = 23.0, 30.0 #range de pressao do turbo expansor a coluna T01 (valor inicial TE-01 = 26.05477424)
FLT_MIN3, FLT_MAX3 = 1.0, 2.5 #range de pressao do propano no segundo trocador
FLT_MIN4, FLT_MAX4 = 3.5, 6.5 #range de pressao do propano no primeiro trocador
FLT_MIN5, FLT_MAX5 = -11., -23. #range de temperatura da corrente 48

N_CYCLES = 1

toolbox = base.Toolbox()

toolbox.register("attr_float", random.uniform, FLT_MIN, FLT_MAX)
toolbox.register("attr_float2", random.uniform, FLT_MIN2, FLT_MAX2)
toolbox.register("attr_float3", random.uniform, FLT_MIN3, FLT_MAX3)
toolbox.register("attr_float4", random.uniform, FLT_MIN4, FLT_MAX4)
toolbox.register("attr_float5", random.uniform, FLT_MIN5, FLT_MAX5)
toolbox.register("aspen", aspen)

func_seq = [toolbox.attr_float, toolbox.attr_float2, toolbox.attr_float3, toolbox.attr_float4, toolbox.attr_float5, toolbox.aspen]

print (func_seq)

In [ ]: toolbox.register("individual", tools.initCycle, creator.Individual, func_seq, n=N_CYCLES)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

pop = toolbox.population(n=10)

In [ ]: def evaloneMin(individual):

#variaveis da otimizacao
aspen.Tree.FindNode("\\Data\\Blocks\\B18\\Input\\FRAC\\9").Value = individual[0] #B18 (split corrente 9) = 0.8
aspen.Tree.FindNode("\\Data\\Blocks\\TE-01\\Input\\PRES").Value = individual[1] #TE-01 = 26.05477424
aspen.Tree.FindNode("\\Data\\Blocks\\V02\\Input\\P_OUT").Value = individual[2] #P_out = 1.5 bar
aspen.Tree.FindNode("\\Data\\Blocks\\V01\\Input\\P_OUT").Value = individual[3] #P_out = 6 bar
aspen.Tree.FindNode("\\Data\\Blocks\\P-04\\Input\\VALUE").Value = individual[4] #Temp original = -17

# pressoes do V-05, T-01, V-06, S-03, valvula B10 e valvula B4
aspen.Tree.FindNode("\\Data\\Blocks\\V-05\\Input\\PRES").Value = individual[1] #26.6469282
aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Input\\PRES").Value = individual[1] - 0.5 #25.56131261
aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES1").Value = individual[1] - 0.6 #25.48235875
aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Input\\PRES").Value = individual[2] # original 1.5 bar
aspen.Tree.FindNode("\\Data\\Blocks\\B4\\Input\\P_OUT").Value = individual[1] #26.6469282
aspen.Tree.FindNode("\\Data\\Blocks\\B10\\Input\\P_OUT").Value = individual[1] #26.6469282
aspen.Tree.FindNode("\\Data\\Streams\\3P\\Input\\TOTFLOW\\MIXED").Value = vazao(individual[2])+vazao_t48(individual[4])
aspen.Tree.FindNode("\\Data\\Blocks\\B7\\Input\\BASIS_FLOW\\5P").Value = vazao_5P(individual[3])
aspen.Tree.FindNode("\\Data\\Blocks\\S02\\Input\\PRES").Value = individual[3] # original 6 bar
aspen.Tree.FindNode("\\Data\\Blocks\\C01\\Input\\PRES").Value = individual[3] # original 6 bar

aspen.Engine.Run2()

