Relatório - Aplicação do ACO

Giovanne da Silva Santos

12 de janeiro de 2021

Resumo

O gás lift é uma das técnicas artificiais mais comum de injeção para a produção de óleo. E isto está vinculado com o aumento de velocidade e precisão na injeção de gás neste petróleo. Contudo, a aplicação demasiada de gás em certo poços, prejudicará na produção de óleo por conta da pressão e atrito que o gás causará nos dutos em que o óleo percorre, assim como o desperdício de gás que afetará o produtor de petróleo economicamente. Além disso, caso não aplique-se gás suficiente, então não haverá pressão suficiente para ejetar o óleo presente no poço. Por conta disso, a resolução do problema de otimização do gás lift é essencial na produção de petróleo a fim de minimizar os custos e maximizar a produção.

1 Introdução

Caso um determinado reservatório de petróleo não consegue emergir os fluídos até a superfície, então uma técnica de injeção faz-se necessário, mas tal técnica terá de ser aplicada da melhor forma e otimizada possível para fins econômicos. Uma técnica artificial de injeção reduz a pressão da coluna do fluido causando, assim, uma redução da pressão do fundo poço. Por conta disso, uma grande diferença de pressão entre o reservatório e o fundo do poço é criada, e, consequentemente, os fluidos irão emergir até a superfície. O processo de gás lift é umas das técnicas artificiais de injeção mais utilizadas. No gás lift, o gás é injetado na parte inferior da tubulação, e o óleo é produzido através da impulsão feita pela expansão do gás e na redução de pressão hidrostática da coluna do fluido que está dentro do poço.

O algoritmo desenvolvido neste trabalho é o método de enxame de partícula, ou PSO. No qual é uns dos principais métodos da computação baseado no comportamento da natureza. Tal técnica será embasada e explicada na seção de fundamentação teórica, assim como o pseudo-algoritmo. Além disso, será mostrado o algoritmo implementado, baseado nesse método, na seção de descrição do algoritmo implementado e as análises e conclusões dos resultados deste algoritmo na seção de resultados obtidos.

2 Descrição do problema

No problema de otimização de gás lift, temos dois cenários. O primeiro cenário consiste em maximizar a produção de óleo dado uma quantidade suficiente disponível de gás para injeção, e o segundo cenário tem o objetivo de minimizar a quantidade de gás dado uma quantidade de óleo produzido pré-planejada. A otimização, de um modo geral, leva em conta com a curva de performance do gás lift e fatores econômicos.

Para o cenário em que queremos maximizar a produção de óleo e tendo uma quantidade suficiente de gás, temos a seguinte função objetiva:

$$max \sum_{1}^{n} Q_{o_i} = f(Q_{g_1}, Q_{g_2}, ..., Qg_i), i = 1, 2, ..., n$$

sendo que a quantidade total de gás injetada nos poços deverá ser igual ou menor que a quantidade de gás disponível conforme é representado nesta equação (Hamedi[3]):

$$\sum_{1}^{n} Q_{g_i} \leq A$$

onde Q_{o_i} é quantidade de óleo produzido por poço, n é a quantidade de poços, Q_{g_i} é quantidade de gás injetada em cada poço e A é a quantidade de gás disponível.

No segundo cenário, que é o cenário estudado neste trabalho, queremos minimizar quantidade de gás injetado e tendo uma quantidade de produção de petróleo já pré-determinada, teremos:

$$min \sum_{1}^{n} Q_{g_i}, i = 1, 2, ..., n$$

em que a quantidade total de petróleo será igual a quantidade já pré-determinada, como mostrado a seguir:

$$\sum_{1}^{n} Q_{o_i} = f(Q_{g_1}, Q_{g_2}, ..., Qg_i) = B$$

sendo B a quantidade pré-determinada.

tendo também estas restrições em ambos os cenários:

