# Relatorio - Aplicação do PSO

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 dos problemas

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:

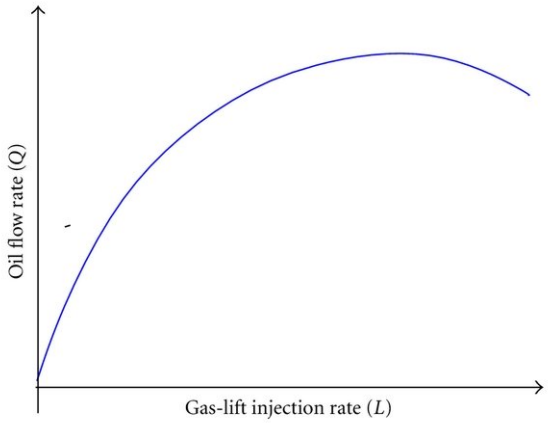
$$Q_{g_i} \geq Q_{g_i min}$$

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

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.

Contudo, como mostrado em Hamedi[3], essa fórmula apresenta problema quando chegamos a valores muitos altos de injeção de gás, pois eles acabam crescendo exponencialmente a partir de uma certa quantidade. Por conta disso, foi sugerido por ele, o uso do seguinte cálculo:

$$Q_o = a + b\sqrt{Q_g} + cQ_g$$

e será ela q utilizaremos para o cálculo de produção de petróleo.

# 3 Fundamentação teórica

O método "*particle swarming optimization*"(PSO), ou método de otimização por enxame de partícula, é baseado em padrões da natureza, pois é inspirado

pelo comportamento social e cooperativo exibido por várias espécies de forma a realizar as suas necessidades no espaço de soluções. Em termos gerais, o algoritmo guia-se por experiência pessoal *Pbest*, experiência geral *Gbest* e o movimento das partículas atual para decidir as posições seguintes no espaço de pesquisa. O PSO resolve um problema criando uma população de soluções candidatas, também conhecidas como partículas, e movendo estas partículas em torno do espaço de pesquisa, de acordo com fórmulas matemáticas simples sobre a posição e velocidade da partícula. O movimento de cada partícula é influenciado pela sua posição do local mais conhecida, mas, também é guiado em direção às posições mais conhecidas do espaço de pesquisa, que são atualizadas como posições melhores quando encontradas por outras partículas. No caso, isso é esperado quando o intuito é mover o enxame em direção da melhor solução.

Sendo k a iteração atual da partícula e k + 1 a próxima iteração, teremos que a próxima posição de uma partícula será

$$x_{k+1}^i = x_k^i + v_{k+1}^i$$

, $v_{k+1}^i$ será a velocidade que é dado por

$$v_{k+1}^i = w_k v_k^i + c_1 r_1 (p_k^i - x_k^i) + c_2 r_2 (p_k^g - x_k^i)$$

No caso, $p_k^i$ será a melhor posição que a partícula achou (Pbest), $p_k^g$ a melhor posição conhecida pelo enxame de partícula, $w_k$ será uma constante de inércia, $c_1$ e $c_2$ serão os parâmetros cognitivos e sociais respectivamente, e, por fim, $r_1$ e $r_2$ serão número aleatórios entre 0 e 1. Note que a locomoção das partículas são determinada por uma soma de vetores em que um representa a inércia da própria partícula, a melhor posição achada pela partícula e a melhor posição achada pelo enxame.

Com isso, será mostrado o pseudo-algoritmo do PSO:

