IBP1057_06

OTIMIZAÇÃO DO SCHEDULING DO REFINO UTILIZANDO ALGORITMOS GENÉTICOS

Leonardo M. Simão[1], Douglas M. Dias[2], Marco Aurélio C. Pacheco[3]

Resumo

Este trabalho introduz um novo conceito de otimização da programação global da produção em refinarias de petróleo, empregando algoritmos genéticos e co-evolução cooperativa, que é capaz de lidar com as restrições do problema e gerar apenas soluções viáveis. São apresentados um sistema protótipo e o estudo de caso, que emprega uma refinaria simples que envolve todos os tipos de equipamentos, atividades e restrições presentes em uma refinaria real. As restrições existentes são tanto as de precedência, quanto às operacionais inerentes ao problema e à planta escolhida. Os testes demonstraram a capacidade dos modelos desenvolvidos em gerar soluções viáveis, sem a necessidade de heurísticas de correção, e os resultados obtidos foram comparados com os de um processo de busca aleatória. Foram criados vários cenários de teste com demandas, restrições e tamanhos diferentes. Os resultados obtidos foram sempre superiores aos obtidos pela busca aleatória, indicando que o modelo de otimização desenvolvido se constitui numa alternativa promissora para a construção de ferramentas de apoio à decisão na programação global de uma refinaria.

Abstract

This work presents a new concept in global programming optimization of oil refineries production, by using genetic algorithms and cooperative co-evolution, to deal with the problem's restrictions and generating feasible solutions only. The work describes the implementation of a tool and the case study, which provides the necessary characteristics to test and evaluate the proposed model. A simple refinery, with all usual types of equipments, activities and restrictions of a real refinery, was created. The existing restrictions are both of precedence and the operational ones that are inherent to the problem and to the chosen plant. The experiments showed the model's capability to generate feasible solutions, with no correction heuristics necessity, and the achieved results were compared with the ones of a random search process. Several test scenarios with different demands, restrictions and sizes, were created. In all cases, the results were always better than the ones achieved by random search, indicating that the developed optimization model represents a promising alternative for the construction of a refinery's global programming decision support tools.

1. Introdução

Refinarias de petróleo constituem um dos mais importantes exemplos de plantas contínuas multiproduto, isto é, um sistema de processamento contínuo gerador de múltiplos produtos simultâneos. A otimização de sua programação é considerada um problema complexo, devido ao número e diversidade de atividades existentes e seus diferentes objetivos. Além disso, trata-se de um problema com restrições de precedência, isto é, o planejamento de algumas atividades depende de que outras atividades já tenham sido planejadas.

Uma refinaria, em geral, processa um ou mais tipos de petróleo, produzindo uma série de produtos derivados, como o GLP (gás liquefeito de petróleo), a nafta, o querosene e o óleo diesel (Hax & Candea, 1984).

Na programação define-se como as quantidades de cada produto, apontadas no planejamento, devem ser produzidas, considerando-se:

- Previsão de produção definida pelo planejamento (preços, datas e quantidades);

[1] Mestre, Engenheiro de Computação – Chemtech - A Siemens Company
[2] Mestre, Engenheiro Eletricista – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
[3] PhD, Engenheiro Eletrônico – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)

- Quantidades e qualidades de matéria-prima comprada;
- Preço, quantidade e qualidades de produtos intermediários (inventário atual);
- Restrições de estoques e alinhamentos (topologia da planta);
- Forma de escoamento e entrega dos produtos finais.

O programador de produção deve definir as atividades que serão realizadas em cada equipamento, a cada instante, de modo a viabilizar da melhor forma possível a produção prevista. Ou seja, o resultado final da programação é um conjunto diário de ordens de tarefas a serem executadas na operação dos equipamentos da refinaria.

Neste projeto o objetivo foi demonstrar a aplicabilidade e o desempenho da computação evolucionária na otimização da programação da produção numa refinaria de petróleo, considerando as diversas etapas do refino. Em particular, busca-se um sistema capaz de tratar o problema de modo global, isto é, considerando todas as etapas do refino e seus respectivos objetivos a serem otimizados.

