

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA QUÍMICA
CURSO DE GRADUAÇÃO EM ENGENHARIA QUÍMICA



**OTIMIZAÇÃO DE UMA REDE DE TROCADORES DE CALOR ASSOCIADOS À
PRODUÇÃO DE ÓLEO VEGETAL**

Lorena Pinto Neto Marçon

Uberlândia – MG

2018

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA QUÍMICA
CURSO DE GRADUAÇÃO EM ENGENHARIA QUÍMICA

**OTIMIZAÇÃO DE UMA REDE DE TROCADORES DE CALOR ASSOCIADOS À
PRODUÇÃO DE ÓLEO VEGETAL**

Lorena Pinto Neto Marçon

**Monografia de graduação apresentada à
Universidade Federal de Uberlândia como
parte dos requisitos necessários para a
aprovação na disciplina de Projeto de
Graduação no curso de Engenharia Química.**

Uberlândia – MG

2018

MEMBROS DA BANCA EXAMINADORA DA MONOGRAFIA DA DISCIPLINA
PROJETO DE GRADUAÇÃO DE LORENA PINTO NETO MARÇON APRESENTADA À
UNIVERSIDADE FEDERAL DE UBERLÂNDIA, EM 17 DE JANEIRO DE 2019.

BANCA EXAMINADORA:

Prof. Ubirajara Coutinho Filho

Orientador – FEQUI/UFU

Prof. Rubens Gedraite

FEQUI/UFU

Prof. Humberto Molinar Henrique

FEQUI/UFU

AGRADECIMENTOS

Ao orientador Prof. Ubirajara Coutinho Filho, por todo o conhecimento e suporte dado durante o desenvolvimento deste trabalho.

Aos meus pais Edson e Mara e à minha irmã Larissa, por sempre estarem do meu lado, pelo incentivo e pelo exemplo que são para mim.

Ao meu namorado Gustavo por todo o suporte, apoio e compreensão durante todos estes anos ao meu lado.

Ao meu cão Hércules por me lembrar todos os dias a importância dos pequenos momentos.

Aos meus amigos por todas as memórias feitas, momentos compartilhados e desafios superados durante a graduação.

*“O que é escrito sem esforço em geral é lido
sem prazer”*

Samuel Johnson

SUMÁRIO

Lista de Abreviaturas.....	i
Resumo.....	ii
 CAPÍTULO 1 – INTRODUÇÃO.....	 1
 CAPÍTULO 2 – REVISÃO BIBLIOGRÁFICA.....	 3
2.1 – BENEFICIAMENTO DE SOJA.....	3
2.2 – MÉTODO DO ENXAME DE PARTÍCULAS.....	5
 CAPÍTULO 3 – DESENVOLVIMENTO.....	 8
3.1 – ELABORAÇÃO DO CÓDIGO-FONTE INICIAL.....	8
3.2 – VALIDAÇÃO DO PROGRAMA PARA OTIMIZAÇÃO DE TROCAS TÉRMICAS..	8
3.3 – APLICAÇÃO DO PROGRAMA À INDÚSTRIA DE SOJA.....	9
 CAPÍTULO 4 – RESULTADOS E DISCUSSÕES.....	 10
4.1 – PERFORMANCE DO PROGRAMA PARA O 1º CASO DE ESTUDO.....	10
4.2 – PERFORMANCE DO PROGRAMA PARA O 2º CASO DE ESTUDO.....	14
4.3 – PERFORMANCE DO PROGRAMA PARA A INDÚSTRIA DE SOJA.....	18
 CAPÍTULO 5 – CONCLUSÕES E SUGESTÕES.....	 23
5.1 – CONCLUSÕES.....	23
5.2 – SUGESTÕES.....	23
 REFERÊNCIAS BIBLIOGRÁFICAS.....	 24
 APÊNDICE A – CÓDIGO-FONTE DO PRIMEIRO CASO DE ESTUDO.....	 25
APÊNDICE B – CÓDIGO-FONTE DO SEGUNDO CASO DE ESTUDO, COM DIVISÃO DE UMA CORRENTE.....	32
APÊNDICE C – CÓDIGO-FONTE DA OTIMIZAÇÃO PARA A INDÚSTRIA DE BENEFICIAMENTO DE SOJA.....	40

LISTA DE ABREVIATURAS

A – Área

$\Delta T_{\min}$ – Diferença de temperatura mínima nas extremidades dos trocadores de calor

g_{best} – Melhor posição conhecida entre todas as partículas

mCp – Capacidade térmica

p_{best} – Melhor posição conhecida da partícula

PSO – *Particle Swarm Optimization* (em português, Otimização por Enxame de Partículas)

Q_F – Taxa de utilidade fria

Q_Q – Taxa de utilidade quente

TAC – *Total Annual Cost* (em português, Custo Total Anual)

T_{entrada} – Temperatura de entrada

$T_{\text{saída}}$ – Temperatura de saída

U – Coeficiente global de troca térmica

UF – Utilidade fria

UQ – Utilidade quente

RESUMO

A otimização de redes de trocadores de calor é uma necessidade atual em diversas indústrias e que pode trazer benefícios, quando feita. Dentre as diversas metodologias possíveis para se fazer uma integração energética, tem-se os métodos heurísticos baseados em comportamentos da natureza, como é o caso do Método do Enxame de Partículas. No presente trabalho é feita a otimização por Enxame de Partículas para as correntes de uma indústria. A fábrica escolhida para a otimização é uma fábrica de processamento de soja, visto a grande importância econômica e social da soja e seus subprodutos para o Brasil. Assim, inicialmente, elaborou-se um código inicial do Método do Enxame de Partículas, que foi validado com a Função de Rosenbrock, de mínimo global já conhecido. Em uma segunda etapa, adaptou-se o código para realizar a integração energética de um grupo de correntes, sendo a função objetivo minimizar o Custo Total Anual. Nesta segunda etapa, ainda, validou-se o código com dois casos já conhecidos na literatura. Por fim, o código foi aplicado para um grupo de correntes da indústria de soja e os resultados foram comparados com os reportados na literatura. Os códigos gerados neste trabalho reportaram redes de trocadores de calor com Custo Total Anual aproximadamente 19% menores quando comparados ao que é encontrado na literatura, em média. Conclui-se, então, que o código gerado neste trabalho, baseado no Método do Enxame de Partículas, gera respostas próximas às da literatura e conseguiu promover redes de trocadores de calor para todos os casos estudados.

Palavras-chave: Integração Energética, Otimização por Enxame de Partículas, Soja.

CAPÍTULO 1

INTRODUÇÃO

A engenharia e seus processos estão sempre evoluindo com as novas tecnologias. Logo, processos de produção que, anteriormente, eram o cenário mais eficiente, hoje podem ser melhorados. A melhoria pode também ocorrer em empresas que acabaram de nascer, pois nem sempre a indústria é criada com seu cenário de produção mais eficiente. Falta de capital para erguer a planta ideal é um desses casos. Assim, a empresa decide iniciar a operação para captar recursos, através dos seus lucros, por exemplo, para investir na melhoria da planta.

A melhoria do processo de produção da indústria geralmente tem por objetivo tornar o processo mais econômico. Um processo com menor custo agregado não representa somente uma economia para a empresa, mas pode refletir também em um menor custo do produto. Além, o capital economizado pode ser investido internamente (expansão da empresa, adoção de um novo processo, pesquisas de otimização, entre outros), o que pode gerar novas oportunidades de emprego, ou externamente (mercado financeiro, por exemplo).

Para diminuir custos do processo, há diversos aspectos que podem ser analisados e otimizados, tais como rendimento do processo, quantidade de matéria-prima utilizada, rota de produção, substituição de equipamentos, entre outros. Outra oportunidade para redução de custos é promover a integração energética da unidade, que pode ser feita através da implementação de uma rede de trocadores de calor.

Num cenário sem otimização, o aquecimento ou resfriamento das correntes é dado pelo uso de utilidades quentes ou frias, que representam um custo dentro do processo. O princípio da integração energética é substituir o uso de utilidades, quando possível, por correntes do próprio processo, ou seja: promove-se trocas térmicas entre as correntes do processo. A viabilidade depende de diversos fatores, como por exemplo distância física das correntes, temperatura e riscos associados ao contato das correntes.

Pode-se transformar a otimização desta rede em um problema matemático, no qual busca-se a resposta que minimize a função objetivo. Esta função objetivo pode ser, por exemplo, o consumo de energia gerado pelo uso de utilidades, a quantidade de utilidades utilizada, o custo para a aquisição e instalação dos equipamentos da rede de trocadores de calor ou pode ser um somatório de diversos fatores.

Neste cenário nasce a ideia deste trabalho, que possui como objetivo o desenvolvimento de um programa capaz de esquematizar uma rede de trocadores de calor para uma unidade fabril que promova o menor custo financeiro, respeitando as limitações impostas pelo usuário do programa.

A fábrica escolhida para a síntese de uma rede de trocadores de calor é uma fábrica de processamento de soja, visto a sua importância para o Brasil. Segundo a Associação Brasileira das Indústrias de Óleos Vegetais, o Brasil é responsável por 32% da produção mundial de soja e é o 2º maior produtor e exportador mundial da soja em grão, farelo e óleo de soja. Ainda, afirmam que, atualmente, o Complexo Soja (cadeia produtiva da soja em grão, farelo e óleo) é um dos principais itens da Balança Comercial Brasileira e exportou US\$ 25,4 bilhões em 2016.

A Figura 1.1 abaixo traz a estruturação definida para estudo e desenvolvimento do presente trabalho, para facilitar a compreensão das etapas envolvidas para elaboração do programa final.

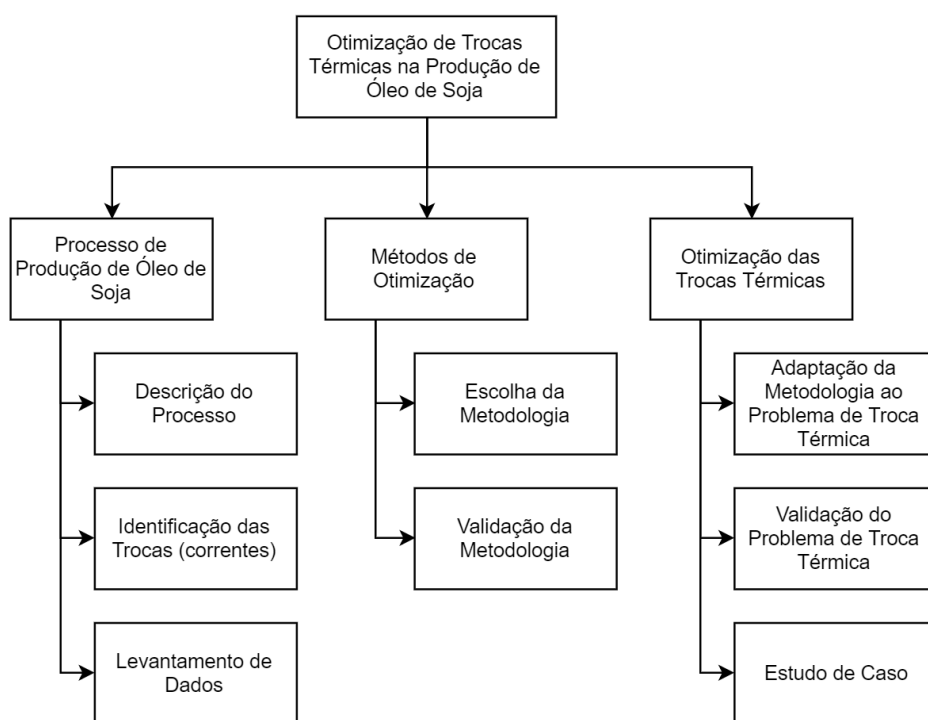


Figura 1.1 – Fluxograma descritivo das etapas do trabalho.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

2.1 – BENEFICIAMENTO DA SOJA

No programa, será feita a síntese da rede de trocadores de calor de uma planta industrial de beneficiamento de soja. A etapa industrial na qual será feita a síntese é a produção de farelo de soja e extração de óleo. Assim, será feita uma explicação geral do funcionamento de uma planta de extração de óleo e produção de farelo de soja para auxiliar na interpretação e compreensão do restante do trabalho.

Inicialmente, a soja passa por um processo de pré-tratamento, no qual passa pelas seguintes etapas: limpeza, quebra, aspiração, condicionamento e laminação. É feita a limpeza dos grãos (através de peneiras para remoção de possíveis impurezas que podem ter vindo do campo, tais como terra ou talos de planta, ou outras impurezas com as quais a soja teve contato). Depois, a soja segue para os quebradores. Os grãos são quebrados com a intenção de aumentar a área superficial (para acelerar a extração do óleo) e facilitar a laminação da mesma posteriormente. A aspiração é feita após a quebra dos grãos para remover as cascas de soja do processo. Após, é feito o condicionamento dos pedaços de soja, os quais podemos chamar de polpa de soja. O condicionamento é feito para aquecer a polpa e prepará-la para a etapa seguinte, laminação. No processo de laminação, a polpa de soja é convertida em uma lâmina fina (lâmina de soja). Esta etapa tem por objetivo aumentar ainda mais a área superficial da soja para aumentar a eficiência do processo de extração sólido-líquido. Neste momento, a lâmina de soja está pronta para ir para a etapa de extração.

É importante ressaltar que há diversas metodologias para extração do óleo; neste trabalho será abordado a extração sólido-líquido através do uso de solvente hexano. Na etapa de extração, o hexano entra em contato com as lâminas de soja para remover o óleo. Assim, geram-se duas correntes: uma corrente sólida com as lâminas de soja e hexano (farelo de soja) e uma corrente líquida de hexano e óleo (miscela). Assim, é necessário fazer uma próxima etapa, a dessolventização, para remover o solvente das duas correntes.

A corrente que contém as lâminas de soja passa ainda por alguns processos para redução da umidade do material (tostagem, secagem, etc). Por fim, é moída para gerar o produto farelo de soja. A miscela ainda passa por processos de degomagem (para remoção de subprodutos), neutralização e clareamento do óleo, entre outros, para, por fim, gerar o óleo de soja.

A Figura 2.1 apresenta um esquema simplificado de todo o processo de produção do óleo de soja e farelo de soja pela rota de extração com solvente orgânico.

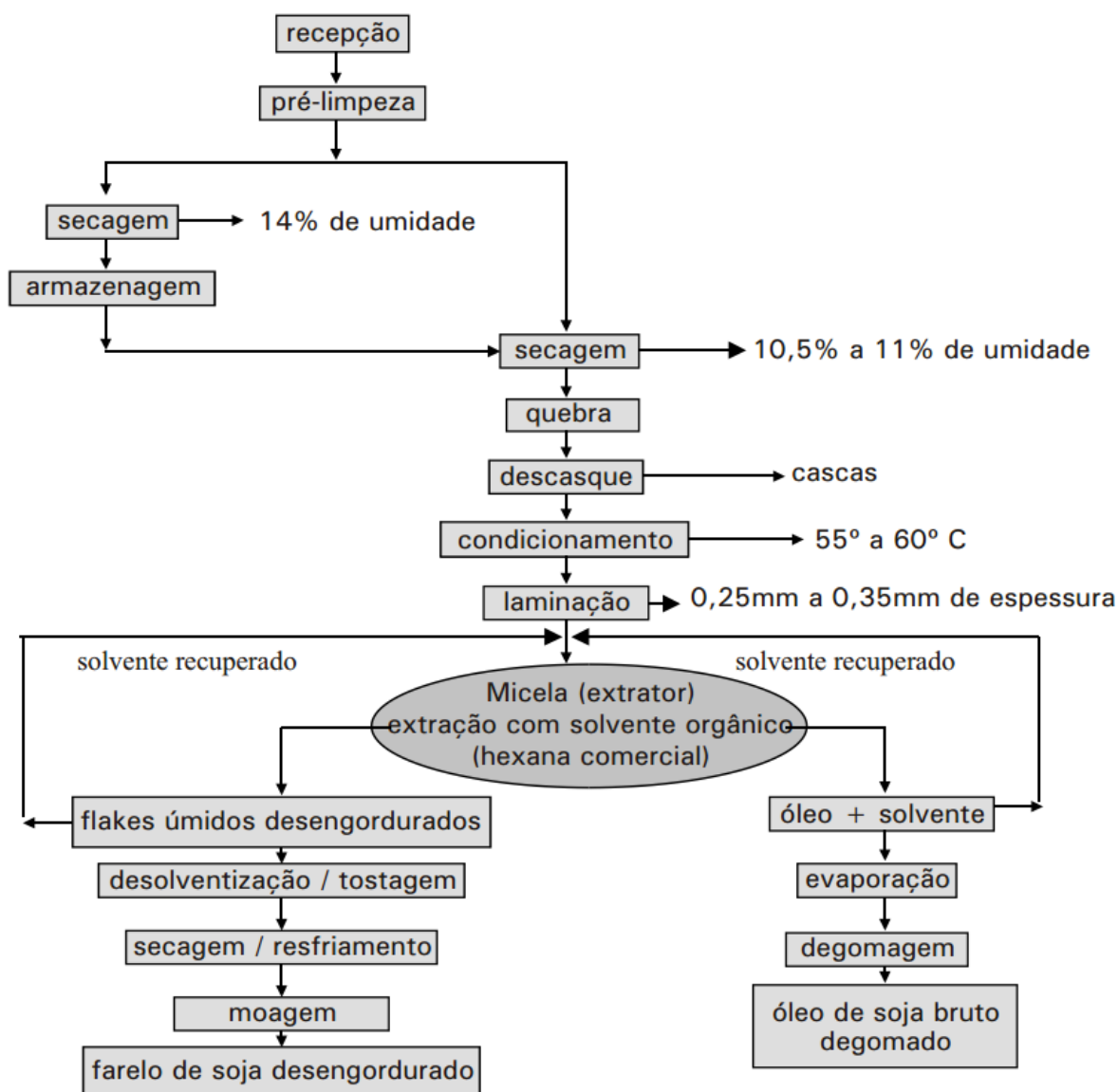


Figura 2.1 – Fluxograma do processo de obtenção do óleo de soja bruto e farelo de soja desengordurado.