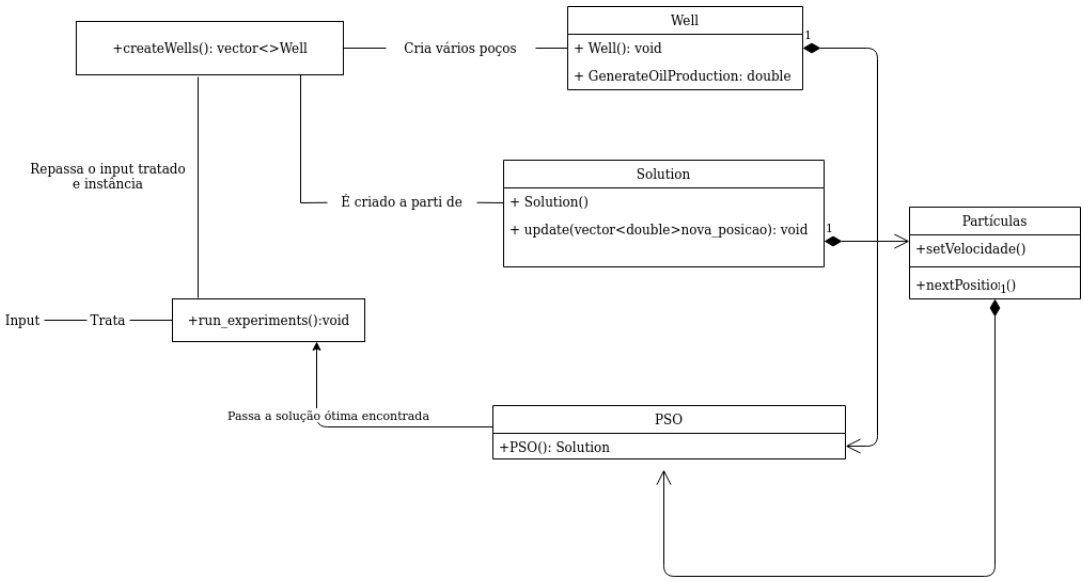
**Algorithm 1** PSO

```
1: procedure PSO(P. w_k, c_1, c_2)
2:     Inicialize aleatoriamente as posições das partículas de P
3:     Inicialize aleatoriamente a velocidade de cada partícula
4:     while Não atingir o número de iterações do
5:     end while
6: end procedure
```

# 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.

Primeiramente será mostrado um diagrama que ilustra o caso de uso do algoritmo.




O input recebido - que é um txt contendo, na primeira linha, a quantidade de poços, e em seguida uma sequência de linha com os valores de gás injetado e outra linha com petróleo produzido baseado nos valores da linha do gás - é convertido pela função run experiments em uma matriz de gás injetado e outra de petróleo produzido baseado em cada elemento da matriz de gás, e que cada linha dessas matrizes representam um poço. Na mesma função, temos a instância e repasse dessas matrizes à função createWells que, a partir do construtor de Well, mapeia as matrizes no respectivo poço. Na classe Well, temos o construtor que a partir de um vetor gás e um vetor petróleo, calcula os coeficientes daquele poço com base em um método de resolução de sistema linear e, também, há o cálculo do petróleo produzido a partir do gás injetado.

A classe solução terá um construtor Solution() que inicializará aleatoriamente os elementos do vetor solução, que, no caso, será a posição da partícula e cada elemento será a injeção de gás no poço. No caso, será o número de poço será o número de dimensões do espaço de busca. A inicialização pegará uma faixa de 50% a 80% da taxa de gás de cada poço, e escolherá aleatoriamente para a construção do da posição. O pseudo-algoritmo será demonstrado a seguir:

**Algorithm 2** Solution

```
1: procedure SOLUTION(wells,número de poços)
2:     s_0 ← {∅}
3:     while Faça até a solução ser construída do
4:         a ← wells.tamanho()*0.5
5:         b ← wells.tamanho()*0.8
6:         Selecione aleatoriamente um elemento de valores entre a e b
7:         Faça s_0 ← s_0 ∪ {s}
8:     end while
9:     return s_0
10: end procedure
```

A construção da solução será O(n), tendo em vista que n é o tamanho do vetor de solução e não precisará fazer nenhuma outro laço além do da construção. Na mesma classe teremos uma função chamada update() que terá o seguinte código:

**Algorithm 3** update

```
1: procedure UPDATE(nova_solucao)
2:     Atualize a quantidade de gás injetado com base da nova solução
3:     Atualize a quantidade de petróleo produzido com base da nova solução
4: end procedure
```

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 classe Partícula que irá ter o Pbest da partícula, a posição atual, solução do tipo Solution, velocidade e as funções setVelocidade() e next-Position(), no qual o pseudo-código é mostrado a seguir:

**Algorithm 4** setVelocidade e nextPosition

```
1: procedure SETVELOCIDADE(c1, c2, inertiaWeight,Gbest)
2:     r1 ← Escolha um número aleatório no intervalo (0,1)
3:     r2 ← Escolha um número aleatório no intervalo(0,1)
4:     while Percorra todo o vetor posição do
5:         velocidadeCognitiva ← c1 × r1 × (Pbest_i − posicao_i)
6:         velocidadeSocial ← c2 × r2 × (Gbest_i − posicao_i)
7:         posicao_i ← inertiaWeight ×velocidade_i + velocidadeCognitiva + velocidadeSocial
8:     end while
9: end procedure
10: procedure NEXTPOSITION
11:     while Percorra todo o vetor posição do
12:         posicao_i ← posicao_i + velocidade_i
13:         if posicao_i > maxGasRate then
14:             posicao_i ← maxGasRate
15:         end if
16:         if posicao_i < 0 then
17:             posicao_i ← 0
18:         end if
19:     end while
20:     if posicao < Pbest then
21:         Pbest ← posicao
22:     end if
23: end procedure
```