A Computação Evolucionária compreende algoritmos de otimização e busca inspirados no princípio Darwiniano da evolução das espécies e na genética. São algoritmos probabilísticos, que fornecem um mecanismo de busca paralela e adaptativa baseado no princípio de sobrevivência dos mais aptos e na reprodução. O mecanismo é obtido a partir de uma população de indivíduos (soluções), representados por cromossomos (palavras binárias, vetores, matrizes etc), cada um associado a uma aptidão (avaliação da solução no problema), que são submetidos a um processo de evolução (seleção, reprodução, cruzamento e mutação) por vários ciclos.

A co-evolução cooperativa foi inspirada no conceito de simbiose, onde duas ou mais espécies interagem e colaboram para a evolução uma da outra. Cada espécie evolui em seu nicho ou especialidade, cooperando para que a união delas faça com que o ecossistema que elas representam evolua como um todo. Como na natureza, as espécies são geneticamente isoladas, ou seja, os indivíduos só podem se reproduzir com outros indivíduos da mesma espécie, que estão em populações separadas. As espécies somente interagem umas com as outras através de um domínio compartilhado e têm uma relação apenas de cooperação.

A seguir é apresentado o modelo desenvolvido para tratar a otimização do refino.

2. Modelo Co-evolucionário para Otimização da Programação da Produção

O modelo proposto para solucionar este problema é um modelo co-evolucionário cooperativo formado por duas espécies. A primeira espécie irá decidir qual tarefa planejar, enquanto que a segunda irá decidir com quais recursos esta tarefa será executada.

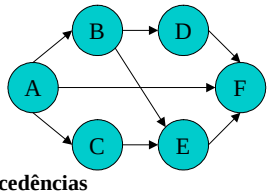
2.1. Representação

A representação adotada neste trabalho conta com duas espécies, que evoluem paralelamente aspectos diversos do problema e colaboram para formar a solução do problema.

A primeira espécie, chamada de "Alocação de Tempo", conta com duas estruturas: a lista de prioridades e o grafo de precedências. A lista de prioridades contém as prioridades de cada tarefa. O grafo contém as informações de precedências (ou dependências) entre tarefas, ou seja, quais tarefas devem ser programadas antes de outras.

Lista de Prioridades

Tarefa	**A**	**B**	**C**	**D**	**E**	**F**
Prioridade	**5**	**1**	**2**	**6**	**4**	**3**



Grafo de Precedências

Figura 1. Representação da espécie "Alocação de Tempo".

Na Figura 1, são mostradas as duas estruturas que compõem a representação da espécie "Alocação de Tempo". A lista de prioridades é o cromossomo onde os operadores genéticos irão atuar. A seguir, na Figura 2, é apresentado um esboço do algoritmo para selecionar as tarefas a programar a partir do cromossomo da espécie 1.

```
procedure Programar tarefas
grafo ← grafo de precedências
while (tamanho(grafo)>0)
    if (todo vértice tem um precedente) then
        grafo inválido
        exit
    else
        v ← vértice com a maior prioridade entre os sem precedentes
        programar v
        remover v do grafo e todos as ligações que partem dele
    end if
end while
```

Figura 2. Algoritmo para programar tarefas a partir da espécie 1.

É importante ressaltar que o algoritmo apresentado ilustra como o sistema utiliza a lista de prioridades, juntamente com o grafo de precedências, para selecionar as tarefas a programar respeitando as restrições de precedência.

A segunda espécie, chamada de "Alocação de Recurso" conta com um vetor de listas. Cada posição do vetor está associada a uma tarefa, enquanto que cada lista contém os recursos que podem executá-la. O algoritmo tenta alocar o primeiro recurso disponível, na ordem em que ele aparece na lista.