print (aspen.Tree.FindNode("\\Data\\Results Summary\\Run-Status\\Output\\PER_ERROR").Value)
print ('TE-01 ', aspen.Tree.FindNode("\\Data\\Blocks\\TE-01\\Input\\PRES").Value)
print ('split_B18 ', aspen.Tree.FindNode("\\Data\\Blocks\\B18\\Input\\FRAC\\9").Value)
print ('pres_prop ', aspen.Tree.FindNode("\\Data\\Blocks\\V02\\Input\\P_OUT").Value )
print ('V-05 ', aspen.Tree.FindNode("\\Data\\Blocks\\V-05\\Input\\PRES").Value)
print ('V-06 ', aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Input\\PRES").Value)
print ('T-01 ', aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES1").Value)
print ('S-03 ', aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Input\\PRES").Value)
print ('S-02 ', aspen.Tree.FindNode("\\Data\\Blocks\\S02\\Input\\PRES").Value)
print ('vazao_3P ', aspen.Tree.FindNode("\\Data\\Streams\\3P\\Output\\MASSFLOW\\MIXED").Value)
print ('vazao_5P ', aspen.Tree.FindNode("\\Data\\Streams\\5P\\Output\\MASSFLOW\\MIXED").Value)
print ('Temp 48 ', aspen.Tree.FindNode("\\Data\\Streams\\48\\Output\\TEMP_OUT\\MIXED").Value)
print ('energia ', Energy.Energy(aspen)[-1], ((Energy.Energy(aspen)[-1]-En_orig)/En_orig)*100)
print ('exergia ', Ex_loss.Exloss(aspen)[-1], ((Ex_loss.Exloss(aspen)[-1]-Exloss_orig)/Exloss_orig)*100)
#print (En_orig)
#print (Exloss_orig)
#print (plt.plot(Energy.exergia(aspen, 'T-01')[0], np.arange(1,47,1)))

print ()
energy_loss = Ex_loss.Exloss(aspen)[-1] #exergia total kW
energy_consumed = Energy.Energy(aspen)[-1] #energia consumida total kW

return energy_loss, #esta virgula no final eh essencial
#return energy_loss, #esta virgula no final eh essencial
```

```

In [ ]: toolbox.register("mate", tools.cxTwoPoint)
        toolbox.register("mutate", tools.mutFlipBit, indpb=0.0) # passo de 0.05 para 0.0
        toolbox.register("select", tools.selTournament, tournsize=3)
        toolbox.register("evaluate", evalOneMin)

In [ ]: ind1 = toolbox.individual()
        print (ind1)
        print ()
        print (ind1.fitness.valid)

In [ ]: toolbox.individual()[1]

In [ ]: ind1.fitness.values = evalOneMin(ind1)
        print (ind1.fitness.valid)
        print (ind1.fitness)
        print (ind1)

In [ ]: result = algorithms.eaSimple(pop, toolbox, cxpb=0.5, mutpb=0., ngen=10, verbose=False)
        print('Current best fitness:', evalOneMin(tools.selBest(pop, k=1)[0]))
        print (tools.selBest(pop, k=1)[0])

In [ ]: Ex_loss.Exloss(aspens)

In [ ]: print ('Energia Consumida - ',Energy.Energy(aspens)[-1], ' kW')
        print ('Exergia Perdida - ',Ex_loss.Exloss(aspens)[-1], ' kW')

        print ('str 16 ',aspens.Tree.FindNode("\\Data\\Streams\\16\\Output\\MASSFLMX\\MIXED").Value)
        print ('str 21 ',aspens.Tree.FindNode("\\Data\\Streams\\21\\Output\\MASSFLMX\\MIXED").Value)