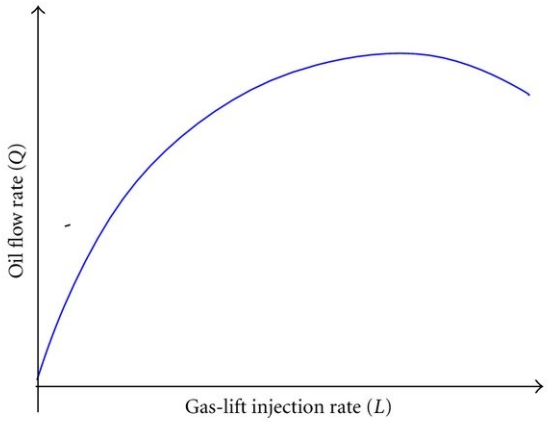
$$Q_{g_i} \geq Q_{g_i min}$$

$$Q_{g_i} \leq Q_{g_i max}$$

Além disso, vale salientar que alguns poços produzem petróleo naturalmente ou que precisam de muito pouco gás injetado, assim como temos poços que não respondem bem à injeção de gás. Ademais, há os poços "mortos"no qual é necessária uma quantidade miníma de gás igual ou maior do que $Q_{g_i min}$ para

poder produzir petróleo. Assim como a quantidade de gás não pode exceder em um poço, ou seja, ser maior do que Q_{g_imax}, a fim de não haver o decaimento da produção de petróleo por conta da percar de pressão que pode ocasionar na injeção do gás.

A quantidade máxima Q_{g_imax} e miníma Q_{g_imin} são obtidas através curva de performance do gás lift(GLPC) é ilustrado na imagem a seguir:



Em que temos no eixo y a produção de petróleo e no eixo x a quantidade de gás injetado. Como podemos ver, a quantidade ideal de gás injetado em um poço, deverá ser no pico da curva caso tenhamos uma quantidade de gás suficiente, senão, teremos que, ao menos, determina uma quantidade miníma de gás de tal modo que produza a maior quantidade possível ou pré-determinada de petróleo.

Para representar a curva GLPC, temos o modelo introduzido por Alarco et al[1] para gerar a equação que calcula a produção de petróleo a partir de uma gás injetado que é:

$$Q_o = a + b \times Q_g + c \times Q_g^2 + d \times \ln(Q_g + 1)$$

, em que a, b, c e d são os coeficientes que cada poço irá ter.

3 Fundamentação teórica

O método "*ant colony optimization*"(ACO), ou método de otimização da colônia de formiga é uma heurística baseada em probabilidade, criada para solução de problemas computacionais que envolvem procura de caminhos em grafos. Este algoritmo foi inspirado na observação do comportamento das formigas ao saírem de sua colônia para encontrar comida.

Com o transcorrer do tempo, entretanto, as trilhas de feromônio começam a evaporar, reduzindo, assim, sua força atrativa. Quanto mais formigas passarem por um caminho predeterminado, mais tempo será necessário para o feromônio

da trilha evaporar. Analogamente, elas marcharão mais rapidamente por sobre um caminho curto, o que implica aumento da densidade de feromônio depositado antes que ele comece a evaporar. A evaporação do feromônio também possui a vantagem de evitar a convergência para uma solução local ótima: se a evaporação não procedesse, todas as trilhas escolhidas pelas primeiras formigas tornar-se-iam excessivamente atrativas para as outras e, neste caso, a exploração do espaço da solução delimitar-se-ia consideravelmente.

Todavia, quando uma formiga encontra um bom (curto) caminho entre a colônia e a fonte de alimento, outras formigas tenderão a seguir este caminho, gerando assim feedback positivo, o que eventualmente torna um determinado caminho mais interessante. A ideia do algoritmo da colônia de formigas é imitar este comportamento através de "formigas virtuais"que caminham por um grafo que por sua vez representa o problema a ser resolvido.

No problema do caixeiro viajante, cada formina coloca um feromônio em uma cidade i assim que a visita, em seguida é escolhido uma próxima cidade j com uma probabilidade baseada na distância entre i e j dada como d_{ij} e a densidade do feromônio. Tal feromônio que está no caminha entre i e j é representado como τ_{ij}. Podemos representar a fórmula assim:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}$$

em que t é a iteração atual, $(1-\rho)$ é o valor de evaporação do feromônio entre a iteração t e $t+1$ e

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

Onde $\Delta\tau_{ij}^{k}$ é a mudança do feromônio entre a locomoção da formiga k na escolha da cidade j e m é o número total de formigas daquela colônia. O valor de

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{se } (i,j) \in S_k \\ 0 & \text{caso contrário} \end{cases} \tag{1}$$

Onde Q é uma constante, L_k é a distância total percorrida pela formiga k e o caminho (i,j) está no caminho S percorrido pela formiga k.