Fonte: EMBRAPA, 2001, p. 19.

2.2 – MÉTODO DO ENXAME DE PARTÍCULAS

Há diversos métodos de otimização que podem ser utilizados na resolução do problema deste trabalho, como métodos baseados em técnicas de Programação Matemática, baseados em regras heurísticas, entre outros. Segundo PAVÃO et al. (2018), abordagens meta-heurísticas se tornaram uma forte tendência na atual literatura e têm gerado resultados notáveis em diversas pesquisas. Assim, a metodologia escolhida para desenvolvimento de trabalho é uma meta-heurística e é conhecida como Método do Enxame de Partículas.

KENNEDY; EBERHART (1995) desenvolveram e publicaram o *Particle Swarm Optimization* (PSO), em português conhecido como Método do Enxame de Partículas. Conforme explica RAVAGNANI; SUÁREZ (2012), o Método do Enxame de Partículas explora a analogia com o comportamento social de animais, como enxames de abelhas, cardumes de peixes ou bando de pássaros. Nestes, cada indivíduo do grupo toma suas próprias decisões, mas sempre de alguma forma baseado na experiência do líder do grupo.

Matematicamente, o método funciona da seguinte forma: inicialmente gera-se uma matriz com a posição das partículas, distribuídas aleatoriamente, no qual cada partícula é uma proposta de solução para o problema. Testa-se o valor da função a ser minimizada para cada partícula. A partir deste ponto, inicia-se a procura pelos melhores resultados e são formados dois novos componentes: uma matriz com a melhor posição que cada partícula já teve e um vetor com a posição da partícula que levou ao melhor resultado até aquele momento, sendo esta última a “líder”. A cada iteração, as partículas caminham em busca da melhor posição (que trará o valor mínimo da função) e alteram suas posições de acordo com o vetor velocidade. A velocidade de cada partícula é calculada através da velocidade anterior da mesma, da melhor posição conhecida que a partícula já teve e da posição da partícula “líder”. A Figura 2.2 representa o que deve acontecer com as partículas à medida que as mesmas são atualizadas.

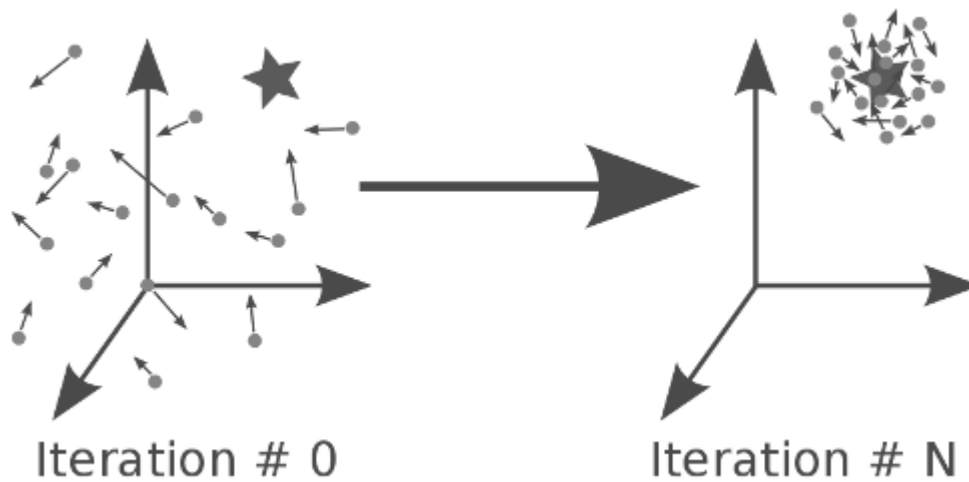


Figura 2.2 – Representação do comportamento das partículas frente ao mínimo global da função.

Fonte: Github. Disponível em: <<https://esa.github.io/pagmo2/docs/cpp/algorithms/psa.html>>. Acesso em 09 set. 2018.

RAO (2009) apresenta as equações da velocidade e posição de cada partícula, que são mostradas abaixo, respectivamente, pelas Equações 2.1 e 2.2.

$$\mathbf{v}_{k+1} = w \cdot \mathbf{v}_k + c_1 \cdot r_1 \cdot (\mathbf{p}_{best,k} - \mathbf{x}_k) + c_2 \cdot r_2 \cdot (\mathbf{g}_{best,k} - \mathbf{x}_k) \quad (2.1)$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_{k+1} \quad (2.2)$$

Onde:

k – iteração

w – coeficiente de inércia

c_1 – coeficiente de aprendizado cognitivo (individual)

c_2 – coeficiente de aprendizado social (em grupo)

r_1, r_2 – números randômicos pertencentes ao intervalo $[0,1]$

v – velocidade da partícula

p_{best} – melhor posição conhecida da partícula

g_{best} – melhor posição conhecida entre todas as partículas

x – posição da partícula

Na primeira iteração, a posição de cada partícula e a velocidade das mesmas são geradas aleatoriamente, bem como todas as posições iniciais são consideradas as melhores posições de cada partícula, pois ainda são as únicas conhecidas. Nas iterações seguintes, há sempre a comparação com os resultados da iteração anterior para atualizar os valores das matrizes e vetores.

A Figura 2.3 mostra um fluxograma simplificado da lógica da metodologia PSO para resolução de problemas matemáticos.

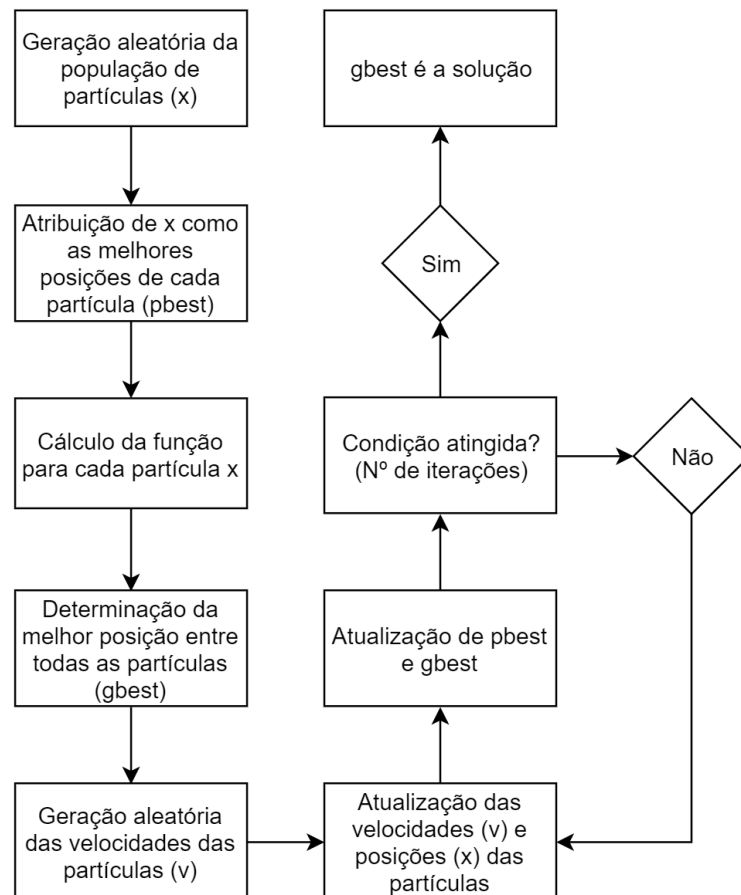


Figura 2.3 – Fluxograma da lógica do método PSO.

CAPÍTULO 3

DESENVOLVIMENTO

3.1 – ELABORAÇÃO DO CÓDIGO-FONTE INICIAL

A plataforma escolhida para desenvolvimento do programa é o Scilab 6.0.0. Durante o desenvolvimento deste trabalho, elaborou-se diversas versões do programa, na intenção de testar o efeito de possíveis modificações no programa e seu impacto nos resultados. O primeiro código-fonte gerado foi o do Método do Enxame de Partículas e o mesmo foi testado e validado com uma função objetivo de resposta conhecida. A função escolhida para o teste foi a função de Rosenbrock, apresentada na Equação 3.1, que possui mínimo global em $(x, y) = (a, a^2)$, onde $f(x, y) = 0$.

$$f(x, y) = (a - x)^2 + b \cdot (y - x^2)^2 \quad (3.1)$$

O programa inicial foi escrito com o objetivo de encontrar a posição que retorna o valor mínimo da função de Rosenbrock. Os testes foram bem-sucedidos e as diversas execuções levaram a posições próximas ao mínimo global da função, conforme era esperado.

3.2 – VALIDAÇÃO DO PROGRAMA PARA OTIMIZAÇÃO DE TROCAS TÉRMICAS

O código-fonte gerado na Seção 3.1 e validado com a função de Rosenbrock é, então, modificado para atender os propósitos deste trabalho. Nesta nova versão, o programa foi transformado para otimizar uma rede de trocadores de calor. Assim, baseado nos dados das correntes, o programa retorna a rede de trocas térmicas que gera o menor Custo Total Anual (TAC), que é composto pelo custo dos trocadores de calor, custos de utilidade quente e utilidade fria. Assim, a função objetivo (neste caso, representada pelo TAC) é dada pela Equação 3.2. As Equações que representam o custo dos trocadores e o custo das utilidades são detalhadas no Capítulo 4.

$$TAC = \text{custo dos trocadores} + \text{custo de } UQ + \text{custo de } UF \quad (3.2)$$

Para validação deste código-fonte, utilizou-se problemas já conhecidos e resolvidos na literatura, a fim de comparar a solução retornada pelo programa. Estes problemas são apresentados no trabalho de AGUITONI et al. (2018). As correntes, equações e outras informações utilizadas para execução do programa são apresentadas no Capítulo 4.

3.3 – APLICAÇÃO DO PROGRAMA À INDÚSTRIA DE SOJA

Após as validações do código para otimização de trocas térmicas, este foi utilizado para otimizar a rede de trocadores de calor de uma indústria de beneficiamento de soja, descrita no trabalho de CERETO (2004), do qual extraiu-se também os dados necessários para realizar a otimização. Os resultados obtidos e demais informações são apresentados no Capítulo 4.

CAPÍTULO 4

RESULTADOS E DISCUSSÕES

4.1 – PERFORMANCE DO PROGRAMA PARA O 1º CASO DE ESTUDO

O primeiro caso de estudo foi extraído do trabalho de AGUITONI et al. (2018). Este é composto por 4 correntes, sendo 2 correntes frias e 2 correntes quentes. Abaixo, é apresentada a Tabela 4.1, que contém as correntes do primeiro caso de estudo.

Tabela 4.1 – Dados das correntes do primeiro caso de estudo.

Corrente	T _{entrada} (K)	T _{saída} (K)	mCp (kW/K)
H1	533	433	3,0
H2	523	403	1,5
C1	393	508	2,0
C2	453	513	4,0

Fonte: AGUITONI et al. (2018).

Para resolução do problema acima, é necessário definir as condições da utilidade quente e utilidade fria. Assim, considerou-se como utilidade quente o vapor de água saturado a 250°C (523,15 K) e 39,8 bar e como utilidade fria a água à temperatura de entrada ambiente de 25°C (298,15 K) e temperatura de saída de 30°C (303,15 K). Além, considerou-se também que, nas extremidades dos trocadores de calor, deve existir uma diferença de temperatura de, no mínimo, 10°C.

Ainda, são extraídas de AGUITONI et al. (2018) outras informações essenciais para a execução adequada do programa, apresentadas nas Equações 4.1 a 4.4, que representam, respectivamente, o custo do trocador de calor em função da área necessária para troca térmica; o custo de utilidade quente - UQ; o custo de utilidade fria - UF e o valor do coeficiente global de troca térmica – U.

$$\text{Custo do trocador (\$)} = 300(A^{0,5}) \quad (4.1)$$

$$\text{Custo de } UQ = 110 \frac{\$}{kW \cdot \text{ano}} \quad (4.2)$$

$$\text{Custo de } UF = 12,2 \frac{\$}{kW \cdot \text{ano}} \quad (4.3)$$

$$U = 0,2 \frac{kW}{m^2 K} \quad (4.4)$$

Logo, a função objetivo para este caso é dada pela substituição das Equações 4.1, 4.2 e 4.3 na Equação 3.2, obtendo-se, então, a Equação 4.5.

$$TAC = 300(A^{0,5}) + 110Q_Q + 12,2Q_F \quad (4.5)$$

A Tabela 4.2 contém soluções de otimização da rede de trocadores de calor do primeiro caso de estudo, desenvolvido por diferentes autores, com as informações de TAC, número de trocadores, área de troca térmica, taxa de utilidade quente e taxa de utilidade fria. É importante ressaltar que nem todos os trabalhos apresentados na Tabela 4.2 utilizam a mesma função objetivo nas suas resoluções e utilizam diferentes metodologias. Contudo, as informações da Tabela 4.2 são utilizadas para avaliar os resultados obtidos pelo código gerado neste trabalho.

Tabela 4.2 – Dados das redes de trocadores de outros trabalhos para o primeiro caso estudado.

	TAC (\$/ano)	Nº de Trocadores	Área (m²)	Q_Q (kW)	Q_F (kW)
AHMAD (1985)	12.870	7	...	50,00	60,00
NIELSEN et al. (1996)	12.306	8	55,83	36,00	45,00
KHORASANY, FESANGHARY (2009)	11.895	7	...	18,10	28,10
HUO et al. (2012)	11.632	7	...	17,80	27,80
AGUITONI et al. (2018)	11.540	7	242,26	18,14	28,14

Fonte: AGUITONI et al. (2018).

O código utilizado para resolver este problema é apresentado no Apêndice A. O mesmo foi executado 10 vezes e, então, selecionou-se o melhor resultado dentre as 10 execuções. A Tabela 4.3 apresenta os resultados obtidos pelo programa, o que contempla o TAC, número de trocadores, área de troca térmica, taxa de utilidade quente e taxa de utilidade fria.

Tabela 4.3 – Dados da rede de trocadores de calor obtida para o primeiro caso.

	TAC (\$/ano)	Nº de Trocadores	Área (m²)	Q_Q (kW)	Q_F (kW)
Este trabalho	9.830	7	146,79	50,00	57,00

Ao avaliar o resultado gerado por este trabalho em comparação com os demais trabalhos, observa-se que o valor de TAC gerado pelo código foi 18,4% menor do que a média dos valores de TAC dos outros trabalhos. A consideração de diferentes utilidades pelos outros autores é um dos fatores que pode gerar desvios no valor de TAC, bem como uma diferente consideração no valor de $\Delta T_{\min}$. Ao avaliar a taxa de utilidades, observa-se que o código propõe um uso de utilidades maior do que a maioria dos outros autores propõem.

O código ainda gera o gráfico apresentado na Figura 4.1, que mostra como as temperaturas das correntes alteram à medida que as trocas térmicas são feitas.