2.2. Decodificação

A decodificação do cromossomo em solução é feita combinando-se as informações de cada uma das espécies. Abaixo, na Figura 3, pode-se ver a essência do algoritmo em pseudocódigo.

```
procedure Decodifica
begin
    cromossomo 1 ← indivíduo "Alocação de Tempo" selecionado
    cromossomo 2 ← indivíduo "Alocação de Recurso" selecionado
    while (existem tarefas a planejar)
    begin
        seleciona tarefa usando cromossomo 1
        seleciona recurso para tarefa usando cromossomo 2
        programa tarefa
    end while
end
```

Figura 3. Rotina de decodificação.

Pode-se perceber que, para decodificar o cromossomo e formar uma solução completa, é preciso selecionar um indivíduo de cada espécie. Esta seleção chama-se colaboração. Existem vários métodos de seleção de colaboradores. Para este trabalho foi escolhido o método otimista, onde se atribui à avaliação do indivíduo o valor da avaliação da melhor colaboração feita por ele.

2.3. Avaliação

A função objetivo deve incorporar cada um dos objetivos do *scheduling*, ou seja:

- **Atender à demanda dos produtos.** Os itens de venda ou envio devem ser atendidos sem atraso;
- **Minimizar os custos de produção.** Minimizar o custo da matéria-prima processada e minimizar o custo operacional. O custo da matéria-prima pode ser calculado a partir das quantidades de determinados petróleos que serão destilados durante o horizonte de programação. Já os custos operacionais estão relacionados à forma de operar a planta, evitando trocas freqüentes de campanhas e evitando trocas de tanques numa mesma atividade;
- **Atender, sem desvio, às especificações dos produtos.** As propriedades do derivados comercializados devem estar dentro dos limites mínimos e máximos aceitos para o produto;

O custo por não atendimento da demanda pode ser quantificado na forma de uma multa por atraso na entrega dos produtos (dos itens de envio ou venda). Para que esta multa fique proporcional à importância deste atraso em relação ao atraso de outros itens, a abordagem utilizada foi a de calcular o preço da parcela do item atrasada, ou seja:

$$C_{AD} = \sum_{i \in IV} preçoproduto_i * volumenãoentregue_i \quad (1)$$

Onde *IV* é o conjunto dos itens de venda do cenário.

Para se calcular o custo de matéria-prima, basta multiplicar o preço dos petróleos utilizados na destilação pela quantidade processada. Ou seja:

$$C_{MP} = \sum_{j \in UDA} \sum_{i \in C} preço_i * volumeprocessado_{i,j} \quad (2)$$

Onde *UDA* é o conjunto das unidades de destilação e *C* é o conjunto dos petróleos crus utilizados na refinaria.

Os custos operacionais considerados neste trabalho compreendem apenas o número de trocas de tanques ou esferas durante uma mesma atividade ou item. Isto porque foi admitido que o número de trocas de campanha é fixo por cenário e conhecido *a priori*. Então este custo pode ser descrito como:

$$C_{OP} = \sum_{i \in AT} trocas_i * custotroca \quad (3)$$

Onde *AT* é o conjunto das atividades de transferências, e o *custotroca* é uma constante que representa o custo de uma troca de tanque numa transferência.

O custo por desvio de especificação foi quantificado de uma forma muito simples, que visa a levar em conta o impacto gerado pela não especificação de um produto que deve ser vendido. Ao não atender às especificações de um produto, este não pode ser vendido, então pode-se deixar de computar o valor da parcela do item de venda que não pôde ser vendida por estar fora de especificação. Ou seja:

$$C_{DE} = \sum_{i \in IV} preçoproduto_i * volumenãoespecificado_i \quad (4)$$

Ao final tem-se a função objetivo (FO), onde *wi* representa a importância de cada objetivo:

$$FO = w_1 C_{AD} + w_2 C_{MP} + w_3 C_{OP} + w_4 C_{DE} \quad (5)$$

2.4. Operadores

Os operadores aplicados aos indivíduos de ambas as espécies são operadores típicos de problemas de ordem (como o problema do caixeiro viajante, por exemplo). É importante ressaltar que, na espécie "Alocação de Recurso", os operadores atuam nos alelos dos genes, ou seja, na lista de recursos de cada atividade, mas não entre atividades.