Em seguida, a próxima cidade escolhida pela formiga k será baseada na probabilidade P_{ij}^{k} que será:

$$P_{ij}^{k} = \frac{(\tau_{ij}(t))^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k\in\text{allowed}}(\tau_{ij}(t))^{\alpha}(\eta_{ij})^{\beta}}$$

Onde, η_{ij}, ou "visibilidade", é dado como $1/d_{ij}$ e d_{ij} é a distância entre duas cidade. α e β são parâmetros que controlam a importância do feromônio e da visibilidade. A partir disso, teremos o pseudo-algoritmo ACO:

Algorithm 1 ACO

```
1: procedure ACO(Grafo)
2:     for t ← 1 até o número de iterações do
3:         for k ← 1 até m do
4:             while A formiga k não construir o caminho S_k do
5:                 Selecione o próximo vértice pela regra P_ij^k
6:             end while
7:             Calcule a distância L_k do caminho S_k
8:             if L_k < L* then
9:                 S* ← S_k,
10:                L* ← L_k
11:            end if
12:        end for
13:        Atualize o feromônio global Δτ_ij
14:    end for
15:    Retorn S*
16: end procedure
```

Onde S^* é a melhor solução encontradas e L^* é o custo da melhor solução. No caso, com o número de iterações, construiremos m formigas. Em seguida, cada formiga irá fazer o trajeto S_k com base na probabilidade para escolhe o próximo vértice, ao terminar, será verificado se o caminho construído é melhor do que o atual para atualizar caso seja. No final, a formiga k atualizará o feromônio.

Para o problema do gás-lift no segundo cenário, baseado no caixeiro viajante, foi modelado para, em vez de ser uma distância entre uma cidade i e j, uma porcentagem da quantidade máxima de gás injetado naquele poço. No caso, em um poço i, teremos 20 posições que correspondem a uma porcentagem. Essas porcentagens serão 5%,10%,15%,...95% e 100% da quantidade de gás injetado para produzir a quantidade máxima de petróleo no poço i. Tal modelagem foi baseado na modelagem de Zerafat et al. [5], que no caso os vértices representam a porcentagem de gás disponível. Por conta disso, teremos a seguinte alteração matemática :

$$\tau_{pi}(t+1) = (1-\rho)\tau_{pi}(t) + \Delta\tau_{ij}$$

onde p representa o poço em que está sendo calculado o feromônio e i a posição em que se encontra uma porcentagem na lista das porcentagens. Logo, o feromônio global será

$$\Delta\tau_{pi} = \sum_{k=1}^{m} \Delta\tau_{pi}^{k}$$

em que k é a formiga atual e o $\Delta\tau_{pi}^{k}$ é o feromônio formiga que será dado por

$$\Delta\tau_{pi}^{k} = \frac{Q}{Q_g^k} \times \frac{Q_T^k}{Q_{req}}$$

sendo Q_g^k o custo total de gás aplicado pela solução construída pela formiga k, Q_T^k a produção total de petróleo produzida pela solução construída pela formiga k e Q_{req} a quantidade de petróleo pré-planejada. Por conta disso a probabilidade P_{pi}^k será