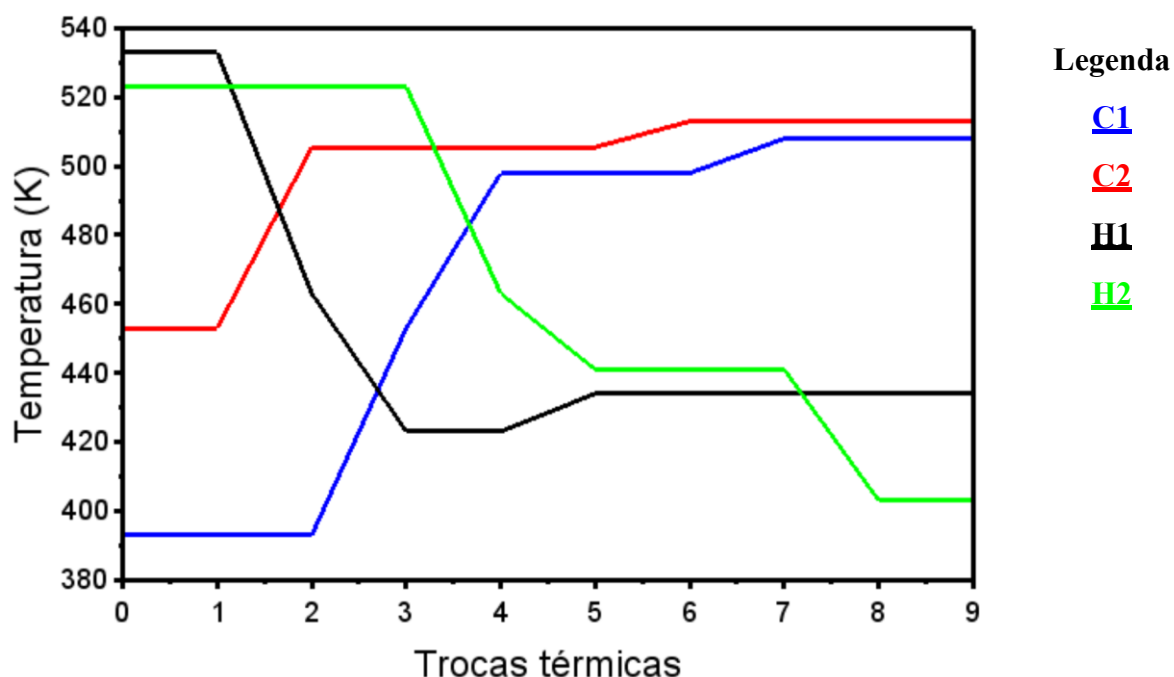


Figura 4.1 – Gráfico da variação das temperaturas das correntes com as trocas térmicas para o primeiro caso.

Este gráfico é útil pois, através dele, é possível ver não só as temperaturas das correntes a cada troca, como também quais correntes trocam calor entre si e quais correntes precisam de utilidades para atingir sua temperatura final. Ao analisar a Figura 4.1, neste caso, observa-se que a primeira troca térmica ocorre entre os índices 1 e 2 do eixo horizontal, onde é possível ver que a corrente H1 (preta) fornece calor para a corrente C2 (vermelha). Depois, a segunda troca ocorre entre os índices 2 e 3, onde a corrente H1 (preta) fornece calor para a corrente C1 (azul). Na terceira troca, a corrente H2 (verde) cede calor para a corrente C1 (azul). Na quarta troca, a corrente H2 (verde) cede calor para a corrente H1 (preta). A partir da quinta troca, observa-se que somente uma corrente varia sua temperatura, o que indica que a mesma está trocando calor com uma utilidade para atingir sua temperatura ideal. Assim, na quinta troca, a corrente C2 (vermelha) tem sua temperatura aumentada, ou seja, troca calor com a utilidade quente. Na sexta troca, a corrente C1 (azul) também troca calor com a utilidade quente. Por fim, na sétima e última troca (delimitada no eixo horizontal pelos números 7 a 8) ocorre com a corrente H2 (verde), que utiliza a utilidade fria para diminuir sua temperatura. Através da leitura no eixo vertical, é possível determinar a temperatura das correntes a cada troca térmica.

4.2 – PERFORMANCE DO PROGRAMA PARA O 2º CASO DE ESTUDO

O segundo caso de estudo também foi retirado do trabalho de AGUITONI et al. (2018). Este caso é composto por 4 correntes, sendo 2 correntes quentes e 2 correntes frias. Abaixo, é apresentada a Tabela 4.4, que contém as correntes deste segundo caso de estudo.

Tabela 4.4 – Dados das correntes do segundo caso de estudo.

Corrente	T_{entrada} (K)	T_{saída} (K)	mCp (kW/K)
H1	423	323	200
H2	443	313	100
C1	323	393	300
C2	353	383	500

Fonte: AGUITONI et al. (2018).

Para a resolução deste segundo caso, considerou-se no código o uso de vapor de água saturado a 142,9°C (416,05 K) e 4 bar como utilidade quente e, como utilidade fria, água à temperatura de entrada ambiente de 25°C (298,15 K) e temperatura de saída de 30°C (303,15K). Neste caso, também deve-se existir uma diferença de temperatura de, no mínimo, 10°C, nas extremidades dos trocadores de calor.

Visto que não há informação sobre a forma como é calculada a taxa de calor – Q para este segundo caso, considerou-se então a mesma fórmula utilizada no primeiro caso. Assim, a Equação 4.4 é empregada neste segundo caso de estudo. São utilizadas também as Equações 4.6 e 4.7, extraídas de AGUITONI et al. (2018), que representam, respectivamente, o custo de aquisição do trocador de calor em função da área necessária para troca térmica e o custo de Utilidade Fria – UF. Neste segundo caso o custo de utilidade quente não varia em relação ao primeiro caso; por conta disso a Equação 4.2 é utilizada aqui também.

$$\text{Custo do trocador (\$)} = 9094 + 221(A^{0,81}) \quad (4.6)$$

$$\text{Custo de UF} = 10 \frac{\$}{\text{kW} \cdot \text{ano}} \quad (4.7)$$

Assim, a função objetivo para o segundo caso é dada pela substituição das Equações 4.2, 4.6 e 4.7 na Equação 3.2. Obtém-se a Equação 4.8:

$$TAC = 9094 + 221(A^{0,81}) + 110Q_Q + 10Q_F \quad (4.8)$$

A Tabela 4.5 contém diferentes soluções de otimização da rede de trocadores de calor do segundo caso de estudo, desenvolvido pelos autores citados na mesma, e contém informações de Custo Total Anual – TAC, número de trocadores, área de troca térmica, taxa de utilidade quente – Q_Q e carga de utilidade fria – Q_F . Estes dados são utilizados como comparação com o resultado da otimização obtida pela metodologia do Enxame de Partículas, apresentado neste trabalho. Deve-se observar que nem todos os trabalhos apresentados na Tabela 4.5 consideram a mesma função objetivo nas suas resoluções, e também utilizam diferentes metodologias na otimização.

Tabela 4.5 – Dados das redes de trocadores obtidas por outros trabalhos para o segundo caso de estudo.

	TAC (\$/ano)	Nº de Trocadores	Área (m²)	Q_Q (kW)	Q_F (kW)
ZHU (1997)	1.818.031	8	19.690,00	7.000,00	7.000,00
SILVA et al. (2010)	1.816.470	8	...	7.260,00	4.260,00
PAVÃO et al. (2016)	1.808.047 ^a	7	18.641,20	7.854,55	4.854,54
AGUITONI et al. (2018)	1.803.626	7	19.493,61	7.653,87	4.653,87

^a Solução recalculada por AGUITONI et al. (2018).

Fonte: AGUITONI et al. (2018).

Este problema foi resolvido de duas maneiras diferentes. A primeira maneira utiliza o mesmo código que foi aplicado ao primeiro caso, com os valores das variáveis e equações

ajustados para o segundo caso. Já a segunda maneira propõe a divisão de uma das correntes do problema. A corrente escolhida para sofrer uma divisão é a corrente C2, que possui maior capacidade térmica em relação às demais. A quantidade da corrente que será fracionada será otimizada pelo próprio algoritmo.

O código utilizado para a resolução que propõe a divisão de uma corrente é apresentado no Apêndice B. Para ambos os casos, sem e com divisão da corrente, os códigos foram executados 10 vezes e selecionou-se, então, o melhor resultado das 10 execuções. A Tabela 4.6 traz os resultados obtidos em ambas aplicações, e informa o TAC, número de trocadores, área e taxas de utilidade quente e fria.

Tabela 4.6 – Dados da rede de trocadores de calor obtida para o segundo caso, sem divisão e com divisão da corrente C2.

	TAC (\$/ano)	Nº de Trocadores	Área (m²)	Q_Q (kW)	Q_F (kW)
Sem divisão da corrente C2	1.463.972	6	7.781,71	9.736,93	7.036,92
Com divisão da corrente C2	1.460.668	7	7.816,21	9.700,02	7.000,02

Ao comparar os resultados da Tabela 4.6 com os resultados reportados na literatura, conclui-se que o valor de TAC gerado pelos códigos sem e com divisão da corrente, respectivamente, foram 19,2% e 19,4% menores do que a média dos valores de TAC da literatura para este segundo caso. Observa-se também que os valores de área obtidos foram, em média, 59,5% menores do que os reportados na literatura. Os desvios nos valores de TAC e da área podem estar baseados no fato de que os demais autores consideram que os trocadores de calor devem possuir uma área mínima, e quando a área do trocador não atinge este número mínimo, a troca entre as duas correntes selecionadas não é feita. Neste trabalho não há nenhuma restrição para o tamanho da área dos trocadores, logo isto pode levar a malhas com áreas menores e TAC menores. Os desvios também podem ter sido gerados pela adoção de diferentes utilidades e diferente $\Delta T_{\min}$.

Ao comparar os resultados sem divisão e com divisão da corrente C2 entre si, verifica-se que o benefício financeiro gerado pela divisão da corrente (\$ 3.304) representa apenas 0,2% do montante total do TAC. Neste caso, é necessário avaliar o custo benefício de dividir uma

corrente, visto que a divisão exigirá nova tubulação, equipamentos de controle e segurança, entre outros. A divisão proposta pelo código também não é atrativa, visto que o mesmo propõe dividir a corrente em 99,94/0,06. Assim, conclui-se que não há benefício em dividir a corrente C2.

Por fim, cada código gerou um gráfico com as modificações das temperaturas das correntes em função das trocas térmicas, apresentados nas Figuras 4.2 e 4.3, para a proposta sem a divisão de correntes e com a divisão da corrente C2, respectivamente.

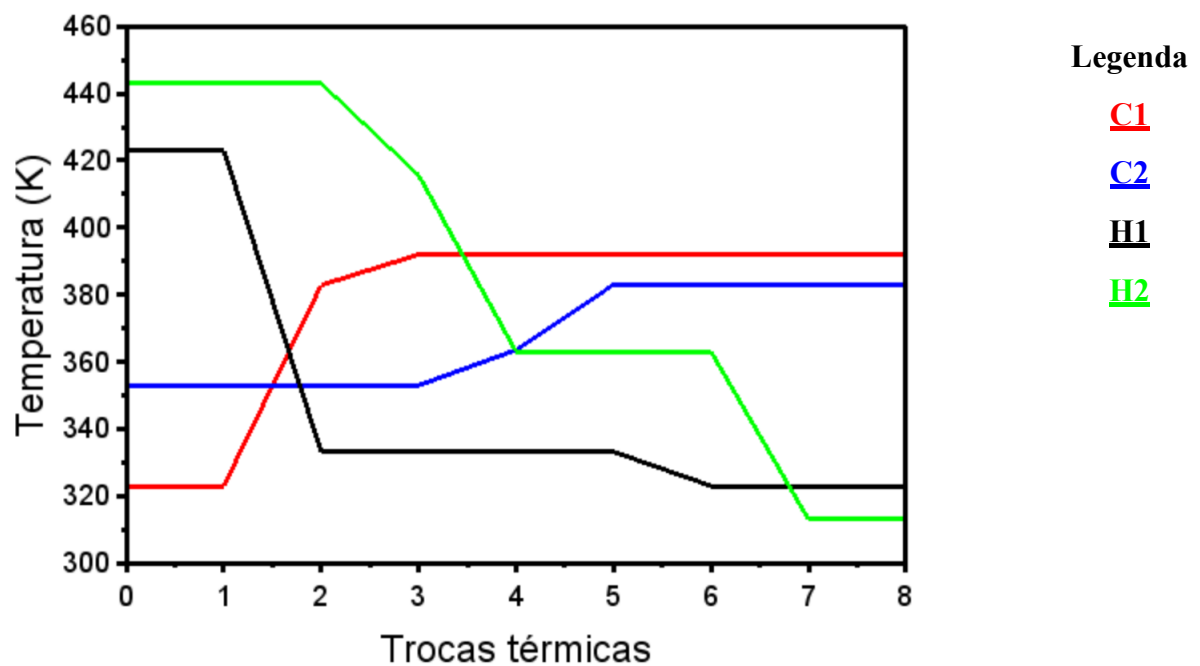


Figura 4.2 – Gráfico da variação das temperaturas das correntes com as trocas térmicas para o segundo caso, sem divisão de correntes.

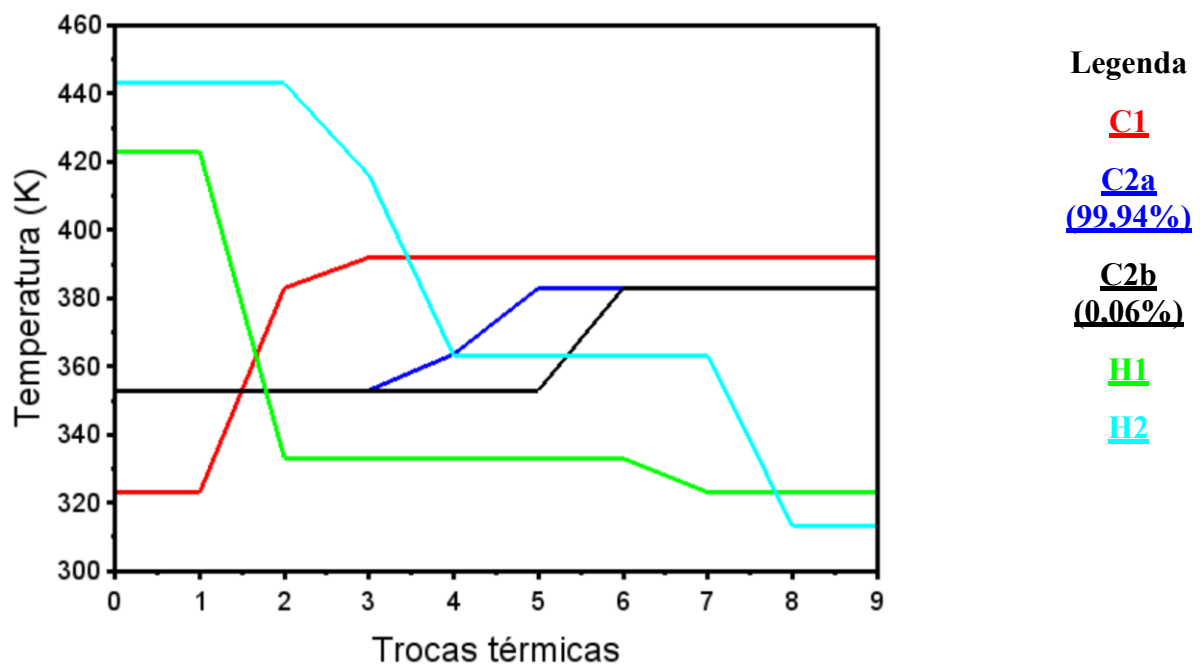


Figura 4.3 – Gráfico da variação das temperaturas das correntes com as trocas térmicas para o segundo caso, com divisão da corrente C2.

4.3 – PERFORMANCE DO PROGRAMA PARA A INDÚSTRIA DE SOJA

Após as validações dos códigos com os dois casos de estudo apresentados nas Seções 4.1 e 4.2, parte-se então para a aplicação do código para a otimização da indústria de beneficiamento de soja. A indústria escolhida para a otimização está descrita no trabalho de CERETO (2004).

O problema original descrito no trabalho de CERETO (2004) contém 21 correntes no total, sendo 9 correntes quentes e 12 correntes frias. Contudo, destas 21 correntes, 10 possuem calor específico dependente da temperatura. Visto que os algoritmos deste trabalho foram elaborados para otimizar correntes com calor específico constante, é necessário, neste caso, elaborar uma nova versão do código para trabalhar com correntes com calor específico dependente da temperatura.

Contudo, há um grupo de 5 correntes, com calor específico constante, que apenas interagem entre si e com as utilidades. Assim, este grupo de 5 correntes é aplicado ao código para comparação de resultados. A Tabela 4.7 mostra as correntes e suas características.

Tabela 4.7 – Dados do grupo de correntes da indústria de beneficiamento de soja.

Corrente	Descrição da Corrente	T_{entrada} (°C)	T_{saída} (°C)	mCp (kcal/(h.°C))
C4	Óleo para o Secador	70	90	4.241
C9	Óleo mineral para “Stripper”	55	90	5.597
H5	Óleo para Degomagem	100	70	4.238
H10	Óleo mineral para Absorção	90	40	5.170
H12	Óleo quente	90	40	4.051

Fonte: CERETO (2004).