Os operadores de *crossover* utilizados foram: *Order Crossover* (OX), *Partially Mapped Crossover* (PMX) e *Cycle Crossover* (CX). Os de mutação foram: *Swap* (SM) e *Position Inversion* (PI) (Michalewicz, 1996).

2.5. Representação do Tempo

Como exposto por Moro (2000), a variável *tempo* assume importância primordial em problemas de programação da produção. Por exemplo, se o horizonte de programação é de sete dias, durante todo este tempo a unidade de destilação tem que estar sendo alimentada com petróleo a uma determinada vazão. Quantas cargas diferentes (a partir de tanques diferentes) serão feitas nesta unidade de processo?

Normalmente, fatias de tempo com duração uniforme são definidas ao longo do horizonte da programação, como na abordagem apresentada por Smania (2002). Nesta abordagem, o número e a duração destas fatias é arbitrado, de modo que as decisões relevantes ocorram na fronteira entre duas fatias. Ou seja, dependendo da granularidade exigida pelo problema, um grande número de fatias de tempo pode ser gerado, tornando computacionalmente inviável a obtenção de uma solução ótima.

O esquema adotado neste trabalho é o de dividir (discretizar) *não* o tempo, mas sim as *quantidades* transferidas em cada uma das atividades. Para isso é adotado um parâmetro chamado *volume máximo transferido*, para certas atividades. Este parâmetro determina uma espécie de "tamanho de lote" para as atividades de transferência. Esta é uma solução que permite ao usuário discretizar conforme o tipo de atividade.

3. Estudo de Caso

3.1. Descrição do Protótipo

A planta de refinaria para o estudo de casos deste trabalho foi uma planta simples, apresentada por Smania (2002), que apresenta todos os tipos de equipamentos e restrições de uma refinaria completa (Figura 4).

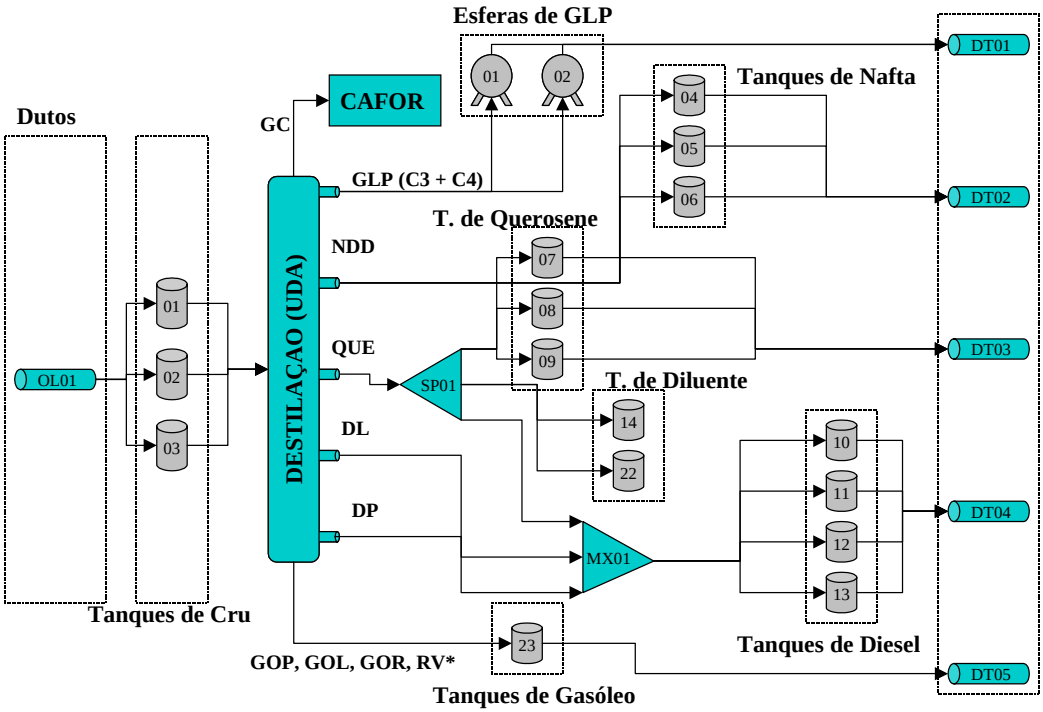


Figura 4. Refinaria do Protótipo.