In [ ]: #Output em molar frac e molar flow para os componentes da corrente 16
        flow_16 = {}
        flow_16['molar_flow'] = []
        for i in comp_list:
            F = aspens.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
            flow_16['molar_flow'].append(F)

        flow_16['molar_frac'] = []
        for i in comp_list:
            F = aspens.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
            flow_16['molar_frac'].append(F)

        flow_16 = pd.DataFrame.from_dict(flow_16, orient='columns', dtype=None)
        flow_16.index = comp_list

        flow_16

In [ ]: #Output em molar frac e molar flow para os componentes da corrente 21
        flow_21 = {}
        flow_21['molar_flow'] = []
        for i in comp_list:
            F = aspens.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
            flow_21['molar_flow'].append(F)

        flow_21['molar_frac'] = []
        for i in comp_list:
            F = aspens.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
            flow_21['molar_frac'].append(F)

        flow_21 = pd.DataFrame.from_dict(flow_21, orient='columns', dtype=None)
        flow_21.index = comp_list

        flow_21

In [ ]: T_B11 = aspens.Tree.FindNode("\\Data\\Blocks\\B11\\Output\\B_TEMP").Value
        T_B12 = aspens.Tree.FindNode("\\Data\\Blocks\\B12\\Output\\B_TEMP").Value
        T_B13 = aspens.Tree.FindNode("\\Data\\Blocks\\B13\\Output\\B_TEMP").Value
        T_B14 = aspens.Tree.FindNode("\\Data\\Blocks\\B14\\Output\\B_TEMP").Value
        print (T_B11,T_B12,T_B13,T_B14)

In [ ]: plt.plot(CGCC.h_CGCC(aspens, 'T-01')[-1],CGCC.stages_T(aspens, 'T-01'))

```



```
In [ ]: fig, axes = plt.subplots(nrows=1, ncols=2)

Data = pd.DataFrame([pd.Series(CGCC.stages_T( Aspen, 'T-01' )), pd.Series(CGCC.h_CGCC( Aspen, 'T-01' )[-1]),
                    pd.Series(Exergy.exergia( Aspen, 'T-01' )[0]), pd.Series(np.arange(1,47,1))])

Data = Data.T
Data.columns = ['Stages_T', 'H_CGCC', 'Exergy', 'Stages']
Data = pd.DataFrame(Data)

Data.plot(x='Exergy', y = 'Stages',xlin = (0, 400), ax=axes[0],figsize=(24, 5))
Data.plot(x='Exergy', y = 'Stages_T',xlin = (0, 400.), ax =axes[1],figsize=(24, 5))

In [ ]: Energy.Energy( Aspen)

In [ ]: pd.Series(Exergy.exergia( Aspen, 'T-01' )[0])
```

Caso Otimizado - Composicao rica

```
In [ ]: Aspen.Close()

In [ ]: run init

In [ ]: import CGCC
import Exergy
import Aspen_error
import Ex_loss
import Energy

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import scipy.integrate
from IPython.display import Image

import random #pacote usado pelo pacote de otimizacao deap
from deap import algorithms, base, creator, tools

In [ ]: comp_list = []

for i in np.arange(0,1000,1):
    try:
        comp = Aspen.Tree.FindNode("\\Data\\Components\\Comp-Lists\\GLOBAL\\Input\\CID\\#{0}".format(i)).Value
        comp_list.append(comp)

    except AttributeError:
        break

comp_list = np.asarray(comp_list) # Converte a lista em vetor

# Dados para confirmar que a simulacao retornou a condicao original
print ('Temp_1 ',Aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\TEMP\\MIXED").Value)
print ('TE-01 ',Aspen.Tree.FindNode("\\Data\\Blocks\\TE-01\\Input\\PRES").Value)
print ('split_B18 ', Aspen.Tree.FindNode("\\Data\\Blocks\\B18\\Input\\FRAC\\9").Value)
print ('pres_prop ', Aspen.Tree.FindNode("\\Data\\Blocks\\W02\\Input\\P_OUT").Value)
print ()