CERETO (2004) também adota o uso de vapor 3 bar g e temperatura de 142°C como utilidade quente e água de resfriamento com temperatura de entrada de 25°C e temperatura de saída de 29°C como utilidade fria. Ainda, para a resolução do problema, considerou-se um $\Delta T_{\min}$ de 10°C.

As Equações 4.9 a 4.14, extraídas de CERETO (2004), fornecem, respectivamente, o custo de utilidade quente, o custo de utilidade fria, o custo dos trocadores de calor em função da área, o coeficiente global de troca térmica para o vapor, o coeficiente global de troca térmica para a água de resfriamento e o coeficiente global de troca térmica para as correntes do processo.

$$\text{Custo de } UQ = 23,5 \frac{\text{US\$}}{\text{Gcal}} \quad (4.9)$$

$$\text{Custo de } UF = 4,5 \frac{\text{US\$}}{\text{Gcal}} \quad (4.10)$$

$$\text{Custo do trocador (US\$)} = 5358,8 + 128,41A \quad (4.11)$$

$$U_{UQ} = 4185 \frac{\text{kcal}}{\text{h.m}^2.\text{°C}} \quad (4.12)$$

$$U_{UF} = 2108 \frac{\text{kcal}}{\text{h.m}^2.\text{°C}} \quad (4.13)$$

$$U_{\text{correntes de processo}} = 112 \frac{\text{kcal}}{\text{h.m}^2.\text{°C}} \quad (4.14)$$

Como o problema utiliza três coeficientes globais de troca térmica diferentes, a depender da corrente, foi necessária então uma adaptação ao código, para que o mesmo utilizasse os três coeficientes globais de troca térmica.

Visto que o interesse aqui é estudar as 5 correntes apresentadas na Tabela 4.7, a Figura 4.4 apresenta apenas a parte da rede de trocadores de calor do trabalho de CERETO (2004) composta por estas 5 correntes.

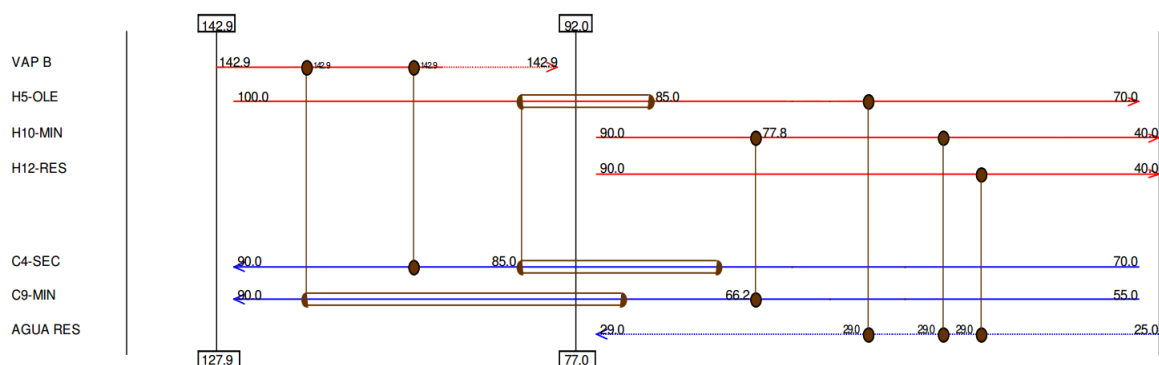


Figura 4.4 – Rede de trocadores de calor para um grupo de correntes da indústria de soja.

Fonte: adaptado de CERETO (2004).

É possível, através da Figura 4.4, calcular os valores de TAC, área, Q_Q e Q_F para o grupo de correntes selecionadas. Estes resultados são apresentados na Tabela 4.8.

Tabela 4.8 – Dados das redes de trocadores de calor da literatura para a indústria de beneficiamento de soja.

	TAC (US\$/ano)	Nº de Trocadores	Área (m ²)	Q_Q (kcal/h)	Q_F (kcal/h)
CERETO (2004) ^a	64.249	7	69,38	154.413,60	461.546,00

^a Solução recalculada pela autora.

Para elaboração da rede de trocadores de calor para o grupo de correntes, utilizou-se o código apresentado no Apêndice C. Este código é similar ao proposto no Apêndice A, porém adaptado para atender aos diferentes coeficientes globais de troca térmica e às variáveis e

equações deste problema. Neste caso, optou-se por não promover a divisão de nenhuma corrente, visto que as correntes possuem capacidade térmica muito próximas entre si. O código foi executado 10 vezes e o resultado selecionado foi o com menor TAC dentre as 10 execuções. A Tabela 4.9 contém os resultados obtidos na execução do código.

Tabela 4.9 – Dados da rede de trocadores de calor obtida para a indústria de beneficiamento de soja.

	TAC (US\$/ano)	Nº de Trocadores	Área (m²)	Q_Q (kcal/h)	Q_F (kcal/h)
Este trabalho	52.518	8	138,80	70.184,23	377.659,23

Pode-se observar que este trabalho propõe uma rede de trocadores com TAC 18,3% menor que o encontrado na literatura. Outro ponto é que este trabalho propõe um menor consumo de utilidades quentes e frias, mas para isso necessita de um trocador de calor a mais do que na literatura.

O esquema da rede de trocadores de calor proposta por este trabalho é apresentado na Figura 4.5.

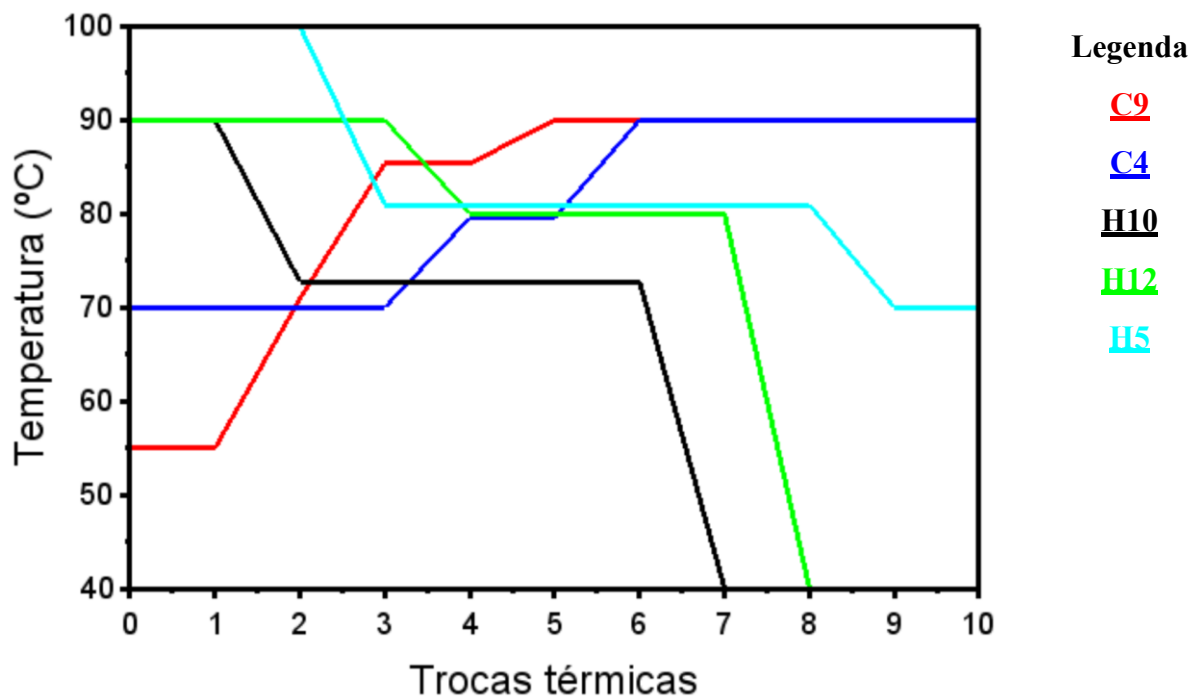


Figura 4.5 – Gráfico da variação das temperaturas das correntes com as trocas térmicas para a indústria de beneficiamento de soja.

Ao analisar a Figura 4.5 e comparar com a Figura 4.4, vê-se que em ambos os esquemas, todas as correntes quentes trocam calor com as utilidades frias e todas as correntes frias trocam calor com as utilidades quentes. A diferença entre os dois esquemas, e o que leva a este trabalho ter um trocador a mais que o da literatura, está no fato de que, na literatura, a corrente C9 recebe calor de H10 e a corrente C4 recebe calor de H5; portanto a corrente H12 somente trocava calor com a utilidade fria. Já neste trabalho, a corrente C9 recebe calor de H10 e H5 e a corrente C4 recebe calor de H12. Assim, é necessário um trocador de calor a mais, contudo isso diminui a necessidade de utilidades quentes e frias, visto que as correntes do processo trocaram mais calor entre si.

CAPÍTULO 5

CONCLUSÕES E SUGESTÕES

5.1 – CONCLUSÕES

Ao analisar todos os resultados obtidos no Capítulo 4, conclui-se que o Método do Enxame de Partículas gera malhas com valores da função objetivo próximas ao encontrado na literatura, sendo que, neste trabalho, para todos os casos estudados, a função objetivo reportada pelo código do PSO gerou respostas menores do que o reportado na literatura. Possíveis razões que justificam tal fato envolvem o uso de metodologias diferentes e de abordagem de área mínima para o trocador de calor, sendo que esta segunda razão não foi abordada neste trabalho. Logo, estes fatores podem ser responsáveis por gerar resultados diferentes, não só de TAC, como também de área de troca térmica, por exemplo.

5.2 – SUGESTÕES

É possível observar que alguns assuntos importantes não foram explorados neste trabalho e representam uma oportunidade para trabalhos futuros. Uma das possibilidades é fazer a implementação do código para otimizar correntes que possuam calor específico variável. Com esta melhoria, é possível fazer um novo estudo de caso com todas as correntes da indústria de soja. Além, outra possibilidade é a avaliação de diferentes metodologias de cálculo, seja para a função objetivo, seja com novas restrições no programa, como, por exemplo, uma restrição de área mínima de troca térmica ou diferentes formas de avaliar o impacto do custo dos trocadores de calor na função objetivo. Por fim, pode-se avaliar também a resposta de outros métodos heurísticos.

REFERÊNCIAS BIBLIOGRÁFICAS

- AGUITONI, M. C.; PAVÃO, L. V.; SIQUEIRA, P. H.; JIMÉNEZ, L.; RAVAGNANI, M. A. S. S. Heat exchanger network synthesis using genetic algorithm and differential evolution. Computers and Chemical Engineering 117, 2018. p. 82-96.
- AHMAD, S. Heat exchanger networks: cost tradeoffs in energy and capital. Manchester, University of Manchester, Institute of Science and Technology. 1985.
- ASSOCIAÇÃO BRASILEIRA DAS INDÚSTRIAS DE ÓLEOS VEGETAIS. História. Disponível em: <<http://www.abiove.org.br/site/index.php?page=historia&area=My0xLTU=>>>. Acesso em: 06 set. 2018.
- BRASIL. Empresa Brasileira de Pesquisa Agropecuária. Tecnologia para produção do óleo de soja: descrição das etapas, equipamentos, produtos e subprodutos. Londrina, 2001. 40p.
- CERETO, A. C. Integração Energética da Rede de Trocadores de Calor em Extração por Solvente para a Produção de Farelo Branco de Soja. Porto Alegre, UFRGS, 2004. 170 p.
- HUO, Z.; ZHAO, L.; YIN, H.; YE, J. A hybrid optimization strategy for simultaneous synthesis of heat exchanger network. Korean J. Chem. Eng. 29, 2012. p. 1298-2309.
- KENNEDY, J.; EBERHART, R. Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks IV, 1995. p. 1942–1948.
- KHORASANY, R. M.; FESANGHARY, M. A novel approach for synthesis of cost-optimal heat exchanger networks. Comput. Chem. Eng. 33, 2009. p. 1363-1370.
- NIELSEN, J. S.; HANSEN, M. W.; BAY JOERGENSEN, S. Heat exchanger network modelling framework for optimal design and retrofitting. Comput. Chem. Eng. 20, 1996. S249-S254.
- PAVÃO, L. V.; COSTA, C. B. B.; RAVAGNANI, M. A. S. S. Automated heat exchanger network synthesis by using hybrid natural algorithms and parallel processing. Comput. Chem. Eng. 94, 2016. p. 370-386.
- PAVÃO, L. V.; MIRANDA, C. B.; COSTA, C. B. B.; RAVAGNANI, M. A. S. S. Efficient multiperiod exchanger network synthesis using a meta-heuristic approach. Energy 142, 2018. p. 356-372.
- RAO, S. S. Engineering Optimization: Theory and Practice. 4. Ed. New Jersey, Hoboken, John Wiley & Sons, 2009. 813 p.
- RAVAGNANI, M. A. S. S.; SUÁREZ, J. A. C. Redes de Trocadores de Calor. Maringá, Eduem, 2012. 471 p.
- SILVA, A. P.; RAVAGNANI, M. A. S. S.; BISCAIA, E. C.; CABALLERO, J. A. Optimal heat exchanger network synthesis using particle swarm optimization. Optim. Eng. 11, 2010. p. 459-470.
- ZHU, X. X. Automated design method for heat exchanger network using block decomposition and heuristic rules. Comput. Chem. Eng. 21, 1997. p. 1095-1104.

APÊNDICE A – CÓDIGO-FONTE DO PRIMEIRO CASO DE ESTUDO

```

1  clc
2  clear
3
4  //DELTAS retorna a matriz MAX e MIN que são usadas para os cálculos
5  function [MIN,MAX]=DELTAS(tempAtual,tempAlvo)
6  ... cols=size(tempAtual,2)
7  ... a=0;numCols=0;
8  ...
9  ... for i=1:cols
10 ...     a=cols-i
11 ...     numCols=numCols+a
12 ... end
13 ...
14 ... numCols=2*numCols
15 ... MAX=zeros(numCols)
16 ... MIN=zeros(numCols)
17 ...
18 ... a=0;cont=0;
19 ... for i=1:cols
20 ...     a=cols-i
21 ...
22 ...     if a>0 then
23 ...         deltaT=tempAlvo(i)-tempAtual(i)
24 ...         for j=1:a
25 ...             cont=cont+1
26 ...             if deltaT>0
27 ...                 MAX(cont)=deltaT
28 ...                 MIN(cont)=0
29 ...             else deltaT<0
30 ...                 MAX(cont)=-deltaT
31 ...                 MIN(cont)=0
32 ...             end
33 ...         end
34 ...     end
35 ...     //para garantir convergência, modificamos MAX e MIN
36 ...     MAX(cont)=3*MAX(cont)
37 ...     MIN(cont)=-1*MAX(cont)
38 ... end
39 ... end
40 ...
41 ... for i=(cont+1):numCols
42 ...     MAX(i)=1
43 ...     MIN(i)=0
44 ... end
45 ...
46 endfunction
47
48 //Enxame retorna a população aleatória com todos valores entre MIN e MAX
49 function x=Enxame(numParametros,numParticulas,deltaMax,deltaMin)
50 ... x=zeros(numParticulas,numParametros)
51 ... for j=1:numParametros
52 ...     for i=1:numParticulas
53 ...         x(i,j)=deltaMin(j)+(deltaMax(j)-deltaMin(j))*rand()
54 ...     end
55 ... end
56 endfunction
57
58 //FOBJ calcula o valor da função a ser minimizada para cada partícula
59 function fobj=FOBJ(x,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
60 ... rows=size(x,1)
61 ... fobj=zeros(rows)
62 ... for i=1:rows
63 ...     fobj(i)=FUN(x(i,:),mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
64 ... end
65 endfunction
66
67 //FUN calcula o TAC para um determinado arranjo
68 function fun=FUN(x,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
69 ... fun=0
70 ...