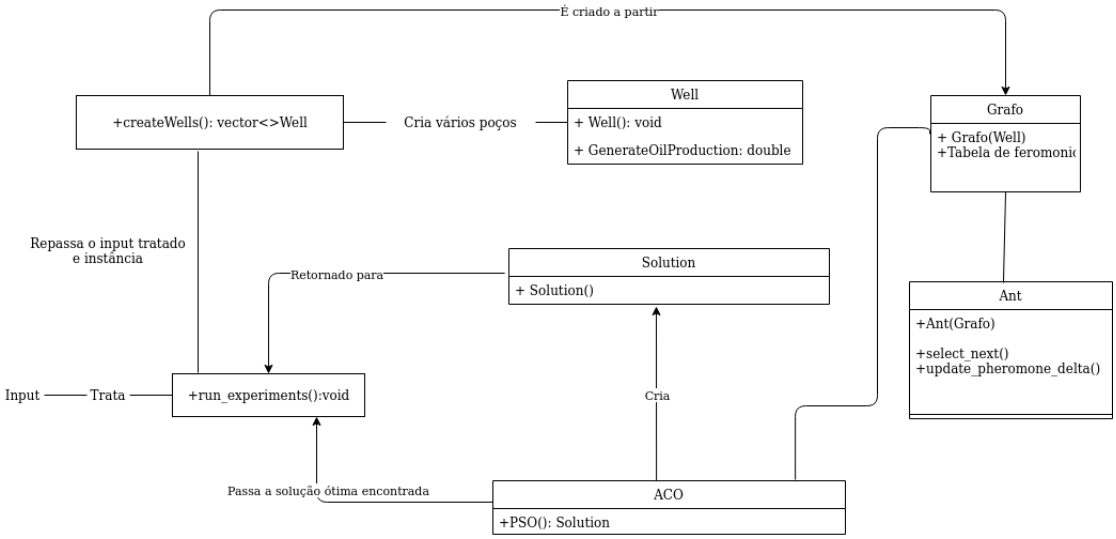
$$P_{pi}^p = \frac{(\tau_{pi}(t))^\alpha (\eta_{pi})^\beta}{\sum_k (\tau_{pi}(t))^\alpha (\eta_{pi})^\beta}$$

No caso, não haverá lista tabu, pois não há restrições de não poder repetir as porcentagens, tendo em vista que há limitação de gás disponível neste cenário.

4 Descrição do algoritmo implementado

Neste seção será explicado as classes e funções mais relevantes para resolução do problema utilizando as técnicas explicadas na seção passada.

Teremos o seguinte caso de uso do algoritmo:



Em relação ao tratamento das entradas, criação e consulta dos poços(wells), serão o mesmo dos trabalhos passado. A diferença que a classe Solution será mais simples, recebendo apenas um vetor de solução, o custo e produção dele. A utilidade será apenas para exibição dos resultados obtidos.

Primeiramente, falaremos da classe Grafo que terá apenas o construtor mostrado a seguir:

Algorithm 2 Grafo

```
1:  procedure GRAFO(wells)
2:      Inicialize a matrizFeromonio com tamanho igual ao nº de poços
3:      Inicialize a matrizGas injetado com tamanho igual ao nº de poços
4:      Inicialize a matrizPetróleo produzido com tamanho igual ao nº de poços
5:      for até que i = nº de poços do
6:          matrizFeromonio[i].resize(20)
7:          matrizGas[i].resize(20)
8:          matrizPetróleo[i].resize(20)
9:          percent = 0.05
10:         for até que j = 20 do
11:             matrizGas[i][j] = percent * wells[i].maxGasWell
12:             matrizPetróleo[i][j] = wells[i].getOilProduction(matrizGas[i][j])
13:             matrizFeromonio[i][j] = 0.4
14:             percent += 0.05
15:         end for
16:     end for
17: end procedure
```

Da linha 2, temos a inicialização da tabela de feromônios global que será alterado na classe ACO mais à frente. Em seguida, na linha 3 e 4, é inicializado as tabelas que conterão os valores de cada vértice presente no grafo que serão percorridos pelas formigas. No caso, nessas três tabelas, teremos que cada linha representará um poço i e em cada poço teremos 20 vértices que representarão 20 porcentagens como já mencionado na seção passada. Por conta disso, cada linha linha/poço i terá um tamanho i como é feito da linha 6 à 8. Na linha 9 teremos uma variável auxiliar que servirá para o cálculo do gás que será injetado baseado na porcentagem daquele vértice j. Na linha 11, temos o cálculo do gás injetado com base na porcentagem do vértice atual vezes o gás necessário para produção máxima de petróleo daquele poço i, ou seja, o gás máximo que deverá ser aplicado naquele poço, em seguida, na linha 12, temos o cálculo do petróleo produzido com base no gás resultante que está no vértice j. Tal cálculo é feito com base na fórmula apresentada na seção Descrição do problema. Na linha 13, temos o valor inicial de todos o feromônios que será 0.4 com base no parâmetro dado em Zerafat et al.[5]. Por fim, há o incremento da variável auxiliar para calcular a próxima porcentagem e, consequentemente o próximo vértice. A complexidade será O(n*20), sendo n o número de poços.