#Output em molar frac e molar flow para os componentes da corrente de entrada
flow_1 = {}
flow_1['molar_flow'] = []
for i in comp_list:
    F = Aspen.Tree.FindNode("\\Data\\Streams\\1\\Output\\MOLEFLOW\\MIXED\\#{0}".format(i)).Value
    flow_1['molar_flow'].append(F)

flow_1['molar_frac'] = []
for i in comp_list:
    F = Aspen.Tree.FindNode("\\Data\\Streams\\1\\Output\\MOLEFRAC\\MIXED\\#{0}".format(i)).Value
    flow_1['molar_frac'].append(F)

flow_1 = pd.DataFrame.from_dict(flow_1, orient='columns', dtype=None)
flow_1.index = comp_list

print (flow_1)
```



```

In [ ]: aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\N2").Value = 64.45
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\CO2").Value = 37.77
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\CH4").Value = 2390.0
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\ETHANE").Value = 500.
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\PROPANE").Value = 250.
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\ISOBU-01").Value = 70.
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\N-BUT-01").Value = 60.
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\2-MET-01").Value = 30.
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\N-PEN-01").Value = 30.
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\N-HEX-01").Value = 25.
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\N-HEP-01").Value = 6.58
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\N-OCT-01").Value = 1.04
aspen.Tree.FindNode("\\Data\\Streams\\1\\Input\\FLOW\\MIXED\\N-NOV-01").Value = 0.35

aspen.Tree.FindNode("\\Data\\Blocks\\87\\Input\\BASIS_FLOW\\5P").Value = 40000
aspen.Tree.FindNode("\\Data\\Streams\\3P\\Input\\TOTFLOW\\MIXED").Value = 79000

aspen.Engine.Run2()
print (aspen.Tree.FindNode("\\Data\\Results Summary\\Run-Status\\Output\\PER_ERROR").Value)

In [ ]: #Output em molar frac e molar flow para os componentes da corrente de entrada
flow_1 = {}
flow_1['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\1\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_1['molar_flow'].append(F)

flow_1['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\1\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_1['molar_frac'].append(F)

flow_1 = pd.DataFrame.from_dict(flow_1, orient='columns', dtype=None)
flow_1.index = comp_list

flow_1

In [ ]: #Output em molar frac e molar flow para os componentes da corrente 16
flow_16 = {}
flow_16['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_16['molar_flow'].append(F)

flow_16['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_16['molar_frac'].append(F)

flow_16 = pd.DataFrame.from_dict(flow_16, orient='columns', dtype=None)
flow_16.index = comp_list

print (flow_16)
print ()

#Output em molar frac e molar flow para os componentes da corrente 21
flow_21 = {}
flow_21['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_21['molar_flow'].append(F)

flow_21['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_21['molar_frac'].append(F)

flow_21 = pd.DataFrame.from_dict(flow_21, orient='columns', dtype=None)
flow_21.index = comp_list

print (flow_21)

In [ ]: En_orig = Energy.Energy(asper)[-1]
Ex_loss_orig = Ex_loss.Exloss(asper)[-1]

print ('Energie Consumide - ',En_orig,' kw')
print ('Exergie Perdide - ',Ex_loss_orig,' kw')
print ('str 16 ',aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MASSFLW\\MIXED").Value)
print ('str 21 ',aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MASSFLW\\MIXED").Value)

```

```

In [ ]: T_B11 = aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Output\\B_TEMP").Value
T_B12 = aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Output\\B_TEMP").Value
T_B13 = aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Output\\B_TEMP").Value
T_B14 = aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Output\\B_TEMP").Value
print (T_B11,T_B12,T_B13,T_B14)

In [ ]: aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Input\\SPEC_OPT").Value = 'TP'
aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Input\\TEMP").Value = T_B11
aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Input\\TEMP").Value = T_B12
aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Input\\TEMP").Value = T_B13
aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Input\\TEMP").Value = T_B14

In [ ]: fig, axes = plt.subplots(nrows=1, ncols=2)

Data = pd.DataFrame([pd.Series(CGCC.stages_T(aspen,'T-01')), pd.Series(CGCC.h_CGCC(aspen,'T-01')[-1]),
pd.Series(Exergy.exergia(aspen,'T-01')[0]), pd.Series(np.arange(1,47,1))])

Data = Data.T
Data.columns = ['Stages_T','h_CGCC', 'Exergy', 'Stages']
Data = pd.DataFrame(Data)