```

```

71 --- cols=size(mCp,2)
72 --- matrizDeltaT=zeros(cols,cols)
73 --- contador=1
74 --- for i=1:cols
75 ---     for j=1:cols
76 ---         if i<j
77 ---             matrizDeltaT(i,j)=x(contador)
78 ---             contador=contador+1
79 ---         end
80 ---     end
81 --- end
82 ---
83 --- matrizBinaria=zeros(cols,cols)
84 --- for i=1:cols
85 ---     for j=1:cols
86 ---         if i<j
87 ---             matrizBinaria(i,j)=round(x(contador))
88 ---             contador=contador+1
89 ---         end
90 ---     end
91 --- end
92 ---
93 --- areaTrocador=0; tempSaidaQ=0; tempSaidaF=0; deltaTMediaLog=0;
94 --- for i=1:cols
95 ---     for j=1:cols
96 ---         if i<j
97 ---             if matrizBinaria(i,j)==1
98 ---                 if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
99 ---                     //i: CQ, j: CF
100 ---                     cargaTermica=mCp(i)*matrizDeltaT(i,j)
101 ---                     tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
102 ---                     tempSaidaF=-cargaTermica/(mCp(j))+tempAtual(j)
103 ---                     if (tempAtual(i)-tempAtual(j)>alfa)&(tempSaidaQ-tempAtual(j)>alfa)&(tempAtual(i)-temp
SaidaF>alfa)
104 ---                         //TQentra-TFentra>alfa & TQsai-TFentra>alfa & TQentra-TFsai>alfa
105 ---                         deltaTMediaLog=((tempSaidaQ-tempAtual(j))-(tempAtual(i)-tempSaidaF))/log((tempSaid
aQ-tempAtual(j))/(tempAtual(i)-tempSaidaF))
106 ---                         areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
107 ---                         tempAtual(i)=tempSaidaQ
108 ---                         tempAtual(j)=tempSaidaF
109 ---                     end
110 ---                 else //a corrente i ganha calor da j
111 ---                     //i: CF, j: CQ
112 ---                     cargaTermica=mCp(i)*matrizDeltaT(i,j)
113 ---                     tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
114 ---                     tempSaidaQ=-cargaTermica/(mCp(j))+tempAtual(j)
115 ---                     if (tempAtual(j)-tempAtual(i)>alfa)&(tempAtual(j)-tempSaidaF>alfa)&(tempSaidaQ-tempAt
ual(i)>alfa)
116 ---                         //TQentra-TFentra>alfa & TQentra-TFsai>alfa & TQsai-TFentra>alfa
117 ---                         deltaTMediaLog=((tempSaidaQ-tempAtual(i))-(tempAtual(j)-tempSaidaF))/log((tempSai
daQ-tempAtual(i))/(tempAtual(j)-tempSaidaF))
118 ---                         areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
119 ---                         tempAtual(i)=tempSaidaF
120 ---                         tempAtual(j)=tempSaidaQ
121 ---                     end
122 ---                 end
123 ---             end
124 ---         end
125 ---     end
126 --- end
127 ---
128 --- cargaUQ=0; cargaUF=0;
129 --- for i=1:cols
130 ---     deltaT=tempAlvo(i)-tempAtual(i)
131 ---     if deltaT>beta //usa UQ
132 ---         cargaUQ=cargaUQ+mCp(i)*deltaT
133 ---         //UQ -> vapor saturado -> entra a 250°C
134 ---         deltaTMediaLog=((523.15-tempAtual(i))-(523.15-tempAlvo(i)))/log((523.15-tempAtual(i))/(523.15-tem
pAlvo(i)))
135 ---         areaTrocador=areaTrocador+abs(mCp(i)*(deltaT)/(U*deltaTMediaLog))
136 ---     elseif deltaT<(-beta) //usa UF
137 ---         cargaUF=cargaUF+mCp(i)*(-deltaT)
138 ---         //UF -> água resfriada -> entra a 25°C e sai a 30°C
139 ---         deltaTMediaLog=((tempAlvo(i)-298.15)-(tempAtual(i)-303.15))/log((tempAlvo(i)-298.15)/(tempAtual(i)
-303.15))
140 ---         areaTrocador=areaTrocador+abs(mCp(i)*(-deltaT)/(U*deltaTMediaLog))

```

```

141 ..... end
142 ... end
143 ...
144 costTrocador=300*(areaTrocador)^0.5
145 costUQ=coefCustoUQ*cargaUQ
146 costUF=coefCustoUF*cargaUF
147 fun=costUQ+costUF+costTrocador
148 endfunction
149
150 //MATBEST retorna a matriz com os deltas das trocas térmicas que ocorreram e a matriz binária
151 function [matrizTrocaBest,matrizBinaria]=MATBEST(x,mCp,tempAtual)
152 ... cols=size(mCp,2)
153 ... matrizDeltaT=zeros(cols,cols)
154 ... matrizTrocaBest=zeros(cols,cols)
155 ... contador=1
156 ... for i=1:cols
157 ..... for j=1:cols
158 ..... if i<j
159 ..... matrizDeltaT(i,j)=x(contador)
160 ..... contador=contador+1
161 ..... end
162 ..... end
163 ... end
164 ...
165 ... matrizBinaria=zeros(cols,cols)
166 ... for i=1:cols
167 ..... for j=1:cols
168 ..... if i<j
169 ..... matrizBinaria(i,j)=round(x(contador))
170 ..... contador=contador+1
171 ..... end
172 ..... end
173 ... end
174 ...
175 ... tempSaidaQ=0;tempSaidaF=0;cargaTermica=0
176 ... for i=1:cols
177 ..... for j=1:cols
178 ..... if i<j
179 ..... if matrizBinaria(i,j)==1
180 ..... if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
181 ..... //i:-CQ,-j:-CF
182 ..... cargaTermica=mCp(i)*matrizDeltaT(i,j)
183 ..... tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
184 ..... tempSaidaF=-cargaTermica/(mCp(j))+tempAtual(j)
185 ..... if (tempAtual(i)-tempAtual(j)>alfa)&(tempSaidaQ-tempAtual(j)>alfa)&(tempAtual(i)-temp
186 ..... SaidaF>alfa)
187 ..... //TQentra-TFentra>alfa & TQsai-TFentra>alfa & TQentra-TFsai>alfa
188 ..... tempAtual(i)=tempSaidaQ
189 ..... tempAtual(j)=tempSaidaF
190 ..... else matrizDeltaT(i,j)=0
191 ..... end
192 ..... else //a corrente i ganha calor da j
193 ..... //i:-CF,-j:-CQ
194 ..... cargaTermica=mCp(i)*matrizDeltaT(i,j)
195 ..... tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
196 ..... tempSaidaQ=-cargaTermica/(mCp(j))+tempAtual(j)
197 ..... if (tempAtual(j)-tempAtual(i)>alfa)&(tempAtual(j)-tempSaidaF>alfa)&(tempSaidaQ-tempAt
198 ..... ual(i)>alfa)
199 ..... //TQentra-TFentra>alfa & TQentra-TFsai>alfa & TQsai-TFentra>alfa
200 ..... tempAtual(i)=tempSaidaF
201 ..... tempAtual(j)=tempSaidaQ
202 ..... else matrizDeltaT(i,j)=0
203 ..... end
204 ..... end
205 ..... end
206 ..... end
207 ... end
208 ...
209 matrizTrocaBest=matrizDeltaT
210 endfunction
211
212 //MATGRAF retorna o gráfico das trocas térmicas em cada trocador
213 function matrizGrafico=MATGRAF(matrizDeltaT,mCp,tempAtual,tempAlvo)
214 ... cols=size(mCp,2)

```

```

215 contador=2
216 matrizGrafico(:,1)=tempAtual
217 for i=1:cols
218     for j=1:cols
219         if i<j
220             if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
221                 //i: CQ, j: CF
222                 cargaTermica=mCp(i)*matrizDeltaT(i,j)
223                 tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
224                 tempSaidaF=-cargaTermica/(mCp(j))+tempAtual(j)
225                 if (tempAtual(i)-tempAtual(j)>alfa)&(tempSaidaQ-tempAtual(j)>alfa)&(tempAtual(i)-tempSaidaF>alfa)
226                     //TQentra-TFentra>alfa & TQsai-TFentra>alfa & TQentra-TFsai>alfa
227                     tempAtual(i)=tempSaidaQ
228                     tempAtual(j)=tempSaidaF
229
230                 matrizGrafico(:,contador)=tempAtual
231                 contador=contador+1
232             end
233             elseif matrizDeltaT(i,j)>0 //a corrente i ganha calor da j
234                 //i: CF, j: CQ
235                 cargaTermica=mCp(i)*matrizDeltaT(i,j)
236                 tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
237                 tempSaidaQ=-cargaTermica/(mCp(j))+tempAtual(j)
238                 if (tempAtual(j)-tempAtual(i)>alfa)&(tempAtual(j)-tempSaidaF>alfa)&(tempSaidaQ-tempAtual(i)>alfa)
239                     //TQentra-TFentra>alfa & TQentra-TFsai>alfa & TQsai-TFentra>alfa
240                     tempAtual(i)=tempSaidaF
241                     tempAtual(j)=tempSaidaQ
242
243                 matrizGrafico(:,contador)=tempAtual
244                 contador=contador+1
245             end
246         end
247     end
248 end
249 end
250
251 for i=1:cols
252     if abs(tempAtual(i)-tempAlvo(i))>betha
253         tempAtual(i)=tempAlvo(i)
254         matrizGrafico(:,contador)=tempAtual
255         contador=contador+1
256     end
257 end
258 endfunction
259
260 //PART atualiza PBEST (melhor de cada particula)
261 function PBEST=PART(x,xnew,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
262 rows=size(x,1)
263 cols=size(x,2)
264 YCAL=FOBJ(x,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
265 NEW=FOBJ(xnew,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF) //xnew vem da rotina velocidade
266 for i=1:rows
267     if(YCAL(i)>NEW(i))
268         x(i,:)=xnew(i,:)
269     end
270 end
271 PBEST=x
272 endfunction
273
274 //RESULT calcula o TAC, área e cargas de UQ e UF para um determinado arranjo
275 function [TAC,areaTrocador,cargaUQ,cargaUF]=RESULT(x,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
276 cols=size(mCp,2)
277 matrizDeltaT=zeros(cols,cols)
278 contador=1
279 for i=1:cols
280     for j=1:cols
281         if i<j
282             matrizDeltaT(i,j)=x(contador)
283             contador=contador+1
284         end
285     end
286 end
287
288 matrizBinaria=zeros(cols,cols)
289 for i=1:cols

```

```

290 ..... for j=1:cols
291 ..... if i<j
292 .....     matrizBinaria(i,j)=round(x(contador))
293 .....     contador=contador+1
294 ..... end
295 ..... end
296 ..... end
297 .....
298 .... TAC=0; areaTrocador=0; tempSaidaQ=0; tempSaidaF=0; deltaTMediaLog=0;
299 .... for i=1:cols
300 ..... for j=1:cols
301 ..... if i<j
302 .....     if matrizBinaria(i,j)==1
303 .....         if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
304 .....             //i: CQ, j: CF
305 .....             cargaTermica=mCp(i)*matrizDeltaT(i,j)
306 .....             tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
307 .....             tempSaidaF=-cargaTermica/(mCp(j))+tempAtual(j)
308 .....             if (tempAtual(i)-tempAtual(j)>alfa) & (tempSaidaQ-tempAtual(j)>alfa) & (tempAtual(i)-temp
SaidaF>alfa)
309 .....                 //TQentra-TFentra>alfa & TQsai-TFentra>alfa & TQentra-TFsai>alfa
310 .....                 deltaTMediaLog=((tempSaidaQ-tempAtual(j))-(tempAtual(i)-tempSaidaF))/log((tempSaid
aQ-tempAtual(j))/(tempAtual(i)-tempSaidaF))
311 .....                 areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
312 .....                 tempAtual(i)=tempSaidaQ
313 .....                 tempAtual(j)=tempSaidaF
314 .....             end
315 .....             else //a corrente i ganha calor da j
316 .....                 //i: CF, j: CQ
317 .....                 cargaTermica=mCp(i)*matrizDeltaT(i,j)
318 .....                 tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
319 .....                 tempSaidaQ=-cargaTermica/(mCp(j))+tempAtual(j)
320 .....                 if (tempAtual(j)-tempAtual(i)>alfa) & (tempAtual(j)-tempSaidaF>alfa) & (tempSaidaQ-tempAt
ual(i)>alfa)
321 .....                     //TQentra-TFentra>alfa & TQentra-TFsai>alfa & TQsai-TFentra>alfa
322 .....                     deltaTMediaLog=((tempSaidaQ-tempAtual(i))-(tempAtual(j)-tempSaidaF))/log((tempSai
daQ-tempAtual(i))/(tempAtual(j)-tempSaidaF))
323 .....                     areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
324 .....                     tempAtual(i)=tempSaidaF
325 .....                     tempAtual(j)=tempSaidaQ
326 .....                 end
327 .....             end
328 .....         end
329 .....     end
330 ..... end
331 ..... end
332 .....
333 .... cargaUQ=0; cargaUF=0;
334 .... for i=1:cols
335 ..... deltaT=tempAlvo(i)-tempAtual(i)
336 ..... if deltaT>beta //usa UQ
337 .....     cargaUQ=cargaUQ+mCp(i)*deltaT
338 .....     //UQ -> vapor saturado -> entra a 250°C
339 .....     deltaTMediaLog=((523.15-tempAtual(i))-(523.15-tempAlvo(i)))/log((523.15-tempAtual(i))/(523.15-tem
pAlvo(i)))
340 .....     areaTrocador=areaTrocador+abs(mCp(i)*(deltaT)/(U*deltaTMediaLog))
341 ..... elseif deltaT<(-beta) //usa UF
342 .....     cargaUF=cargaUF+mCp(i)*(-deltaT)
343 .....     //UF -> água resfriada -> entra a 25°C e sai a 30°C
344 .....     deltaTMediaLog=((tempAlvo(i)-298.15)-(tempAtual(i)-303.15))/log((tempAlvo(i)-298.15)/(tempAtual(i)
-303.15))
345 .....     areaTrocador=areaTrocador+abs(mCp(i)*(-deltaT)/(U*deltaTMediaLog))
346 ..... end
347 ..... end
348 .....
349 .... costTrocador=300*(areaTrocador)^0.5
350 .... costUQ=coefCustoUQ*cargaUQ
351 .... costUF=coefCustoUF*cargaUF
352 .... TAC=costUQ+costUF+costTrocador
353 endfunction
354
355 //VE calcula a velocidade para cada partícula
356 function [X, VEL]=VE(X,BEST,PBEST,W,C1,C2,MAX,MIN)
357 rows=size(X,1)
358 cols=size(X,2)
359 MAX=MAX(:)

```

```

360 -- MIN=MIN(:)
361 -- VEL=zeros(rows, cols)
362 -- for i=1:rows
363 --     for j=1:cols
364 --         R1=rand()
365 --         R2=rand()
366 --         VELOCIDADE=W*VEL(i,j)+C1*R1*(PBEST(i,j)-X(i,j))+C2*R2*(BEST(j)-X(i,j))
367 --         if(VELOCIDADE> MAX(j))
368 --             VELOCIDADE=0.2*MAX(j)
369 --         end
370 --         if(VELOCIDADE<MIN(j))
371 --             VELOCIDADE=0.2*MIN(j)
372 --         end
373 --         VEL(i,j)=VELOCIDADE
374 --         X(i,j)=X(i,j)+VEL(i,j)
375 --         if(X(i,j)>MAX(j))
376 --             X(i,j)=MAX(j)
377 --         end
378 --         if(X(i,j)<MIN(j))
379 --             X(i,j)=MIN(j)
380 --         end
381 --     end
382 -- end
383 endfunction
384 function [BEST,GBEST,Vetor]=PSO(W,C1,C2,NPAR,ITE,PAR,MAX,MIN,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
385 )
386 -- Vetor=0
387 -- x=Enxame(PAR,NPAR,MAX,MIN) //gera população inicial aleatória
388 -- PBEST=x //inicialmente as melhores partículas são aleatórias
389 -- YCAL=FOBJ(x,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF) //estima a função nos valores aleatórios
390 -- [valor,best]=min(YCAL)
391 -- GBEST=PBEST(best,:) //melhor global na 1ª interação
392 -- BEST=YCAL(best)
393 -- VEL=Enxame(PAR,NPAR,MAX,MIN) //velocidades aleatórias na 1ª interação
394
395 -- for i=1:ITE
396 --     [xnew,VEL]=VE(x,GBEST,PBEST,W,C1,C2,MAX,MIN) //nova atualização de x utilizando as velocidades...
397 --     PBEST=PART(PBEST,xnew,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF) //atualização de PBEST (sempre
398 --     atualiza com a nova busca)
399 --     YCAL=FOBJ(PBEST,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
400 --     [valor,new]=min(YCAL)
401 --     NEW=YCAL(new)
402 --     if(NEW<BEST)
403 --         GBEST=xnew(new,:)
404 --         BEST=NEW
405 --     end
406 --     Vetor(i)=BEST;
407 --     x=xnew
408 -- end
409 endfunction
410
411 //*****PROGRAMA PRINCIPAL*****
412
413
414 W=0.9
415 C1=2
416 C2=2
417 NPAR=100 //Número de partículas
418 ITE=300 //Número de iterações
419 alfa=10
420 betha=1
421
422 //Ordem: C2.C1.H1.H2
423 mCp=[4 2 3 1.5] //[kW/K]
424 tempAtual=[453 393 533 523] //[K]
425 tempAlvo=[513 508 433 403] //[K]
426 U=0.2 //[kW/(m2.K)]
427 coefCustoUQ=110 //[$/(kW.ano)]
428 coefCustoUF=12.2 //[$/(kW.ano)]
429
430 //Cálculo do número de parâmetros
431 somatorio=0
432 for i=1:(size(mCp,2)-1)
433 -- somatorio=somatorio+i