Foram estudados cenários com informações de planejamento diferentes tais como itens, campanhas de unidades e manutenção de equipamentos. O número de dias do cenário (horizonte de programação) também foi variado para mostrar o comportamento do algoritmo diante do aumento do número de atividades.

O estado inicial da planta (conteúdo dos tanques e esferas da refinaria) é apresentado na Tabela 1.

Tabela 1. Estado inicial da planta.

Equipamento	Mínimo	Máximo	Atual	Composição	Densidade (g/cm^2)	Enxofre (%m)
TQ01	5.000	65.000	45.000	70% Petróleo 1 20% Petróleo 2 10% Petróleo 3	0,8927	0,4963
TQ02	5.000	65.000	60.000	100% Petróleo 2	0,9290	0,5500
TQ03	5.000	65.000	30.000	20% Petróleo 2 80% Petróleo 3	0,9130	0,6695
TQ04	1.500	17.500	1.550	Nafta	0,7100	0,0015
TQ05	1.500	17.500	6.500	Nafta	0,7100	0,0015
TQ06	1.500	17.500	10.000	Nafta	0,7100	0,0015
TQ07	1.500	14.500	1.510	Querosene	0,8230	0,1300
TQ08	1.500	14.500	1.510	Querosene	0,8230	0,1300
TQ09	1.500	14.500	11.000	Querosene	0,8230	0,1300
TQ10	2.000	32.000	2.010	Diesel	0,8800	0,4500
TQ11	2.000	32.000	2.010	Diesel	0,8300	0,3100
TQ12	2.000	32.000	7.000	Diesel	0,8670	0,4000
TQ13	2.000	32.000	12.000	Diesel	0,8900	0,5000
TQ14	85	1.400	95	Diluente	0,8500	0,1300
TQ22	85	1.400	95	Diluente	0,8500	0,1300
TQ23	6.000	90.000	26.000	Diluente	0,9200	0,7500
EF01	100	2.850	110	GLP	0,5500	-
EF02	100	2.850	1.600	GLP	0,5600	-

Nesta planta foram consideradas duas campanhas para a unidade de destilação. A primeira (Normal) tem como objetivo a produção de querosene de aviação. A segunda (Qcap) envolve a produção de diluente asfáltico. Para cada campanha foram definidos rendimentos de cada cru (fração volumétrica) e propriedades nas correntes de saída.

A seguir podemos ver um dos cenários criados para avaliação do desempenho do algoritmo co-evolucionário.

Informações de Planejamento

Tempo	1/1/03				2/1/03						3/1/03						4/1/03						5/1/03						6/1/03	
Itens de Envio	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h
GLP	250	250	250	250	250		500						500						500						500					
Nafta	5000							3000						3000						3000						3000				
QAV	6000							2400						2400						2400						2400				
Diesel 2			5000	2000				4000		2400				4000		2400				4800						4800				
Diesel 1					2000																									

Itens de Recebimento	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h
PET1						18000			18000																36000					
PET3												36000						36000												

Campanhas	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h
UDA	Normal						Qcap						Normal						Qcap						Normal					

Manutenção	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h	8h	12h	16h	20h	0h	4h
-																														

Figura 5. Cenário de teste.

O cenário apresentado na Figura 5 é de 5 dias e serve para avaliar o algoritmo na programação de um volume de atividades próximo ao real. Quase todas as características da planta apresentada acima, assim como algumas informações de planejamento apresentadas em cada um dos cenários, foram retiradas de Smania (2002).