Na classe Ant, teremos o seguinte pseudo-algoritmo:

Algorithm 3 Ant

```
1:  procedure ANT(grafo,q,α, β, ρ)
2:      colony ← q,α, β, ρ
3:      ant.grafo ← grafo
4:      Inicialize a matriz η com tamanho igual ao nº de poços
5:      Inicialize a matriz matrizFeromonioDelta com tamanho igual ao nº de
    poços
6:      Inicialize o vetor de índice com o tamanho igual ao nº de poços
7:      for até que i = nº de poços do
8:          η[i].resize(20)
9:          matrizFeromonioDelta.resize(20)
10:         for até que j = 20 do
11:             if grafo.matrizGas[i][j] > 0 then
12:                 η[i][j] ← 1 / grafo.matrizGas[i][j]
13:             else
14:                 η[i][j] ← 0
15:             end if
16:             matrizFeromonioDelta[i][j] ← 0
17:         end for
18:     end for
19:     poçoAtual ← 0
20: end procedure
```

No construtor da formiga, será recebido o grafo e as constantes como parâmetro. Na linha 3 será inicializado a matriz η que guardará a visibilidade de cada vértice por parte da formiga. Já na linha 4 é inicializado a matriz $\Delta\tau_{pi}^{k}$ que terá o feromônio que a formiga aplicará no trajeto dela. Em seguida, na linha 5, teremos a inicialização do vetor de índice que guardará os índices das porcentagens visitadas pela formiga. Na linha 7 e 8, temos um procedimento semelhante ao pseudo-algoritmo anterior, e na linha 10 à 14, temos uma atribuição de 1/quantidade de gás injetado da porcentagem à visibilidade daquela porcentagem, mas caso essa quantidade seja zero, então teremos visibilidade zero à aquele vértice. Por fim, temos uma variável auxiliar na linha 18 que auxiliará na verificação do poço atual. A complexidade será, também, O(n*20), sendo n o número de poços.

Neste caso, teremos uma complexidade O(n), pois será necessário percorrer o vetor da nova solução para atualizar a quantidade de gás injetado e a quantidade de petróleo produzido.

Agora teremos a função select_next também presenta na classe Ant:

Algorithm 4 Ant

1: **procedure** SELECT_NEXT
2: denominador = 0
3: **for** até i = 20 **do**
4: denominador += grafo.matrizFeromonio[poçoAtual][i]$^{\alpha}$ × η[poçoAtual][i]$^{\beta}$
5: **end for**
6: **for** até i = 20 **do**
7: probabilidade = grafo.matrizFeromonio[poçoAtual][i]$^{\alpha}$ × η[poçoAtual][i]$^{\beta}$
8: **end for**
9: **Escolha** aleatoriamente um índice i em matrizGas com base na probabilidade
10: vetor de índice.push_back(i)
11: poçoAtual += 1
12: **end procedure**

Aqui, a formiga escolherá o índice da porcentagem no poço atual. Primeiramente, é calculado o valor do denominador que é

$$\sum_{k} (\tau_{pi}(t))^{\alpha} (\eta_{pi})^{\beta}$$

. Em seguida, será calculado a probabilidade que será

$$(\tau_{pi}) \times (\eta_{pi})^{\beta} / \text{denominador}$$

em que p é o poço atual e i o índice de porcentagem daquele poço. Por fim, será escolhido aleatoriamente um índice com base na probabilidade. A complexidade será O(20) pois só será verificado os 20 intervalos daquele poço.

Na esma classe, teremos a seguinte função:

Algorithm 5 Ant

1: **procedure** UPDATE_PHEROMONE_DELTA
2: **for** até i = tamanho do vetor de índice **do**
3: pheromone_delta[i][vetor de índice[i]] = (Q/gás total) * Produção total/Qtd pré-planejada
4: **end for**
5: **end procedure**

No caso, essa função irá, com base nos índices de porcentagem escolhido, ou seja, no caminha percorrido pela formiga, será atualizado o feromônio dessas porcentagens a partir do custo do gás injetado do caminho e da produção de petróleo. Teremos a complexidade O(tamanho do vetor de índice)

Por fim, teremos a classe ACO que terá um construtor, que apenas recebe o número de iterações, número de formigas e as constantes, a função update_pheromone e a função solve que:

Algorithm 6 ACO

```
1: procedure UPDATE_PHEROMONE(grafo, formigas)
2:     while  percorrer a matriz grafo.pheromone do
3:         grafo.pheromone[i][j] *= (1-ρ)
4:         while percorrer todas as formigas do
5:             grafo.pheromone[i][j] += formiga.matrizFeromonioDelta[i][j]
6:         end while
7:     end while
8: end procedure
9: procedure SOLVE(Grafo)
10:    for t ← 1 até o número de iterações do
11:        for k ← 1 até m do
12:            while A formiga k não construir o caminho S_k do
13:                Selecione o próximo vértice pela regra P_ij^k
14:            end while
15:            Calcule a distância L_k do caminho S_k
16:            if L_k < L* then
17:                S* ← S_k,
18:                L* ← L_k
19:            end if
20:            k.update_pheromone_delta()
21:        end for
22:        update_pheromone()
23:        Atualize o feromônio global Δτ_ij
24:    end for
25:    Retorn S*
26: end procedure
```

Na primeira função, temos a atualização do feromônio global baseado na fórmula e evaporação e intensificação do feromônio e teremos complexidade O(nº de poços*20*nº de formigas). No solve, temos o mesmo princípio do ACO explicado na seção de fundamentação teórica, sendo que ao fim do trajeto da formiga k, há atualização $\Delta\tau_{pi}^{k}$ e no fim da iteração a atualização do $Delta\tau_{pi}$. Nesta função teremos complexidade O(iterações * (nº de formigas * (tamanho da solução + complexidade do update_pheromone_delta) + complexidade do update_pheromone)).

5 Descrição dos experimentos computacionais

As instâncias utilizadas foram retiradas no Buitrago et al[2], lá temos duas tabelas, um com 6 poços e outras com 56 poços, além de que foi criado uma instância própria com 24 poços. Em run_experiments, é feito a leitura em do txt que tem, na primeira linha o número de poços, e a sequência, em seguida, da linha de gás injetado em u poço e na próxima linha o petróleo produzido no poço

com base nos valores do gás injetado na linha anterior e assim sucessivamente com cada dupla de linha representando um poço. A saída do programa grava em um arquivo e imprime no termina o número de poços da instância, quantidade de petróleo pré-planejada, o vetor da solução gerada, petróleo produzido, a quantidade mínima e a máxima de petróleo possível e o gás injetado a partir da solução encontrada. A partir dessas instâncias, utiliza-se um método gaussiano para calcular os coeficientes de cada poço que irão calcular a quantidade de petróleo baseado no gás injetado no poço. Foi utilizado os parâmetros de Zerafat et al.[5]

No. of ants	τ_0 (initial phermone)	ρ_0 (evaporation rate)	Iteration
30	0.4	0.2	100

O ambiente de teste foi um notebook Lenovo com processador Intel Core i7-7500U CPU 2.70GHz x 4, memória RAM de 3,6 GB e HD de 500 GB e sendo o sistema operacional o Ubuntu 20.0.

Para a compilação do programa, através do terminal, basta digitar "make"na pasta do programa. Para execução do programa, será necessário entrar na pasta "build"e digitar ./exe.out [nome da instâncias]. Até o momento, temos as instâncias well-six.txt, well-fifty-six.txt e well-twenty-four.txt.

6 Resultados obtidos: análise e discussão

Nesta seção será analisado os experimentos através de tabelas e, para efeitos de comparação, será mostrado a quantidade de gás injetado pela melhor solução encontrada e a produção de petróleo produzido. Para efeito de comparação, será comparado os resultados obtidos com o PSO(enxame de partícula).

Primeiro, será apresentado os resultado com 6 poços que terá uma produção mínima de petróleo de 1772.9, utilizando 1587 de gás, e máxima de 4172.3, logo:

Qtd de petróleo pré-planejada	Média do petróleo produzido	Média de gás injetado
4772.9	3855.78	8128.08
3772.9	3773.69	6863.55
1772.9	1787.69	1592.11

Tabela 1: Resultados com 6 poços utilizando PSO

Qtd de petróleo pré-planejada	Média do petróleo produzido	Média do gás injetado
4772.9	3069.87	3401.29
3772.9	3021.75	2130.46
1772.9	1819.34	823.703

Tabela 2: Resultados com 6 poços utilizando ACO

Vemos que para as quantidades de petróleo de 4772.9 e 3772.9, temos que as quantidades de petróleo pré-planejadas foram respeitadas utilizando em PSO, mas no ACO não foram encontradas quantidades superiores, contudo as quantidades de gás injetado são bem inferiores no ACO. Já para 1772.9, tivemos desempenho superior com ACO, tanto na quantidade de petróleo quanto na quantidade de gás injetado sendo bem inferior.