```

```

434 end
435 PAR=2*somatorio //Número de parâmetros a ser otimizados
436
437 //cálculo de MÁX e MIN
438 [MIN,MAX]=DELTAS(tempAtual,tempAlvo)
439
440 [BEST,GBEST,Vetor]=PSO(W,C1,C2,NPAR,ITE,PAR,MAX,MIN,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
441 Iter=1:ITE;
442
443 [matrizTrocaBest,matrizBinaria]=MATBEST(GBEST,mCp,tempAtual)
444 matrizGrafico=MATGRAF(matrizTrocaBest,mCp,tempAtual,tempAlvo)
445
446 printf('Melhor solução para %d iterações com %d partículas\n',ITE,NPAR); disp(GBEST)
447 printf('Valor da função f(gbest)=%f\n',BEST)
448
449 printf('Matriz de Trocas Térmicas\n')
450 disp(matrizTrocaBest)
451 printf('Matriz Binária\n')
452 disp(matrizBinaria)
453
454
455 cols=size(matrizGrafico,2)
456 lins=size(matrizGrafico,1)
457
458 y=zeros(lins,cols+2);
459 y(:,1)=matrizGrafico(:,1);
460 y(:,cols+2)=matrizGrafico(:,cols);
461
462 for j=1:cols
463     y(:,j+1)=matrizGrafico(:,j);
464 end
465
466 x=0:(cols+1)
467
468 a=get('current_axes') //Identificador dos eixos criados
469 a.axes_visible='on'; //Torna os eixos visíveis
470 a.font_size=3; //Define o tamanho da fonte
471 a.sub_tics=[1,50];
472 a.thickness=3;
473 a.font_size=4;
474
475 Cor=['red','blue','black','green','cyan','magenta','blackread'];
476
477 scf(0);
478 for i=1:lins
479     plot(x,y(i,:),Cor(i));
480 end
481
482 xlabel("Trocas térmicas","fontsize", 5)
483 ylabel("Temperatura (K)","fontsize", 5)
484
485 TAC=0;areaTrocador=0;cargaUQ=0;cargaUF=0;
486 [TAC,areaTrocador,cargaUQ,cargaUF]=RESULT(GBEST,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
487
488 printf('TESTE ADICIONAL\n')
489 printf('Custo Total Anual=%f\n',TAC)
490 printf('Area de Trocador=%f m2\n',areaTrocador)
491 printf('Carga de UQ=%f kW\n',cargaUQ)
492 printf('Carga de UF=%f kW\n',cargaUF)
493
494 //plot(Iter,Vetor,'black')

```

APÊNDICE B – CÓDIGO-FONTE DO SEGUNDO CASO DE ESTUDO, COM DIVISÃO DE UMA CORRENTE

```

1  clc
2  clear
3
4  //DELTAS: retorna a matriz MAX e MIN que são usadas para os cálculos
5  function [MIN,MAX]=DELTAS(tempAtual,tempAlvo)
6  cols=size(tempAtual,2)
7  a=0; numCols=0;
8
9  for i=1:cols
10     a=cols-i
11     numCols=numCols+a
12 end
13
14 numCols=2*numCols+1
15 MAX=zeros(numCols)
16 MIN=zeros(numCols)
17
18 a=0; cont=0;
19 for i=1:cols
20     a=cols-i
21
22     if a>0 then
23         deltaT=tempAlvo(i)-tempAtual(i)
24         for j=1:a
25             cont=cont+1
26             if deltaT>0
27                 MAX(cont)=deltaT
28                 MIN(cont)=0
29             else deltaT<0
30                 MAX(cont)=-deltaT
31                 MIN(cont)=0
32             end
33         end
34         //para garantir convergência, modificamos MAX e MIN
35         MAX(cont)=3*MAX(cont)
36         MIN(cont)=-1*MAX(cont)
37     end
38 end
39 end
40
41 for i=(cont+1):numCols
42     MAX(i)=1
43     MIN(i)=0
44 end
45
46 endfunction
47
48 //Enxame retorna a população aleatória com todos valores entre MIN e MAX
49 function x=Enxame(numParametros,numParticulas,deltaMax,deltaMin)
50 x=zeros(numParticulas, numParametros)
51 for j=1:numParametros
52     for i=1:numParticulas
53         x(i,j)=deltaMin(j)+(deltaMax(j)-deltaMin(j))*rand()
54     end
55 end
56 endfunction
57
58 //FOBJ calcula o valor da função a ser minimizada para cada partícula
59 function fobj=FOBJ(x,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
60 rows=size(x,1)
61 fobj=zeros(rows)
62 for i=1:rows
63     fobj(i)=FUN(x(i,:),mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
64 end
65 endfunction
66
67 //FUN calcula o TAC para um determinado arranjo
68 function fun=FUN(x,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
69 fun=0
70

```

```

71  cols=size(mCp,2)
72  matrizDeltaT=zeros(cols,cols)
73  contador=1
74  for i=1:cols
75      for j=1:cols
76          if i<j
77              matrizDeltaT(i,j)=x(contador)
78              contador=contador+1
79          end
80      end
81  end
82  ---
83  matrizBinaria=zeros(cols,cols)
84  for i=1:cols
85      for j=1:cols
86          if i<j
87              matrizBinaria(i,j)=round(x(contador))
88              contador=contador+1
89          end
90      end
91  end
92  ---
93  //Fator para corrigir o mCp or conta da divisão de uma corrente
94  vetor_mCp=zeros(size(mCp,2))
95  for i=1:size(mCp,2)
96      vetor_mCp(i)=1
97  end
98  vetor_mCp(posicaoCorrente)=x(contador)
99  vetor_mCp(posicaoCorrente+1)=(1-x(contador))
100 ---
101 ---
102 areaTrocador=0; tempSaidaQ=0; tempSaidaF=0; deltaTMediaLog=0;
103 for i=1:cols
104     for j=1:cols
105         if i<j
106             if matrizBinaria(i,j)==1
107                 if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
108                     //i: CQ, j: CF
109                     cargaTermica=mCp(i)*vetor_mCp(i)*matrizDeltaT(i,j)
110                     tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
111                     tempSaidaF=-cargaTermica/(mCp(j)*vetor_mCp(j))+tempAtual(j)
112                     if (tempAtual(i)-tempAtual(j)>alfa) & (tempSaidaQ-tempAtual(j)>alfa) & (tempAtual(i)-temp
SaidaF>alfa)
113                         //TQentra-TFentra>alfa & TQsai-TFentra>alfa & TQentra-TFsai>alfa
114                         deltaTMediaLog=((tempSaidaQ-tempAtual(j))-(tempAtual(i)-tempSaidaF))/log((tempSaid
aQ-tempAtual(j))/(tempAtual(i)-tempSaidaF))
115                         areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
116                         tempAtual(i)=tempSaidaQ
117                         tempAtual(j)=tempSaidaF
118                     end
119                 else //a corrente i ganha calor da j
120                     //i: CF, j: CQ
121                     cargaTermica=mCp(i)*vetor_mCp(i)*matrizDeltaT(i,j)
122                     tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
123                     tempSaidaQ=-cargaTermica/(mCp(j)*vetor_mCp(j))+tempAtual(j)
124                     if (tempAtual(j)-tempAtual(i)>alfa) & (tempAtual(j)-tempSaidaF>alfa) & (tempSaidaQ-tempAt
ual(i)>alfa)
125                         //TQentra-TFentra>alfa & TQentra-TFsai>alfa & TQsai-TFentra>alfa
126                         deltaTMediaLog=((tempSaidaQ-tempAtual(i))-(tempAtual(j)-tempSaidaF))/log((tempSai
daQ-tempAtual(i))/(tempAtual(j)-tempSaidaF))
127                         areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
128                         tempAtual(i)=tempSaidaF
129                         tempAtual(j)=tempSaidaQ
130                     end
131                 end
132             end
133         end
134     end
135 end
136 ---
137 cargaUQ=0; cargaUF=0;
138 for i=1:cols
139     deltaT=tempAlvo(i)-tempAtual(i)
140     if deltaT>beta //usa UQ
141         cargaUQ=cargaUQ+mCp(i)*vetor_mCp(i)*deltaT
142         //UQ -> vapor saturado -> entra e sai a 142,9°C

```

```

143 .....deltaTMediaLog=((416.05-tempAtual(i))-(416.05-tempAlvo(i)))/log((416.05-tempAtual(i))/(416.05-temp
pAlvo(i)))
144 .....areaTrocador=areaTrocador+abs(mCp(i)*vetor_mCp(i)*(deltaT)/(U*deltaTMediaLog))
145 .....elseif deltaT<(-betha) //usa UF
146 .....cargaUF=cargaUF+mCp(i)*vetor_mCp(i)*(-deltaT)
147 .....//UF-> água resfriada-> entra a 25°C e sai a 30°C
148 .....deltaTMediaLog=((tempAlvo(i)-298.15)-(tempAtual(i)-303.15))/log((tempAlvo(i)-298.15)/(tempAtual(i)
)-303.15))
149 .....areaTrocador=areaTrocador+abs(mCp(i)*vetor_mCp(i)*(-deltaT)/(U*deltaTMediaLog))
150 .....end
151 --- end
152 ---
153 --- costTrocador=9094+221*(areaTrocador)^0.81
154 --- costUQ=coefCustoUQ*cargaUQ
155 --- costUF=coefCustoUF*cargaUF
156 --- fun=costUQ+costUF+costTrocador
157 endfunction
158
159 //MATBEST retorna a matriz com os deltas das trocas térmicas que ocorreram e a matriz binária
160 function [matrizTrocaBest,matrizBinaria]=MATBEST(x,mCp,tempAtual)
161 --- cols=size(mCp,2)
162 --- matrizDeltaT=zeros(cols,cols)
163 --- matrizTrocaBest=zeros(cols,cols)
164 --- contador=1
165 --- for i=1:cols
166 ..... for j=1:cols
167 ..... if i<j
168 ..... matrizDeltaT(i,j)=x(contador)
169 ..... contador=contador+1
170 ..... end
171 ..... end
172 --- end
173 ---
174 --- matrizBinaria=zeros(cols,cols)
175 --- for i=1:cols
176 ..... for j=1:cols
177 ..... if i<j
178 ..... matrizBinaria(i,j)=round(x(contador))
179 ..... contador=contador+1
180 ..... end
181 ..... end
182 --- end
183 ---
184 --- tempSaidaQ=0;tempSaidaF=0;cargaTermica=0
185 --- for i=1:cols
186 ..... for j=1:cols
187 ..... if i<j
188 ..... if matrizBinaria(i,j)==1
189 ..... if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
190 ..... //i: CQ, j: CF
191 ..... cargaTermica=mCp(i)*matrizDeltaT(i,j)
192 ..... tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
193 ..... tempSaidaF=-cargaTermica/(mCp(j))+tempAtual(j)
194 ..... if (tempAtual(i)-tempAtual(j)>alfa)&(tempSaidaQ-tempAtual(j)>alfa)&(tempAtual(i)-temp
SaidaF>alfa)
195 ..... //TQentra-TFentra>alfa & TQsai-TFentra>alfa & TQentra-TFsai>alfa
196 ..... tempAtual(i)=tempSaidaQ
197 ..... tempAtual(j)=tempSaidaF
198 ..... else matrizDeltaT(i,j)=0
199 ..... end
200 ..... else //a corrente i ganha calor da j
201 ..... //i: CF, j: CQ
202 ..... cargaTermica=mCp(i)*matrizDeltaT(i,j)
203 ..... tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
204 ..... tempSaidaQ=-cargaTermica/(mCp(j))+tempAtual(j)
205 ..... if (tempAtual(j)-tempAtual(i)>alfa)&(tempAtual(j)-tempSaidaF>alfa)&(tempSaidaQ-tempAt
ual(i)>alfa)
206 ..... //TQentra-TFentra>alfa & TQentra-TFsai>alfa & TQsai-TFentra>alfa
207 ..... tempAtual(i)=tempSaidaF
208 ..... tempAtual(j)=tempSaidaQ
209 ..... else matrizDeltaT(i,j)=0
210 ..... end
211 ..... end
212 ..... else matrizDeltaT(i,j)=0
213 ..... end
214 ..... end
215 ..... end

```

```

216     end
217     ...
218     matrizTrocaBest=matrizDeltaT
219 endfunction
220
221 //MATGRAF retorna o gráfico das trocas térmicas em cada trocador
222 function matrizGrafico=MATGRAF(matrizDeltaT,mCp,tempAtual,tempAlvo)
223     cols=size(mCp,2)
224     contador=2
225     matrizGrafico(:,1)=tempAtual
226     for i=1:cols
227         for j=1:cols
228             if i<j
229                 if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
230                     //i: CQ, j: CF
231                     cargaTermica=mCp(i)*matrizDeltaT(i,j)
232                     tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
233                     tempSaidaF=-cargaTermica/(mCp(j))+tempAtual(j)
234                     if (tempAtual(i)-tempAtual(j)>alfa)&(tempSaidaQ-tempAtual(j)>alfa)&(tempAtual(i)-tempSaid
aF>alfa)
235                         //TQentra-TFentra>alfa & TQsai-TFentra>alfa & TQentra-TFsai>alfa
236                         tempAtual(i)=tempSaidaQ
237                         tempAtual(j)=tempSaidaF
238
239                         matrizGrafico(:,contador)=tempAtual
240                         contador=contador+1
241                     end
242                 elseif matrizDeltaT(i,j)>0 //a corrente i ganha calor da j
243                     //i: CF, j: CQ
244                     cargaTermica=mCp(i)*matrizDeltaT(i,j)
245                     tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
246                     tempSaidaQ=-cargaTermica/(mCp(j))+tempAtual(j)
247                     if (tempAtual(j)-tempAtual(i)>alfa)&(tempAtual(j)-tempSaidaF>alfa)&(tempSaidaQ-tempAtual(
i)>alfa)
248                         //TQentra-TFentra>alfa & TQentra-TFsai>alfa & TQsai-TFentra>alfa
249                         tempAtual(i)=tempSaidaF
250                         tempAtual(j)=tempSaidaQ
251
252                         matrizGrafico(:,contador)=tempAtual
253                         contador=contador+1
254                     end
255                 end
256             end
257         end
258     end
259
260     for i=1:cols
261         if abs(tempAtual(i)-tempAlvo(i))>betha
262             tempAtual(i)=tempAlvo(i)
263             matrizGrafico(:,contador)=tempAtual
264             contador=contador+1
265         end
266     end
267 endfunction
268
269 //PART atualiza PBEST (melhor de cada partícula)
270 function PBEST=PART(x,xnew,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
271     rows=size(x,1)
272     cols=size(x,2)
273     YCAL=FORJ(x,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
274     NEW=FORJ(xnew,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF) //xnew vem da rotina velocidade
275     for i=1:rows
276         if (YCAL(i)>NEW(i))
277             x(i,:)=xnew(i,:)
278         end
279     end
280     PBEST=x
281 endfunction
282
283 //PARTICAO define as novas matrizes quando ocorre divisão de uma corrente
284 function [mCpNovo,tempAtualNovo,tempAlvoNovo]=PARTICAO(mCp,tempAtual,tempAlvo,posicaoCorrente)
285     cols=size(mCp,2)
286     mCpNovo=zeros(mCp)
287     tempAtualNovo=zeros(tempAtual)
288     tempAlvoNovo=zeros(tempAlvo)
289
290     for i=1:posicaoCorrente