Data.plot(x='Exergy', y = 'Stages',xlim = (0, 400), ax=axes[0],figsize=(24, 5))
Data.plot(x='Exergy', y = 'Stages_T',xlim = (0, 400), ax =axes[1],figsize=(24, 5))

In [ ]: # obtencao do polinomio para o calculo da vazao de C3 em funcao da pressao do C3 no trocador P-04 e P-02
Pressao_C3_rico = [1., 1.2, 1.3, 1.5, 1.7, 1.8, 2., 2.3, 2.5]
vazao_C3_rico = [81800., 80100., 79800., 79000., 78500., 78300., 77800., 77000., 76650.]
y = np.polyfit(Pressao_C3_rico, vazao_C3_rico, 3)# ,full = True)
vazao_rico = np.poly1d(y)

Pressao_V01_rico = [3.5, 3.8, 4., 4.3, 4.5, 4.8, 5., 5.3, 5.5, 5.8, 6., 6.3, 6.5]
vazao_5P_1_rico = [44900., 44100., 43700., 42900., 42500., 42000., 41600., 41100., 40850., 40450., 40000., 39600., 39300.]
x2 = np.polyfit(Pressao_V01_rico, vazao_5P_1_rico, 3)# ,full = True)
vazao_5P_rico = np.poly1d(x2)

#polinomio para variacao de temperatura em 48
Temp_48_rico = [-11., -13., -15., -17., -19., -21., -23]
vazao_T_rico = [-8500., -5700., -3000., 0., 2900., 6000., 9000.]
t_48_rico = np.polyfit(Temp_48_rico, vazao_T_rico, 3)# ,full = True)
vazao_t48_rico = np.poly1d(t_48_rico)

In [ ]: creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

FLT_MIN, FLT_MAX = 0.02, 0.98 #range do split B18
FLT_MIN2, FLT_MAX2 = 23.0, 30.0 #range de pressao do turbo expensor a coluna T01 (valor inicial TE-01 = 26.05477424)
FLT_MIN3, FLT_MAX3 = 1.0, 2.5 #range de pressao do propano no segundo trocador
FLT_MIN4, FLT_MAX4 = 3.5, 6.5 #range de pressao do propano no primeiro trocador
FLT_MIN5, FLT_MAX5 = -11., -23. #range de temperatura da corrente 48

N_CYCLES = 1

toolbox = base.Toolbox()

toolbox.register("attrflt", random.uniform, FLT_MIN, FLT_MAX)
toolbox.register("attrflt2", random.uniform, FLT_MIN2, FLT_MAX2)
toolbox.register("attrflt3", random.uniform, FLT_MIN3, FLT_MAX3)
toolbox.register("attrflt4", random.uniform, FLT_MIN4, FLT_MAX4)
toolbox.register("attrflt5", random.uniform, FLT_MIN5, FLT_MAX5)
toolbox.register("aspen", aspen)

func_seq = [toolbox.attrflt, toolbox.attrflt2, toolbox.attrflt3, toolbox.attrflt4, toolbox.attrflt5, toolbox.aspen]
print (func_seq)

In [ ]: toolbox.register("individual", tools.initCycle, creator.Individual,func_seq, n=N_CYCLES)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

pop = toolbox.population(n=10)

```

```

In [ ]: def evalOneMin(individual):

    #variaveis da otimizacao
    aspen.Tree.FindNode("\\Data\\Blocks\\B18\\Input\\FRAC\\9").Value = individual[0]    #B18 (split corrente 9) = 0.8
    aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES").Value = individual[1]    #T-01 = 26.85477424
    aspen.Tree.FindNode("\\Data\\Blocks\\V02\\Input\\P_OUT").Value = individual[2]    #P_out = 1.5 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\V01\\Input\\P_OUT").Value = individual[3]    #P_out = 6 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\P-04\\Input\\VALUE").Value = individual[4]    #Temp original = -17