Os resultados com 24 poços que tem quantidade mínima de produção de petróleo de 10269, que utiliza 1564 de gás, e máximo de petróleo de 14222:

Qtd de petróleo pré-planejada	Média de petróleo produzido	Média de gás injetado
13623	13118.1	18486.7
12623	12624.1	11029
10623	10629	2343.77

Tabela 3: Resultados com 24 poços utilizando PSO

Qtd de petróleo pré-planejada	Média do petróleo produzido	Média do gás injetado
13623	13935	8259.29
12623	13012.8	7129.67
10623	11433.9	4725.94

Tabela 4: Resultados com 24 poços utilizando ACO

Já para 24 poços, temos desempenho superior na técnica ACO em todas as quantidades de petróleo pré-planejada, pois em todos foram encontradas quantidades de petróleo mais elevada em quantidades de gás injetadas mais inferiores, com exceção para quantidade de 10623, que apesar da quantidade de petróleo achada pela ACO ser superior, a quantidade de gás injetada pela PSO é menor.

Já para os resultados da instância com 56 poços que tem quantidade mínima de produção de petróleo de 18163 com injeção de gás de 23425 e máxima de 25114:

Qtd de petróleo pré-planejada	Média de petróleo produzido	Média de gás injetado
22475	22460.2	43477
21475	21500.3	36069.7
19475	19489.8	26311.1

Tabela 5: Resultados com 56 poços utilizando PSO

Qtd de petróleo pré-planejada	Média do petróleo produzido	Média do gás injetado
22475	16076.5	21498
21475	15611	21353.9
19475	15640.5	21043.9

Tabela 6: Resultados com 56 poços utilizando ACO

Já para a instância 56, tivemos um desempenho pior com ACO, tendo em vista que não foi chegado a nenhum valor de petróleo pré-planejado. Já no PSO, tivemos que foi achado e ultrapassado os valores pré-planejado, embora no ACO tenha achado custos de gás menores. O que pode-se suspeitar é que as formigas no ACO, irão tender a soluções com custo de gás mais baixo e a condição de produção de petróleo seja prejudicada em instâncias maiores, tendo em vista que o número de caminho possível será maior.

7 Conclusão

Os problemas de otimização associados ao gás lift são bastante estudados, principalmente, por conta do valor econômico que traz com a resolução de tais problemas. Por conta disso, temos várias meta-heurísticas e modelagens aplicadas a fim de trazer melhores resultados. Contudo, o segundo cenário ainda não tem muitos estudos envolvendo ele, tornando-se necessário criar e aplicar novos algoritmos para a resolução como Namdar[4] mencionou.

Como podemos ver, o algoritmo ACO é poderoso com instância pequenas e médias, sendo superior ao ACO. Contudo, em instância maiores, o algoritmo ACO demonstra desempenho pior em relação ao PSO.

Referências

[1] Gabriel A Alarco´ n, Carlos F Torres, and Luis E Go´ mez. Global optimization of gas allocation to a group of wells in artificial lift using nonlinear constrained programming. *J. Energy Resour. Technol.*, 124(4):262–268, 2002.

[2] S Buitrago, E Rodriguez, D Espin, et al. Global optimization techniques in gas allocation for continuous flow gas lift systems. In *SPE gas technology symposium*. Society of Petroleum Engineers, 1996.

[3] H Hamedi, F Rashidi, and E Khamehchi. A novel approach to the gas-lift allocation optimization problem. *Petroleum Science and Technology*, 29(4):418–427, 2011.

[4] Hamed Namdar. Developing an improved approach to solving a new gas lift optimization problem. *Journal of Petroleum Exploration and Production Technology*, 9(4):2965–2978, 2019.

[5] Mohammad M Zerafat, Shahab Ayatollahi, and Ali A Roosta. Genetic algorithms and ant colony approach for gas-lift allocation optimization. *Journal of the Japan Petroleum Institute*, 52(3):102–107, 2009.