No caso, aplicaremos a fórmula para obter a velocidade de cada elemento do vetor da posição atual baseada na fórmula da mostrada na seção de fundamentação teórica. Em seguida, calcularemos a nova posição somando cada elemento da posição com a nova velocidade, e caso o elemento da nova posição exceda o limite da taxa de gás de injeção, tal elemento terá o valor dessa taxa, assim como se ele acabar sendo menor do que zero, então receberá o valor de 0. Por fim, é verificado se essa nova posição é a melhor posição daquela partícula, caso seja, então Pbest passa a ser essa nova posição, senão, apenas ignora. A complexidade de ambas as funções será O(n) sendo n o tamanho do vetor da posição.

Por fim, teremos a classe PSO que instância as partículas e aplica a técnica de enxame de partícula de fato. Sendo o pseudo-código:

**Algorithm 5** PSO

```
1: procedure PSO(wells,c1, c2, inertiaWeight, numeroDeParticulas)
2:     vetor_particulas ←{⊘}
3:     while Enquanto o número de particulas não é atingido do
4:         vetor_particulas ← vetor_particulas ∪ Particula()
5:     end while
6:     Gbest ← Melhor Pbest atual
7:     while Faça até atingir o número de iterações do
8:         while Faça até percorrer todas as partículas do
9:             vetor_particulas_i.setVelocidade(c1, c2, inertiaWeight,Gbest)
10:            vetor_particulas_i.nextPosition()
11:            if ( thenvetor_particulas_i.Pbest < Gbest)
12:                Gbest ← vetor_particulas_i.Pbest
13:            end if
14:        end while
15:    end while
16:    return Gbest
17: end procedure
```

O PSO começa inicializando as partículas quem por sua vez são inicializadas em posições aleatórias através do construtor Solution(), e cada partículas será guardada como $vetor\_particulas_i$ em $vetor\_particulas$. Em seguida, é inicialidado o Gbest como o melhor Pbest atual e, depois, é iniciado o processo de enxame que será feito baseado na quantidade de iterações colocado como parâmetro. A cada iteração é percorrido todas as partículas $vetor\_particulas_i$ e atribuído a nova posição da partículas. Depois, é verificado se há um Pbest melhor que o Gbest atual, ou seja, é verificado se há uma posição em que o custo de gás injetado é menor e que a quantidade de petróleo pré-planejada é respeitada. No fim, é retornado o Gbest. Por conta disso,do processo de setVelocidade e nextPosition, teremos a complexidade dada como O(Nº de iterações × Nº de partículas ×(n+n)), sendo n o tamanho do vetor posição ou da solução.

# 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 [5]

Parameters of the PSO model for solving Nishikiori's (1989) five-well problem

| Number of particles | $\alpha$ | $C_1$ | $C_2$ | Iterations |
|---|---|---|---|---|
| 10 | 1 | 0.1 | 0.1 | 90 |

Além disso, foi testado também parâmetros semelhantes com a diferença de 900 iterações e 100 partícula. Neste caso chamaremos de parâmetros próprios(PP) para efeito de comparações futuras. E é comparado aos resultado do algoritmo GRASP implementado pelo autor.

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

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(Nishikiori) | Média de gás injetado(Nishikiori) |
|---|---|---|
| 2086.15 | 2127.23 | 2187.06 |
| 3172.3 | 3216.68 | 3942.44 |
| 1772.9 | 1802.97 | 1627.33 |

Tabela 1: Resultados com 6 poços utilizando os parâmetros de Nishikiori

| Qtd de petróleo pré-planejada | Média do petróleo produzido(PP) | Média do gás injetado(PP) | Média do petróleo produzido(GRASP) | Média de gás injetado(GRASP) |
|---|---|---|---|---|
| 2086.15 | 2090.68 | 2023.08 | 2806.8 | 3367.74 |
| 3172.3 | 3183.56 | 3783.65 | 2544.45 | 2352.83 |
| 1772.9 | 1778.82 | 1593.69 | 2535.41 | 2324.51 |