4. Resultados

4.1. Análise de Desempenho

Foram realizados experimentos com o algoritmo co-evolucionário e com a busca aleatória, utilizando a mesma representação e heurísticas presentes no algoritmo genético, exceto os operadores genéticos. Para se ter uma idéia da dificuldade deste problema, a busca aleatória, em 5 experimentos de 6.000 indivíduos (5.600s ou 1h34min), nunca conseguiu encontrar uma solução que atendesse os itens sem atraso e desvio.

A seguir, os gráficos de desempenho: função objetivo (Figura 6) e objetivos separados (Figuras 7 a 10).

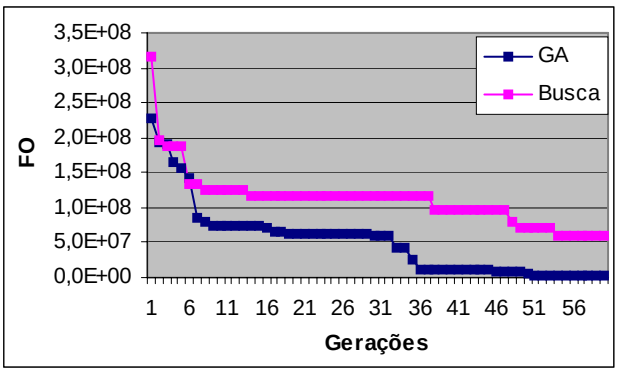


Figura 6. Desempenho do Algoritmo em relação à busca aleatória.

Ao se examinar cada um dos aspectos separadamente, tem-se a impressão que a busca aleatória está indo tão bem quanto ou até melhor que o algoritmo evolucionário. Todavia, ao se combinar cada um dos objetivos, fica claro que as soluções são inferiores às apresentadas pelo algoritmo genético. A evolução dos objetivos na busca aleatória apresenta um comportamento "guloso", tentando atender os objetivos que sejam ao mesmo tempo fáceis (com poucas combinações possíveis) e com um grande impacto na função objetivo. Já no algoritmo genético, a convergência ocorre de forma diferente, mais gradual, levando em conta todos os objetivos: primeiro realizando uma busca mais global no espaço das soluções e menos gananciosa para, então, quando está próximo das melhores soluções, fazer uma busca local nos objetivos mais sutis, como minimização das trocas de tanques.