    # pressoes do V-05, T-01, V-06, S-02, valvula B10 e valvula B4
    aspen.Tree.FindNode("\\Data\\Blocks\\V-05\\Input\\PRES").Value = individual[1]    #26.6469282
    aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Input\\PRES").Value = individual[1] - 0.5 #25.55131261
    aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES1").Value = individual[1] - 0.6 #25.48235875
    aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Input\\PRES").Value = individual[2]    # original 1.5 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\B4\\Input\\P_OUT").Value = individual[1]    #26.6469282
    aspen.Tree.FindNode("\\Data\\Blocks\\B10\\Input\\P_OUT").Value = individual[1]    #26.6469282
    aspen.Tree.FindNode("\\Data\\Streams\\3P\\Input\\TOTFLOW\\MIXED").Value = vazao_rico(individual[2])+vazao_t48_rico(individ
    aspen.Tree.FindNode("\\Data\\Blocks\\B7\\Input\\BASIS_FLOW\\5P").Value = vazao_5P_rico(individual[3])
    aspen.Tree.FindNode("\\Data\\Blocks\\S02\\Input\\PRES").Value = individual[3]    # original 6 bar
    aspen.Tree.FindNode("\\Data\\Blocks\\C01\\Input\\PRES").Value = individual[3]    # original 6 bar

    aspen.Engine.Run2()

    print (aspen.Tree.FindNode("\\Data\\Results Summary\\Run-Status\\Output\\PER_ERROR").Value)
    print ('T-01 ', aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES").Value)
    print ('split_B18 ', aspen.Tree.FindNode("\\Data\\Blocks\\B18\\Input\\FRAC\\9").Value)
    print ('pres_prop ', aspen.Tree.FindNode("\\Data\\Blocks\\V02\\Input\\P_OUT").Value )
    print ('V-05 ', aspen.Tree.FindNode("\\Data\\Blocks\\V-05\\Input\\PRES").Value)
    print ('V-06 ', aspen.Tree.FindNode("\\Data\\Blocks\\V-06\\Input\\PRES").Value)
    print ('T-01 ', aspen.Tree.FindNode("\\Data\\Blocks\\T-01\\Input\\PRES1").Value)
    print ('S-03 ', aspen.Tree.FindNode("\\Data\\Blocks\\S03\\Input\\PRES").Value)
    print ('S-02 ', aspen.Tree.FindNode("\\Data\\Blocks\\S02\\Input\\PRES").Value)
    print ('vazao_3P ', aspen.Tree.FindNode("\\Data\\Streams\\3P\\Output\\MASSFLMX\\MIXED").Value)
    print ('vazao_5P ', aspen.Tree.FindNode("\\Data\\Streams\\5P\\Output\\MASSFLMX\\MIXED").Value)
    print ('Temp 48 ', aspen.Tree.FindNode("\\Data\\Streams\\48\\Output\\TEHP_OUT\\MIXED").Value)
    print ('energia ', Energy.Energy(aspen)[-1], ((Energy.Energy(aspen)[-1]-En_orig)/En_orig)*100)
    print ('exergia ', Ex_loss.ExLoss(aspen)[-1], ((Ex_loss.ExLoss(aspen)[-1]-ExLoss_orig)/ExLoss_orig)*100)
    #print (En_orig)
    #print (ExLoss_orig)
    #print (plt.plot(Exergy.exergia(aspen, 'T-01')[0], np.arange(1,47,1)))

    print ()
    exergy_loss = Ex_loss.ExLoss(aspen)[-1] #exergia total kw
    #energy_loss = Energy.Energy(aspen)[-1] #energia consumida total kw

    return exergy_loss, #esta virgula no final eh essencial
    #return energy_loss, #esta virgula no final eh essencial

```