```

```

291 ..... tempAtualNovo(i)=tempAtual(i)
292 ..... tempAlvoNovo(i)=tempAlvo(i)
293 ..... mCpNovo(i)=mCp(i)
294 ..... end
295 .....
296 ..... for i=posicaoCorrente:cols
297 ..... tempAtualNovo(i+1)=tempAtual(i)
298 ..... tempAlvoNovo(i+1)=tempAlvo(i)
299 ..... mCpNovo(i+1)=mCp(i)
300 ..... end
301 .....
302 endfunction
303 .....
304 //RESULT calcula o TAC, área e cargas de UQ e UF para um determinado arranjo
305 function [TAC,areaTrocador,cargaUQ,cargaUF]=RESULT(x,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
306 ..... cols=size(mCp,2)
307 ..... matrizDeltaT=zeros(cols,cols)
308 ..... contador=1
309 ..... for i=1:cols
310 ..... for j=1:cols
311 ..... if i<j
312 ..... matrizDeltaT(i,j)=x(contador)
313 ..... contador=contador+1
314 ..... end
315 ..... end
316 ..... end
317 .....
318 ..... matrizBinaria=zeros(cols,cols)
319 ..... for i=1:cols
320 ..... for j=1:cols
321 ..... if i<j
322 ..... matrizBinaria(i,j)=round(x(contador))
323 ..... contador=contador+1
324 ..... end
325 ..... end
326 ..... end
327 .....
328 ..... TAC=0;areaTrocador=0;tempSaidaQ=0;tempSaidaF=0;deltaTMediaLog=0;
329 ..... for i=1:cols
330 ..... for j=1:cols
331 ..... if i<j
332 ..... if matrizBinaria(i,j)==1
333 ..... if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
334 ..... //i: CQ, j: CF
335 ..... cargaTermica=mCp(i)*matrizDeltaT(i,j)
336 ..... tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
337 ..... tempSaidaF=-cargaTermica/(mCp(j))+tempAtual(j)
338 ..... if (tempAtual(i)-tempAtual(j)>alfa)&(tempSaidaQ-tempAtual(j)>alfa)&(tempAtual(i)-temp
SaidaF>alfa)
339 ..... //TQentra-TFentra>alfa & TQsai-TFentra>alfa & TQentra-TFsai>alfa
340 ..... deltaTMediaLog=((tempSaidaQ-tempAtual(j))-(tempAtual(i)-tempSaidaF))/log((tempSaid
aQ-tempAtual(j))/(tempAtual(i)-tempSaidaF))
341 ..... areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
342 ..... tempAtual(i)=tempSaidaQ
343 ..... tempAtual(j)=tempSaidaF
344 ..... end
345 ..... else //a corrente i ganha calor da j
346 ..... //i: CF, j: CQ
347 ..... cargaTermica=mCp(i)*matrizDeltaT(i,j)
348 ..... tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
349 ..... tempSaidaQ=-cargaTermica/(mCp(j))+tempAtual(j)
350 ..... if (tempAtual(j)-tempAtual(i)>alfa)&(tempAtual(j)-tempSaidaF>alfa)&(tempSaidaQ-tempAt
ual(i)>alfa)
351 ..... //TQentra-TFentra>alfa & TQentra-TFsai>alfa & TQsai-TFentra>alfa
352 ..... deltaTMediaLog=((tempSaidaQ-tempAtual(i))-(tempAtual(j)-tempSaidaF))/log((tempSai
daQ-tempAtual(i))/(tempAtual(j)-tempSaidaF))
353 ..... areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
354 ..... tempAtual(i)=tempSaidaF
355 ..... tempAtual(j)=tempSaidaQ
356 ..... end
357 ..... end
358 ..... end
359 ..... end
360 ..... end
361 ..... end
362 .....
363 ..... cargaUQ=0; cargaUF=0;

```

```

364 --- for i=1:cols
365 ---     deltaT=tempAlvo(i)-tempAtual(i)
366 ---     if deltaT>beta //usa UQ
367 ---         cargaUQ=cargaUQ+mCp(i)*deltaT
368 ---         //UQ-> vapor saturado-> entra a 142,9°C
369 ---         deltaTMediaLog=((416.05-tempAtual(i))-(416.05-tempAlvo(i)))/log((416.05-tempAtual(i))/(416.05-tempAlvo(i)))
370 ---         areaTrocador=areaTrocador+abs(mCp(i)*(deltaT)/(U*deltaTMediaLog))
371 ---     elseif deltaT<(-beta) //usa UF
372 ---         cargaUF=cargaUF+mCp(i)*(-deltaT)
373 ---         //UF-> água resfriada-> entra a 25°C e sai a 30°C
374 ---         deltaTMediaLog=((tempAlvo(i)-298.15)-(tempAtual(i)-303.15))/log((tempAlvo(i)-298.15)/(tempAtual(i)-303.15))
375 ---         areaTrocador=areaTrocador+abs(mCp(i)*(-deltaT)/(U*deltaTMediaLog))
376 ---     end
377 --- end
378 ---
379 --- costTrocador=9094+221*(areaTrocador)^0.81
380 --- costUQ=coefCustoUQ*cargaUQ
381 --- costUF=coefCustoUF*cargaUF
382 --- TAC=costUQ+costUF+costTrocador
383 endfunction
384
385 //VE calcula a velocidade para cada partícula
386 function [X, VEL]=VE(X,BEST,PBEST,W,C1,C2,MAX,MIN)
387 --- rows=size(X,1)
388 --- cols=size(X,2)
389 --- MAX=MAX(:)
390 --- MIN=MIN(:)
391 --- VEL=zeros(rows, cols)
392 --- for i=1:rows
393 ---     for j=1:cols
394 ---         R1=rand()
395 ---         R2=rand()
396 ---         VELOCIDADE=W*VEL(i,j)+C1*R1*(PBEST(i,j)-X(i,j))+C2*R2*(BEST(j)-X(i,j))
397 ---         if(VELOCIDADE> MAX(j))
398 ---             VELOCIDADE=0.2*MAX(j)
399 ---         end
400 ---         if(VELOCIDADE<MIN(j))
401 ---             VELOCIDADE=0.2*MIN(j)
402 ---         end
403 ---         VEL(i,j)=VELOCIDADE
404 ---         X(i,j)=X(i,j)+VEL(i,j)
405 ---         if(X(i,j)>MAX(j))
406 ---             X(i,j)=MAX(j)
407 ---         end
408 ---         if(X(i,j)<MIN(j))
409 ---             X(i,j)=MIN(j)
410 ---         end
411 ---     end
412 --- end
413 endfunction
414
415 function [BEST,GBEST,Vetor]=PSO(W,C1,C2,NPAR,ITE,PAR,MAX,MIN,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
416 --- Vetor=0
417 --- x=Enxame(PAR,NPAR,MAX,MIN) //gera população inicial aleatória
418 --- PBEST=x //inicialmente as melhores partículas são aleatórias
419 --- YCAL=FOBJ(x,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF) //estima a função nos valores aleatórios
420 --- [valor,best]=min(YCAL)
421 --- GBEST=PBEST(best,:) //melhor global na 1ª interação
422 --- BEST=YCAL(best)
423 --- VEL=Enxame(PAR,NPAR,MAX,MIN) //velocidades aleatórias na 1ª interação
424
425 --- for i=1:ITE
426 ---     [xnew,VEL]=VE(x,GBEST,PBEST,W,C1,C2,MAX,MIN) //nova atualização de x utilizando as velocidades
427 ---     PBEST=PART(PBEST,xnew,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF) //atualização de PBEST (sempre atualiza com a nova busca)
428 ---     YCAL=FOBJ(PBEST,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
429 ---     [valor,new]=min(YCAL)
430 ---     NEW=YCAL(new)
431 ---     if(NEW<BEST)
432 ---         GBEST=xnew(new,:)
433 ---         BEST=NEW
434 ---     end
435 ---     Vetor(i)=BEST;

```

```

436 ..... x=xnew
437 .... end
438 endfunction
439
440
441
442
443 *****PROGRAMA PRINCIPAL*****
444
445
446 W=0.9
447 C1=2
448 C2=2
449 NPAR=100 //Número de partículas
450 ITE=300//Número de iterações
451 alfa=10
452 betha=1
453
454 //Ordem: C1 C2 H1 H2
455 mCp=[300 500 200 100] //[kW/K]
456 tempAtual=[323 353 423 443] //[K]
457 tempAlvo=[393 383 323 313] //[K]
458 U=0.2 //[kW/(m2.K)]
459 coefCustoUQ=110 //[$/(kW.ano)]
460 coefCustoUF=10 //[$/(kW.ano)]
461
462 //Redefinição das matrizes com a partição de uma corrente
463 posicaoCorrente=2
464 [mCp,tempAtual,tempAlvo]=PARTICAO(mCp,tempAtual,tempAlvo,posicaoCorrente)
465
466 //Cálculo do número de parâmetros
467 somatorio=0
468 for i=1:(size(mCp,2)-1)
469     somatorio=somatorio+i
470 end
471 PAR=2*somatorio+1 //Número de parâmetros a ser otimizados
472
473 //Cálculo de MÁX e MIN
474 [MIN,MAX]=DELTAS(tempAtual,tempAlvo)
475
476 [BEST,GBEST,Vetor]=PSO(W,C1,C2,NPAR,ITE,PAR,MAX,MIN,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
477 Iter=1:ITE;
478
479 //Para corrigir mCp por conta da divisão de correntes
480 mCp(posicaoCorrente)=mCp(posicaoCorrente)*GBEST(PAR)
481 mCp(posicaoCorrente+1)=mCp(posicaoCorrente+1)*(1-GBEST(PAR))
482 printf('Vetor mCp após a divisão.\n')
483 disp(mCp)
484
485 [matrizTrocaBest,matrizBinaria]=MATBEST(GBEST,mCp,tempAtual)
486 matrizGrafico=MATGRAF(matrizTrocaBest,mCp,tempAtual,tempAlvo)
487
488 printf('Melhor solução para %d iterações com %d partículas.\n',ITE,NPAR);disp(GBEST)
489 printf('Valor da função f(gbest)=%f.\n',BEST)
490
491 printf('Matriz de Trocas Térmicas.\n')
492 disp(matrizTrocaBest)
493 printf('Matriz Binária.\n')
494 disp(matrizBinaria)
495
496
497 cols=size(matrizGrafico,2)
498 lins=size(matrizGrafico,1)
499
500 y=zeros(lins,cols+2);
501 y(:,1)=matrizGrafico(:,1);
502 y(:,cols+2)=matrizGrafico(:,cols);
503
504 for j=1:cols
505     y(:,j+1)=matrizGrafico(:,j);
506 end
507
508 x=0:(cols+1)
509
510 a=get('current_axes')//Identificador dos eixos criados
511 a.axes_visible='on'; //Torna os eixos visíveis
512 a.font_size=3; //Define o tamanho da fonte

```

```

513 a.sub_tics=[1,50];
514 a.thickness=3;
515 a.font_size=4;
516
517 Cor=['red','blue','black','green','cyan','magenta','blackread'];
518
519 scf(0);
520 for i=1:lines
521     plot(x,y(i,:),Cor(i));
522 end
523
524 xlabel("Trocas térmicas","fontsize",-5)
525 ylabel("Temperatura (K)","fontsize",-5)
526
527 TAC=0;areaTrocador=0;cargaUQ=0;cargaUF=0;
528 [TAC,areaTrocador,cargaUQ,cargaUF]=RESULT(GBEST,mCp,tempAtual,tempAlvo,U,coefCustoUQ,coefCustoUF)
529
530 printf('TESTE ADICIONAL\n')
531 printf('Custo Total Anual=%f\n',TAC)
532 printf('Area de Trocador=%f.m2\n',areaTrocador)
533 printf('Carga de UQ=%f.kW\n',cargaUQ)
534 printf('Carga de UF=%f.kW\n',cargaUF)
535
536 //plot(Iter,Vetor,'black')

```

APÊNDICE C – CÓDIGO-FONTE DA OTIMIZAÇÃO PARA A INDÚSTRIA DE BENEFICIAMENTO DE SOJA

```

1  clc
2  clear
3
4  //DELTAS: retorna a matriz MAX e MIN que são usadas para os cálculos
5  function [MIN,MAX]=DELTAS(tempAtual,tempAlvo)
6  cols=size(tempAtual,2)
7  a=0; numCols=0;
8
9  for i=1:cols
10     a=cols-i
11     numCols=numCols+a
12  end
13
14  numCols=2*numCols
15  MAX=zeros(numCols)
16  MIN=zeros(numCols)
17
18  a=0; cont=0;
19  for i=1:cols
20     a=cols-i
21
22     if a>0 then
23         deltaT=tempAlvo(i)-tempAtual(i)
24         for j=1:a
25             cont=cont+1
26             if deltaT>0
27                 MAX(cont)=deltaT
28                 MIN(cont)=0
29             else deltaT<0
30                 MAX(cont)=-deltaT
31                 MIN(cont)=0
32             end
33         end
34         //para garantir convergência, modificamos MAX e MIN
35         MAX(cont)=3*MAX(cont)
36         MIN(cont)=-1*MAX(cont)
37     end
38 end
39
40
41 for i=(cont+1):numCols
42     MAX(i)=1
43     MIN(i)=0
44 end
45
46 endfunction
47
48 //Enxame: retorna a população aleatória com todos valores entre MIN e MAX
49 function x=Enxame(numParametros,numParticulas,deltaMax,deltaMin)
50 x=zeros(numParticulas, numParametros)
51 for j=1:numParametros
52     for i=1:numParticulas
53         x(i,j)=deltaMin(j)+(deltaMax(j)-deltaMin(j))*rand()
54     end
55 end
56 endfunction
57
58 //FOBJ: calcula o valor da função a ser minimizada para cada partícula
59 function fobj=FOBJ(x,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF)
60 rows=size(x,1)
61 fobj=zeros(rows)
62 for i=1:rows
63     fobj(i)=FUN(x(i,:),mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF)
64 end
65 endfunction
66
67 //FUN: calcula o TAC para um determinado arranjo
68 function fun=FUN(x,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF)
69 fun=0
70

```

```

71  -- cols=size(mCp,2)
72  -- matrizDeltaT=zeros(cols,cols)
73  -- contador=1
74  -- for i=1:cols
75  --     for j=1:cols
76  --         if i<j
77  --             matrizDeltaT(i,j)=x(contador)
78  --             contador=contador+1
79  --         end
80  --     end
81  -- end
82  --
83  -- matrizBinaria=zeros(cols,cols)
84  -- for i=1:cols
85  --     for j=1:cols
86  --         if i<j
87  --             matrizBinaria(i,j)=round(x(contador))
88  --             contador=contador+1
89  --         end
90  --     end
91  -- end
92  --
93  -- areaTrocador=0; tempSaidaQ=0; tempSaidaF=0; deltaTMediaLog=0;
94  -- for i=1:cols
95  --     for j=1:cols
96  --         if i<j
97  --             if matrizBinaria(i,j)==1
98  --                 if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
99  --                     //i: CQ, j: CF
100  --                     cargaTermica=mCp(i)*matrizDeltaT(i,j)
101  --                     tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
102  --                     tempSaidaF=-cargaTermica/(mCp(j))+tempAtual(j)
103  --                     if (tempAtual(i)-tempAtual(j)>alfa) & (tempSaidaQ-tempAtual(j)>alfa) & (tempAtual(i)-temp
SaidaF>alfa)
104  --                         //TQentra-TFentra>alfa & TQsai-TFentra>alfa & TQentra-TFsai>alfa
105  --                         deltaTMediaLog=((tempSaidaQ-tempAtual(j))-(tempAtual(i)-tempSaidaF))/log((tempSaid
aQ-tempAtual(j))/(tempAtual(i)-tempSaidaF))
106  --                         areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
107  --                         tempAtual(i)=tempSaidaQ
108  --                         tempAtual(j)=tempSaidaF
109  --                     end
110  --                 else //a corrente i ganha calor da j
111  --                     //i: CF, j: CQ
112  --                     cargaTermica=mCp(i)*matrizDeltaT(i,j)
113  --                     tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
114  --                     tempSaidaQ=-cargaTermica/(mCp(j))+tempAtual(j)
115  --                     if (tempAtual(j)-tempAtual(i)>alfa) & (tempAtual(j)-tempSaidaF>alfa) & (tempSaidaQ-tempAt
ual(i)>alfa)
116  --                         //TQentra-TFentra>alfa & TQentra-TFsai>alfa & TQsai-TFentra>alfa
117  --                         deltaTMediaLog=((tempSaidaQ-tempAtual(i))-(tempAtual(j)-tempSaidaF))/log((tempSai
daQ-tempAtual(i))/(tempAtual(j)-tempSaidaF))
118  --                         areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
119  --                         tempAtual(i)=tempSaidaF
120  --                         tempAtual(j)=tempSaidaQ
121  --                     end
122  --                 end
123  --             end
124  --         end
125  --     end
126  -- end
127  --
128  -- cargaUQ=0; cargaUF=0;
129  -- for i=1:cols
130  --     deltaT=tempAlvo(i)-tempAtual(i)
131  --     if deltaT>beta //usa UQ
132  --         cargaUQ=cargaUQ+mCp(i)*deltaT
133  --         //UQ -> vapor saturado -> entra a 250°C
134  --         deltaTMediaLog=((523.15-tempAtual(i))-(523.15-tempAlvo(i)))/log((523.15-tempAtual(i))/(523.15-tem
pAlvo(i)))
135  --         areaTrocador=areaTrocador+abs(mCp(i)*(deltaT)/(UUQ*deltaTMediaLog))
136  --     elseif deltaT<(-beta) //usa UF
137  --         cargaUF=cargaUF+mCp(i)*(-deltaT)
138  --         //UF -> água resfriada -> entra a 25°C e sai a 29°C
139  --         deltaTMediaLog=((tempAlvo(i)-298.15)-(tempAtual(i)-302.15))/log((tempAlvo(i)-298.15)/(tempAtual(i)
-302.15))
140  --         areaTrocador=areaTrocador+abs(mCp(i)*(-deltaT)/(UUF*deltaTMediaLog))
141  --     end