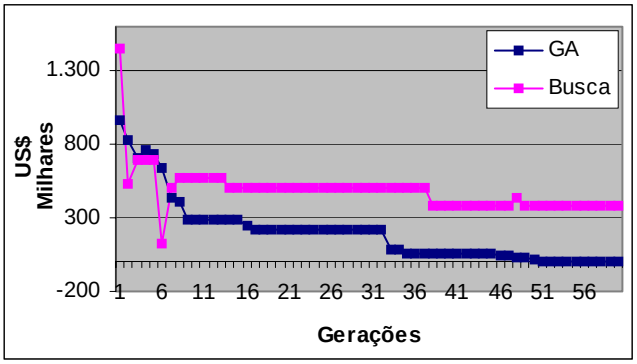


Figura 7. Custo por atraso nos itens

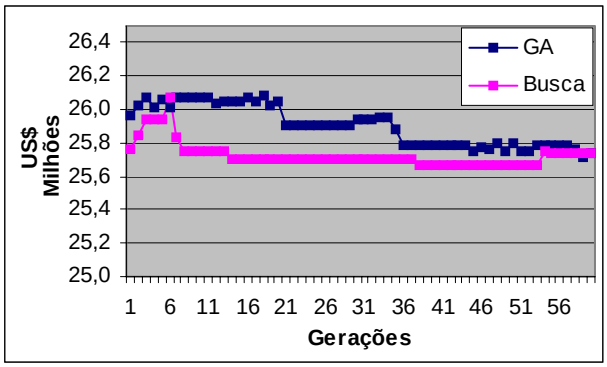


Figura 8. Custo de matéria-prima

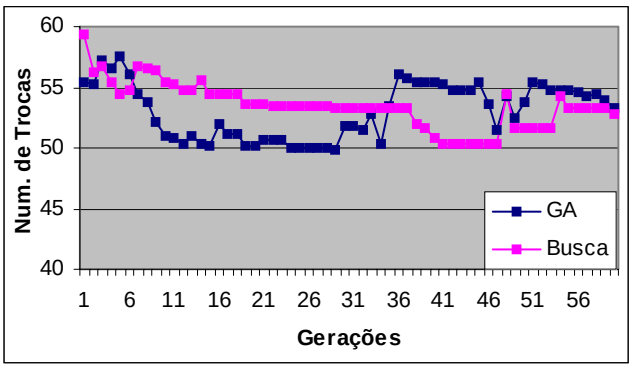


Figura 9. Trocas de tanques

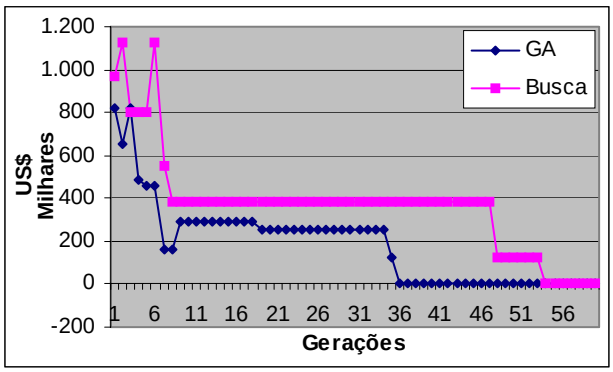


Figura 10. Custo por desvio de especificação

É possível observar por estes gráficos que o algoritmo genético, em média, teve um desempenho bem superior à busca aleatória. É importante mencionar que, mesmo para cenários maiores como este, o tempo de processamento não torna o uso do algoritmo inviável, como na maioria das abordagens para problemas de *scheduling* do mundo real.

A seguir é apresentada a Tabela 2, com os resultados das melhores soluções encontradas em 60 gerações de 100 indivíduos, para o algoritmo genético co-evolucionário.

Tabela 2. Desempenho geral da melhor solução encontrada.

Num. Gerações	60
População	100
Volume Máx. – Itens (m^3)	6.000
Volume Máx. – Carga UDA (m^3)	12.000
Atividades Geradas	228
Tempo (s)[1]	1.463
Itens não atendidos (US$)	0,00
Custo de Mat. Prima (US$)	25.156.357,89
Num. de Trocas	32
Desvio (US$)	0,00
Avaliação	2.547.635,7895

O parâmetro de volume máximo transferido determina o número de atividades que serão programadas pelo algoritmo e afeta o desempenho do algoritmo. Isto porque, na representação adotada, o número de genes dos cromossomos das duas espécies é dependente do número de atividades. O volume máximo de 12000 m^3 cria atividades

[1] Num PC Pentium III – 866 Mhz e 256 Mb de memória.

de carga e retirada da unidade de processo com 8 horas de duração, o que pode ser considerado pouco para a maioria das atividades de uma refinaria. Desta forma, há espaço para reduzir o tempo de processamento para cenários maiores.