```

In [ ]: toolbox.register("mate", tools.cxTwoPoint)
        toolbox.register("mutate", tools.mutFlipBit, indpb=0.0) # passo de 0.05 para 0.0
        toolbox.register("select", tools.selTournament, tournsize=3)
        toolbox.register("evaluate", evalOneMin)

```

```

In [ ]: ind1 = toolbox.individual()
        print (ind1)
        print ()
        print (ind1.fitness.valid)

```

```

In [ ]: ind1.fitness.values = evalOneMin(ind1)
        print (ind1.fitness.valid)
        print (ind1.fitness)
        print (ind1)

```

```

In [ ]: result = algorithms.eaSimple(pop, toolbox, cxpb=0.5, mutpb=0., ngen=10, verbose=False)
        print('Current best fitness:', evalOneMin(tools.selBest(pop, k=1)[0]))
        print (tools.selBest(pop, k=1)[0])

```

```

In [ ]: print ('Energia Consumida - ', Energy.Energy(aspen)[-1], ' kw')
        print ('Exergia Perdida - ', Ex_loss.ExLoss(aspen)[-1], ' kw')
        print ('str 16 ', aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MASSFLMX\\MIXED").Value)
        print ('str 21 ', aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MASSFLMX\\MIXED").Value)

```

```

In [ ]: Ex_loss.ExLoss(aspen)

```



```
In [ ]: #Output em molar frac e molar flow para os componentes da corrente 16
flow_16 = {}
flow_16['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_16['molar_flow'].append(F)

flow_16['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\16\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_16['molar_frac'].append(F)

flow_16 = pd.DataFrame.from_dict(flow_16, orient='columns', dtype=None)
flow_16.index = comp_list

flow_16
```

```
In [ ]: #Output em molar frac e molar flow para os componentes da corrente 21
flow_21 = {}
flow_21['molar_flow'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFLOW\\MIXED\\{0}".format(i)).Value
    flow_21['molar_flow'].append(F)

flow_21['molar_frac'] = []
for i in comp_list:
    F = aspen.Tree.FindNode("\\Data\\Streams\\21\\Output\\MOLEFRAC\\MIXED\\{0}".format(i)).Value
    flow_21['molar_frac'].append(F)

flow_21 = pd.DataFrame.from_dict(flow_21, orient='columns', dtype=None)
flow_21.index = comp_list

flow_21
```

```
In [ ]: T_B11 = aspen.Tree.FindNode("\\Data\\Blocks\\B11\\Output\\B_TEMP").Value
T_B12 = aspen.Tree.FindNode("\\Data\\Blocks\\B12\\Output\\B_TEMP").Value
T_B13 = aspen.Tree.FindNode("\\Data\\Blocks\\B13\\Output\\B_TEMP").Value
T_B14 = aspen.Tree.FindNode("\\Data\\Blocks\\B14\\Output\\B_TEMP").Value
print (T_B11,T_B12,T_B13,T_B14)
```

```
In [ ]: Energy.Energy(aspern)
```

```
In [ ]: plt.plot(CGCC.h_CGCC(aspern, 'T-01')[-1], CGCC.stages_T(aspern, 'T-01'))
```

```
In [ ]: fig, axes = plt.subplots(nrows=1, ncols=2)

Data = pd.DataFrame([pd.Series(CGCC.stages_T(aspern, 'T-01')), pd.Series(CGCC.h_CGCC(aspern, 'T-01')[-1]),
                    pd.Series(Exergy.exergia(aspern, 'T-01')[0]), pd.Series(np.arange(1,47,1))])

Data = Data.T
Data.columns = ['Stages_T', 'h_CGCC', 'Exergy', 'Stages']
Data = pd.DataFrame(Data)

Data.plot(x='Exergy', y='Stages', xlim = (0, 400), ax=axes[0], figsize=(24, 5))
Data.plot(x='Exergy', y='Stages_T', xlim = (0, 400.), ax =axes[1], figsize=(24, 5))
```

```
In [ ]: Exergy.exergia(aspern, 'T-01')[0]
```