```

```

142     end
143
144     costTrocador=5358.8+128.41*areaTrocador
145     costUQ=coefCustoUQ*cargaUQ
146     costUF=coefCustoUF*cargaUF
147     fun=costUQ+costUF+costTrocador
148 endfunction
149
150 //MATBEST retorna a matriz com os deltas das trocas térmicas que ocorreram e a matriz binária
151 function [matrizTrocaBest,matrizBinaria]=MATBEST(x,mCp,tempAtual)
152     cols=size(mCp,2)
153     matrizDeltaT=zeros(cols,cols)
154     matrizTrocaBest=zeros(cols,cols)
155     contador=1
156     for i=1:cols
157         for j=1:cols
158             if i<j
159                 matrizDeltaT(i,j)=x(contador)
160                 contador=contador+1
161             end
162         end
163     end
164
165     matrizBinaria=zeros(cols,cols)
166     for i=1:cols
167         for j=1:cols
168             if i<j
169                 matrizBinaria(i,j)=round(x(contador))
170                 contador=contador+1
171             end
172         end
173     end
174
175     tempSaidaQ=0;tempSaidaF=0;cargaTermica=0
176     for i=1:cols
177         for j=1:cols
178             if i<j
179                 if matrizBinaria(i,j)==1
180                     if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
181                         //i: CQ, j: CF
182                         cargaTermica=mCp(i)*matrizDeltaT(i,j)
183                         tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
184                         tempSaidaF=-cargaTermica/(mCp(j))+tempAtual(j)
185                         if (tempAtual(i)-tempAtual(j)>alfa) & (tempSaidaQ-tempAtual(j)>alfa) & (tempAtual(i)-temp
SaidaF>alfa)
186                             //TQentra-TFentra>alfa & TQsai-TFentra>alfa & TQentra-TFsai>alfa
187                             tempAtual(i)=tempSaidaQ
188                             tempAtual(j)=tempSaidaF
189                         else matrizDeltaT(i,j)=0
190                         end
191                     else //a corrente i ganha calor da j
192                         //i: CF, j: CQ
193                         cargaTermica=mCp(i)*matrizDeltaT(i,j)
194                         tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
195                         tempSaidaQ=-cargaTermica/(mCp(j))+tempAtual(j)
196                         if (tempAtual(j)-tempAtual(i)>alfa) & (tempAtual(j)-tempSaidaF>alfa) & (tempSaidaQ-tempAt
ual(i)>alfa)
197                             //TQentra-TFentra>alfa & TQentra-TFsai>alfa & TQsai-TFentra>alfa
198                             tempAtual(i)=tempSaidaF
199                             tempAtual(j)=tempSaidaQ
200                         else matrizDeltaT(i,j)=0
201                         end
202                     end
203                 else matrizDeltaT(i,j)=0
204                 end
205             end
206         end
207     end
208
209     matrizTrocaBest=matrizDeltaT
210 endfunction
211
212 //MATGRAF retorna o gráfico das trocas térmicas em cada trocador
213 function matrizGrafico=MATGRAF(matrizDeltaT,mCp,tempAtual,tempAlvo)
214     cols=size(mCp,2)
215     contador=2
216     matrizGrafico(:,1)=tempAtual

```

```

217 for i=1:cols
218     for j=1:cols
219         if i<j
220             if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
221                 //i: CQ, j: CF
222                 cargaTermica=mCp(i)*matrizDeltaT(i,j)
223                 tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
224                 tempSaidaF=-cargaTermica/(mCp(j))+tempAtual(j)
225                 if (tempAtual(i)-tempAtual(j)>alfa) & (tempSaidaQ-tempAtual(j)>alfa) & (tempAtual(i)-tempSaidaF>alfa)
226                     //Tqentra-TFentra>alfa & Tqsai-TFentra>alfa & Tqentra-TFsai>alfa
227                     tempAtual(i)=tempSaidaQ
228                     tempAtual(j)=tempSaidaF
229                 end
230                 matrizGrafico(:,contador)=tempAtual
231                 contador=contador+1
232             end
233             elseif matrizDeltaT(i,j)>0 //a corrente i ganha calor da j
234                 //i: CF, j: CQ
235                 cargaTermica=mCp(i)*matrizDeltaT(i,j)
236                 tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
237                 tempSaidaQ=-cargaTermica/(mCp(j))+tempAtual(j)
238                 if (tempAtual(j)-tempAtual(i)>alfa) & (tempAtual(j)-tempSaidaF>alfa) & (tempSaidaQ-tempAtual(i)>alfa)
239                     //Tqentra-TFentra>alfa & Tqentra-TFsai>alfa & Tqsai-TFentra>alfa
240                     tempAtual(i)=tempSaidaF
241                     tempAtual(j)=tempSaidaQ
242                 end
243                 matrizGrafico(:,contador)=tempAtual
244                 contador=contador+1
245             end
246         end
247     end
248 end
249 end
250
251 for i=1:cols
252     if abs(tempAtual(i)-tempAlvo(i))>betha
253         tempAtual(i)=tempAlvo(i)
254         matrizGrafico(:,contador)=tempAtual
255         contador=contador+1
256     end
257 end
258 endfunction
259
260 //PART atualiza PBEST (melhor de cada particula)
261 function PBEST=PART(x,xnew,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF)
262     rows=size(x,1)
263     cols=size(x,2)
264     YCAL=FOBJ(x,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF)
265     NEW=FOBJ(xnew,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF) //xnew vem da rotina velocidade
266     for i=1:rows
267         if(YCAL(i)>NEW(i))
268             x(i,:)=xnew(i,:)
269         end
270     end
271     PBEST=x
272 endfunction
273
274 //RESULT calcula o TAC, área e cargas de UQ e UF para um determinado arranjo
275 function [TAC,areaTrocador,cargaUQ,cargaUF]=RESULT(x,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF)
276     cols=size(mCp,2)
277     matrizDeltaT=zeros(cols,cols)
278     contador=1
279     for i=1:cols
280         for j=1:cols
281             if i<j
282                 matrizDeltaT(i,j)=x(contador)
283                 contador=contador+1
284             end
285         end
286     end
287
288     matrizBinaria=zeros(cols,cols)
289     for i=1:cols
290         for j=1:cols
291             if i<j

```

```

292 ..... matrizBinaria(i,j)=round(x(contador))
293 ..... contador=contador+1
294 ..... end
295 ..... end
296 ..... end
297 .....
298 ..... TAC=0;areaTrocador=0;tempSaidaQ=0;tempSaidaF=0;deltaTMediaLog=0;
299 ..... for i=1:cols
300 .....     for j=1:cols
301 .....         if i<j
302 .....             if matrizBinaria(i,j)==1
303 .....                 if matrizDeltaT(i,j)<0 //a corrente i perde calor para a j
304 .....                     //i: CQ, j: CF
305 .....                     cargaTermica=mCp(i)*matrizDeltaT(i,j)
306 .....                     tempSaidaQ=tempAtual(i)+matrizDeltaT(i,j)
307 .....                     tempSaidaF=-cargaTermica/(mCp(j))+tempAtual(j)
308 .....                     if (tempAtual(i)-tempAtual(j)>alfa)&(tempSaidaQ-tempAtual(j)>alfa)&(tempAtual(i)-temp
SaidaF>alfa)
309 .....                         //TQentra-TFentra>alfa & TQsai-TFentra>alfa & TQentra-TFsai>alfa
310 .....                         deltaTMediaLog=((tempSaidaQ-tempAtual(j))-(tempAtual(i)-tempSaidaF))/log((tempSaid
aQ-tempAtual(j))/(tempAtual(i)-tempSaidaF))
311 .....                         areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
312 .....                         tempAtual(i)=tempSaidaQ
313 .....                         tempAtual(j)=tempSaidaF
314 .....                     end
315 .....                 else //a corrente i ganha calor da j
316 .....                     //i: CF, j: CQ
317 .....                     cargaTermica=mCp(i)*matrizDeltaT(i,j)
318 .....                     tempSaidaF=tempAtual(i)+matrizDeltaT(i,j)
319 .....                     tempSaidaQ=-cargaTermica/(mCp(j))+tempAtual(j)
320 .....                     if (tempAtual(j)-tempAtual(i)>alfa)&(tempAtual(j)-tempSaidaF>alfa)&(tempSaidaQ-tempAt
ual(i)>alfa)
321 .....                         //TQentra-TFentra>alfa & TQentra-TFsai>alfa & TQsai-TFentra>alfa
322 .....                         deltaTMediaLog=((tempSaidaQ-tempAtual(i))-(tempAtual(j)-tempSaidaF))/log((tempSai
daQ-tempAtual(i))/(tempAtual(j)-tempSaidaF))
323 .....                         areaTrocador=areaTrocador+abs(cargaTermica/(U*deltaTMediaLog))
324 .....                         tempAtual(i)=tempSaidaF
325 .....                         tempAtual(j)=tempSaidaQ
326 .....                     end
327 .....                 end
328 .....             end
329 .....         end
330 .....     end
331 ..... end
332 .....
333 ..... cargaUQ=0; cargaUF=0;
334 ..... for i=1:cols
335 .....     deltaT=tempAlvo(i)-tempAtual(i)
336 .....     if deltaT>beta //usa UQ
337 .....         cargaUQ=cargaUQ+mCp(i)*deltaT
338 .....         //UQ -> vapor saturado -> entra a 250°C
339 .....         deltaTMediaLog=((523.15-tempAtual(i))-(523.15-tempAlvo(i)))/log((523.15-tempAtual(i))/(523.15-tem
pAlvo(i)))
340 .....         areaTrocador=areaTrocador+abs(mCp(i)*(deltaT)/(UUQ*deltaTMediaLog))
341 .....     elseif deltaT<(-beta) //usa UF
342 .....         cargaUF=cargaUF+mCp(i)*(-deltaT)
343 .....         //UF -> água resfriada -> entra a 25°C e sai a 29°C
344 .....         deltaTMediaLog=((tempAlvo(i)-298.15)-(tempAtual(i)-302.15))/log((tempAlvo(i)-298.15)/(tempAtual(i)
)-302.15))
345 .....         areaTrocador=areaTrocador+abs(mCp(i)*(-deltaT)/(UUF*deltaTMediaLog))
346 .....     end
347 ..... end
348 .....
349 ..... costTrocador=5358.8+128.41*areaTrocador
350 ..... costUQ=coefCustoUQ*cargaUQ
351 ..... costUF=coefCustoUF*cargaUF
352 ..... TAC=costUQ+costUF+costTrocador
353 endfunction
354 .....
355 //VE calcula a velocidade para cada partícula
356 function [X, VEL]=VE(X,BEST,PBEST,W,C1,C2,MAX,MIN)
357 ..... rows=size(X,1)
358 ..... cols=size(X,2)
359 ..... MAX=MAX(:)
360 ..... MIN=MIN(:)
361 ..... VEL=zeros(rows, cols)

```

```

362     for i=1:rows
363         for j=1:cols
364             R1=rand()
365             R2=rand()
366             VELOCIDADE=W*VEL(i,j)+C1*R1*(PBEST(i,j)-X(i,j))+C2*R2*(BEST(j)-X(i,j))
367             if(VELOCIDADE> MAX(j))
368                 VELOCIDADE=0.2*MAX(j)
369             end
370             if(VELOCIDADE<MIN(j))
371                 VELOCIDADE=0.2*MIN(j)
372             end
373             VEL(i,j)=VELOCIDADE
374             X(i,j)=X(i,j)+VEL(i,j)
375             if(X(i,j)>MAX(j))
376                 X(i,j)=MAX(j)
377             end
378             if(X(i,j)<MIN(j))
379                 X(i,j)=MIN(j)
380             end
381         end
382     end
383 endfunction
384
385 function [BEST,GBEST,Vetor]=PSO(W,C1,C2,NPAR,ITE,PAR,MAX,MIN,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF)
386     Vetor=0
387     x=Enxame(PAR,NPAR,MAX,MIN) //gera população inicial aleatória
388     PBEST=x //inicialmente as melhores partículas são aleatórias
389     YCAL=FOBJ(x,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF) //estima a função nos valores aleatórios
390     [valor,best]=min(YCAL)
391     GBEST=PBEST(best,:) //melhor global na 1ª interação
392     BEST=YCAL(best)
393     VEL=Enxame(PAR,NPAR,MAX,MIN) //velocidades aleatórias na 1ª interação
394
395     for i=1:ITE
396         [xnew,VEL]=VE(x,GBEST,PBEST,W,C1,C2,MAX,MIN) //nova atualização de x utilizando as velocidades
397         PBEST=PART(PBEST,xnew,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF) //atualização de PBEST (sempre
398         //atualiza com a nova busca)
399         YCAL=FOBJ(PBEST,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF)
400         [valor,new]=min(YCAL)
401         NEW=YCAL(new)
402         if(NEW<BEST)
403             GBEST=xnew(new,:)
404             BEST=NEW
405         end
406         Vetor(i)=BEST;
407         x=xnew
408     end
409 endfunction
410
411 //*****PROGRAMA PRINCIPAL*****
412
413
414 W=0.9
415 C1=2
416 C2=2
417 NPAR=100 //Número de partículas
418 ITE=300 //Número de iterações
419 alfa=10
420 betha=1
421
422 //Ordem: C9-C4-H10-H12-H5
423 mCp=[5597 4241 5170 4051 4238] //[kcal/h.°C]
424 tempAtual=[55 70 90 90 100] //[°C]
425 tempAlvo=[90 90 40 40 70] //[°C]
426 UUQ=4185 //[kcal/h.m2.°C]
427 UUF=2108 //[kcal/h.m2.°C]
428 U=112 //[kcal/h.m2.°C]
429
430 coefCustoUQ=0.20586 //[US$.h/(kcal.ano)]
431 coefCustoUF=0.03942 //[US$.h/(kcal.ano)]
432
433 //Cálculo do número de parâmetros
434 somatorio=0
435 for i=1:(size(mCp,2)-1)
436     somatorio=somatorio+i

```

```

437 end
438 PAR=2*somatorio //Número de parâmetros a ser otimizados
439
440 //cálculo de MÁX e MIN
441 [MIN,MAX]=DELTAS(tempAtual,tempAlvo)
442
443 [BEST,GBEST,Vetor]=PSQ(W,C1,C2,NPAR,ITE,PAR,MAX,MIN,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF)
444 Iter=1:ITE;
445
446 [matrizTrocaBest,matrizBinaria]=MATBEST(GBEST,mCp,tempAtual)
447 matrizGrafico=MATGRAF(matrizTrocaBest,mCp,tempAtual,tempAlvo)
448
449 printf('Melhor solução para %d iterações com %d partículas.\n',ITE,NPAR); disp(GBEST)
450 printf('Valor da função f(gbest)=%f.\n',BEST)
451
452 printf('Matriz de Trocas Térmicas.\n')
453 disp(matrizTrocaBest)
454 printf('Matriz Binária.\n')
455 disp(matrizBinaria)
456
457
458 cols=size(matrizGrafico,2)
459 lins=size(matrizGrafico,1)
460
461 y=zeros(lins,cols+2);
462 y(:,1)=matrizGrafico(:,1);
463 y(:,cols+2)=matrizGrafico(:,cols);
464
465 for j=1:cols
466     y(:,j+1)=matrizGrafico(:,j);
467 end
468
469 x=0:(cols+1)
470
471 a=get('current_axes')//Identificador dos eixos criados
472 a.axes_visible="on"; //Torna os eixos visíveis
473 a.font_size=3; //Define o tamanho da fonte
474 a.sub_tics=[1,50];
475 a.thickness=3;
476 a.font_size=4;
477
478 Cor=['red','blue','black','green','cyan','magenta','blackread'];
479
480 scf(0);
481 for i=1:lins
482     plot(x,y(i,:),Cor(i));
483 end
484
485 xlabel("Trocas térmicas","fontsize",-5)
486 ylabel("Temperatura (°C)","fontsize",-5)
487
488 TAC=0;areaTrocador=0;cargaUQ=0;cargaUF=0;
489 [TAC,areaTrocador,cargaUQ,cargaUF]=RESULT(GBEST,mCp,tempAtual,tempAlvo,coefCustoUQ,coefCustoUF)
490
491 printf('TESTE ADICIONAL.\n')
492 printf('Custo Total Anual=%f.\n',TAC)
493 printf('Area de Trocador=%f m2\n',areaTrocador)
494 printf('Carga de UQ=%f kcal/h\n',cargaUQ)
495 printf('Carga de UF=%f kcal/h\n',cargaUF)
496
497 //plot(Iter,Vetor,'black')

```