5.2. Programação Gerada pelo Decodificador

Um trecho de programação gerada pelo sistema pode ser visto na Figura 11. Elas correspondem às ordens de produção que são passadas para a operação da refinaria.

```
1/1/2003 08:00:00 - 2/1/2003 08:00:00 -> Operation Mode 'Normal (01/01/03 - 02/01/03)' on the 'UDA' Process Unit.
1/1/2003 08:00:00 - 1/1/2003 12:00:00 -> Feeding Unit 'UDA' from storage 'TQ01' at a flow rate of 1500 m3/h (Volume = 6000 m3).
1/1/2003 08:00:00 - 1/1/2003 12:00:00 -> Delivering Item 'GLP-1' from storage 'EF02' to pipeline 'DT01' at a flow rate of 112.5 m3/h (Volume = 450
m3).
1/1/2003 08:00:00 - 2/1/2003 04:00:00 -> Delivering Item 'QAV' from storage 'TQ09' to pipeline 'DT03' at a flow rate of 300 m3/h (Volume = 6000 m3).
1/1/2003 08:00:00 - 2/1/2003 04:00:00 -> Delivering Item 'NPTQ' from storage 'TQ05' to pipeline 'DT02' at a flow rate of 250 m3/h (Volume = 5000
m3).
1/1/2003 08:00:00 - 1/1/2003 12:00:00 -> Sending (Gas Processing) from 'UDA' to 'CAFOR' at a flow rate of 1.8 m3/h (Volume = 7.2 m3).
1/1/2003 08:00:00 - 1/1/2003 12:00:00 -> Sending (Store LPG) from 'UDA' to 'EF01' at a flow rate of 39 m3/h (Volume = 156 m3).
1/1/2003 08:00:00 - 1/1/2003 12:00:00 -> Sending (Store Naptha) from 'UDA' to 'TQ04' at a flow rate of 207 m3/h (Volume = 828 m3).
1/1/2003 08:00:00 - 1/1/2003 12:00:00 -> Sending (Store Gasoil) from 'UDA' to 'TQ23' at a flow rate of 766.2 m3/h (Volume = 3064.8 m3).
1/1/2003 08:00:00 - 1/1/2003 12:00:00 -> Sending (Split Kerosene) from 'UDA' to 'SP01' at a flow rate of 174 m3/h (Volume = 696 m3).
1/1/2003 08:00:00 - 1/1/2003 12:00:00 -> Sending (Store Kerosene) from 'SP01' to 'TQ07' at a flow rate of 174 m3/h (Volume = 696 m3).
```

Figura 11. Trecho de ordens de produção

6. Conclusões

O objetivo deste trabalho foi mostrar a aplicabilidade de um modelo co-evolucionário cooperativo na otimização da programação de produção em refinarias de petróleo.

Um modelo co-evolucionário cooperativo foi adotado por decompor o problema e favorecer o desempenho do sistema. O decodificador implementado para a geração das programações a partir dos cromossomos utilizou duas espécies para programar as atividades no horizonte de tempo, respeitando sempre as restrições de precedência (já resolvidas pelo grafo da primeira espécie) e, principalmente, as outras restrições operacionais da planta, tais como: manutenção de equipamentos, tempo de preparação de produtos, volume disponível para enviar, espaço disponível para receber, entre outras. Utilizando tal abordagem não foi necessário penalizar, corrigir ou descartar indivíduos inválidos, o que tornou bem mais eficiente a evolução. A função de avaliação dos indivíduos foi concebida incorporando vários objetivos importantes na programação de uma refinaria.

Uma refinaria fictícia contendo todos tipos de atividades e restrições encontradas no dia-a-dia de uma refinaria real, foi criada para validar e avaliar o desempenho do modelo desenvolvido.

Os resultados obtidos comprovam a eficácia do algoritmo co-evolucionário, tanto se forem quantitativamente comparados aos da busca aleatória, quanto se forem avaliados como soluções viáveis e de qualidade na prática, geradas em tempo aceitável.

Finalmente, este trabalho demonstra que o emprego de computação evolucionária na otimização da programação do refino é uma alternativa viável, prática e eficiente na solução deste desafiante problema.

7. Referências

HAX, A.C.; CANDEA, D. Production and inventory management. Englewood Cliffs: Prentice-Hall, 1984.

MICHALEWICZ, Z. Genetic algorithms + data structures = evolution programs. New York: Springer-Verlag, 1996.

MORO, L. F. L. Técnicas de Otimização Mista Inteira para o Planejamento e Programação de Produção em Refinarias de Petróleo. 2000. Tese de Doutorado, USP, São Paulo, 2000.

SMANIA, P. Técnicas de Otimização Mista-Inteira Aplicadas à Programação da Produção em Refinarias de Petróleo. 2002. Dissertação de Mestrado, Escola Politécnica da Universidade de São Paulo, São Paulo, 2002.