Tabela 2: Resultados com 6 poços utilizando os parâmetros próprios

Na tabela vemos ótimo resultados nos dois parâmetros em relação ao GRASP, tendo em vista que a quantidade de petróleo achado não ultrapassa tanto da

quantidade de petróleo pré-planejada, assim como não ultrapassa muito a quantidade de gás mínima possível em relação a quantidade de petróleo produzida pela solução encontrada pelo GRASP. Contudo, utilizando os parâmetros próprios com 900 iterações e 100 partículas, ou seja, dez vezes mais do que o parâmetros de Nishikiori, temos uma redução considerável da quantidade de petróleo achado da pré-planejada e na quantidade de gás injetado. Sendo a quantidade de gás injetado para a quantidade mínima possível de petróleo mais próxima da quantidade mínima de gás e bem melhor do que a GRASP.

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 |
|---|---|---|
| 7288 | 10269.7 | 1601.33 |
| 13576 | 13107.7 | 18151 |
| 10269 | 10270.4 | 1605.15 |

Tabela 3: Resultados com 24 poços utilizando os parâmetros de Nishikiori

| Qtd de petróleo pré-planejada | Média do petróleo produzido(PP) | Média do gás injetado(PP) | Média do petróleo produzido(GRASP) | Média de gás injetado(GRASP) |
|---|---|---|---|---|
| 7288 | 10269.2 | 1575.2 | 13047.3 | 38414.2 |
| 13576 | 13065.2 | 18784.5 | 13034.8 | 38378.6 |
| 10269 | 10269.3 | 1576.58 | 13050.2 | 38422.9 |

Tabela 4: Resultados com 24 poços utilizando os parâmetros próprios

Aqui, temos resultados com uma grande diferença em relação ao GRASP, contudo, vale salientar que quando a quantidade de petróleo pré-planejada abaixo da quantidade mínima possível, temos uma quantidade de petróleo produzido muito próximo dessa quantidade mínima. Porém, a quantidade de gás injetado para essa quantidade de petróleo produzido ainda é um pouco alta, embora utilizando os parâmetros próprios tenha encontrado uma quantidade de gás injetado mais próximo. Em relação ao GRASP, temos uma diferença notória em relação ao resultado do PSO utilizando ambos os parâmetros.

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 |
|---|---|---|
| 13306 | 18164.2 | 23783.5 |
| 25612 | 23255.4 | 56610.4 |
| 18163 | 18171.2 | 23871.1 |

Tabela 5: Resultados com 56 poços utilizando os parâmetros de Nishikiori

| Qtd de petróleo pré-planejada | Média do petróleo produzido(PP) | Média do gás injetado(PP) | Média do petróleo produzido(GRASP) | Média de gás injetado(GRASP) | |
|---|---|---|---|---|---|
| 13306 | 18163.5 | 23670.5 | 21783.4 | 105771 | |
| 25612 | 23251.1 | 55681.1 | 22942 | 105687 | |
| 18163 | 18163.2 | 23467 | 18163 | 50986.9 | |

Tabela 6: Resultados com 56 poços utilizando os parâmetros próprios

Os resultados para 56 poços também tivemos uma minimização a quantidade de gás injetado assim como os de 24 poços. No caso, também podemos ver que a quantidade de gás injetado com parâmetros próprios foram menor do que os parâmetros de Nishikiori, contudo a quantidade não tem diferenças muito grandes. Porém, podemos ver que a minimização é maior mas não tão grande neste caso, ficando talvez melhor utilizar parâmetro menores para não ter custos computacionais maiores, assim como se preferimos minimizarmos, principalmente em instâncias muito grandes, então será melhor termos um número maior de iterações e de partículas. Em relação ao GRASP, vemos que quanto maior a instância, maior a diferença dos resultados encontrados entre o GRASP e o PSO.

# 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 do PSO é muito poderoso pois trouxe resultados com custo de gás mais próximo do mínimo e que, em instância grandes, o uso de parâmetros maiores sejam melhores para minimizar mais ainda a quantidade de gás injetado. Vemos também a superioridade em relação ao GRASP principalmente em instâncias maiores. O fato de utilizarmos vários pontos de busca e avanço do espaço de busca, melhorou na procura de ótimo globais. Tendo em vista que o GRASP faz uma busca mais simples utilizando uma única posição para encontrar uma solução ótima.

# 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] Nobuo Nishikiori. *Gas allocation optimization for continuous flow gas lift systems*. PhD thesis, University of Tulsa, 1989.