Recebido: 22/07/10
Aprovado: 09/09/11

Algoritmo memético com *vocabulary building* para o problema de roteamento de unidades móveis de pistoneio

João Paulo Lima do Nascimento (Programa de Pós-Graduação em Engenharia de Produção) - jp_astron@yahoo.com.br
• Universidade Federal do Rio Grande do Norte, Centro de Tecnologia
Campus Universitário, Lagoa Nova, s/n, Natal-RN
Dario José Aloise (Departamento de Informática - Mestrado em Ciência da Computação - UERN/UFERSA) - aloisedj@gmail.com

RESUMO O presente trabalho apresenta um algoritmo evolutivo memético com *vocabulary building* para a resolução do Problema de Roteamento de Unidades Móveis de Pistoneio - PRUMP. Por se tratar de um problema de otimização combinatória NP-Difícil, uma modelagem matemática para o problema foi utilizada, permitindo obter a solução exata de um conjunto de instâncias testes, as quais foram usadas nos experimentos computacionais e apresentaram resultados competitivos, comprovando a eficiência do método proposto e validando a estratégia metaheurística proposta.

Palavras-chave Algoritmo Memético; *Vocabulary Building*; Roteamento de Veículos; Unidades Móveis de Pistoneio; Otimização Combinatória; NP-árduo.

ABSTRACT *This paper presents an evolutionary memetic algorithm with vocabulary building to solve the Routing Problem of Mobile Oil Recovery Units. Because this is an NP-Hard problem in combinatorial optimization, a mathematical model for the problem was used, allowing for the exact solution of a set of test situations, which were used in computational experiments and demonstrated competitive results, proving the efficiency of validating the proposed metaheuristic method and strategy.*

Keywords *Memetic Algorithms; Vocabulary Building; Vehicle Routing; Mobile Oil Recovery Units; Combinatorial Optimization; NP-hard.*

NASCIMENTO, J. P. L.; ALOISE, D. J. Algoritmo memético com vocabulary building para o problema de roteamento de unidades móveis de pistoneio. **GEPROS. Gestão da Produção, Operações e Sistemas**, Ano 7, nº 1, jan-mar/2012, p. 81-95.

1. INTRODUÇÃO

Atualmente, é importante para qualquer empresa se manter competitiva no mercado global, procurando a perfeita utilização de seus recursos, investindo em pesquisas em seu setor de atuação e evitando gastos desnecessários. Com essa intenção, empresas do ramo petrolífero, ao longo de décadas, veem investindo em tecnologias de produção e exploração de petróleo no Brasil, otimizando seus recursos para melhor desenvolver seus produtos.

A exploração de petróleo no Brasil acontece tanto em campos terrestres como em campos marítimos. Nesse trabalho, é dada ênfase à exploração de petróleo em campos terrestres, onde existe toda uma logística no transporte de óleo e gás através de uma frota de veículos, desde sua origem nos campos de exploração até o consumidor final, que deve ser considerada.

Otimizando a logística de transporte terrestre e exploração de petróleo, obtêm-se ganhos maiores na produção total, uma vez que se economiza com os gastos provenientes de locomoção e a exploração do produto é feita de maneira eficiente.

Os poços terrestres de petróleo da bacia potiguar são explotados há cerca de 40 anos, sendo esse um dos motivos pelo qual a grande maioria, em torno de 98% deles, utiliza-se de algum método de elevação artificial. Um desses métodos é a utilização de um veículo equipado para coleta e transporte de petróleo, denominado por Unidade Móvel de Pistoneio – UMP. Este veículo encontra-se inicialmente num depósito de onde parte em busca dos poços para pistoneio, e após conclusão da jornada de trabalho, retorna ao depósito de origem. O problema em questão neste artigo é uma generalização do problema de otimização do emprego de uma única UMP. Portanto, consiste em traçar as melhores rotas considerando mais de uma UMP, de forma que seja maximizada toda a produção de petróleo respeitando-se as jornadas de trabalho.

Na literatura, este problema com uma única UMP é conhecido por: Problema de Otimização do Emprego de Unidades Móveis de Pistoneio – POE-UMP. Neste trabalho, o problema abordado caracteriza-se como um Problema de Roteamento de Unidades Móveis de Pistoneio – PRUMP, pela ênfase à aplicação a mais de uma UMP.

No presente trabalho, estratégias metaheurísticas fundamentadas nos conceitos dos algoritmos evolutivos, mais especificamente no Algoritmo Memético – AM, e na técnica heurística conhecida por *Vocabulary Building* – VB, foram desenvolvidos para a resolução de instâncias do Problema de Roteamento de Unidades Móveis de Pistoneio – PRUMP. Os resultados obtidos por essas estratégias heurísticas são comparados com os resultados obtidos pela modelagem de Programação Inteira formulada para o problema.

O restante deste trabalho encontra-se organizado da seguinte forma: seção 2 – trata o Problema de Roteamento de Unidades Móveis de Pistoneio – PRUMP; seção 3 – aborda o desenvolvimento das Estratégias Metaheurísticas aplicadas ao PRUMP; seção 4 – apresenta os Testes Computacionais e Resultados Obtidos; e por fim, a seção 5 – expõe as Conclusões.

2. O PROBLEMA DE ROTEAMENTO DE UNIDADES MÓVEIS DE PISTONEIO – PRUMP

Diariamente, é definido para cada UMP um trajeto contendo *n* poços para pistoneio. Estes poços situam-se numa região denominada Campo de Produção, que dependendo de seu tamanho, pode conter de um poço até centenas de poços, todos formando um conjunto conexo. O mais importante no problema abordado não é explotar qualquer poço neste conjunto conexo, mas sim operar as UMP's sobre um conjunto deles que proporcione um bom retorno em termos de total de óleo produzido. Idealmente, o objetivo deve ser o de buscar produzir o máximo possível de óleo a cada jornada diária de trabalho. Isso caracteriza o problema como sendo de otimização combinatória.

A operação de pistoneio, entretanto, está sujeita a um conjunto definido de restrições, a saber:

- **A quantidade de UMPs:** um número *m* de UMP's é definido para pistoneio, para um determinado campo de produção, de maneira que, quanto mais unidades são disponibilizadas, maior será a produção diária.
- **Quantidade de Depósitos:** neste trabalho, é considerado apenas um único depósito, conhecido por Estação de Tratamento de Óleo – ETO. No caso, as UMP's iniciam nesse depósito, dirigem-se aos poços designados e depois retornam a mesma ao final da jornada de trabalho.
- **A capacidade de transporte de uma UMP:** uma UMP possui um pequeno tanque que suporta até 5m^3 de óleo. Entretanto, para que a mesma não retorne à ETO antes do término da jornada de trabalho, é associado à mesma um caminhão-tanque de apoio, cuja função é a de esvaziar a UMP e transportar o óleo explotado para a ETO. Dessa forma, pode-se considerar que a capacidade de armazenamento do tanque das UMP's seja ilimitada.
- **Jornada diária de uma UMP:** cada UMP deve seguir o percurso fechado que lhe é destinado, respeitando o tempo total de sua jornada de trabalho. Então cada UMP *k* possui uma jornada de trabalho (L_k). Nos campos da Bacia Potiguar, as jornadas diárias são de 8 ou 16 horas (L_k = 480 minutos ou 960 minutos, respectivamente).
- **O tempo do trajeto da *k-ésima* UMP de um poço *i* para um poço *j*:** o tempo que a UMP *k* gasta para se deslocar entre os poços ou entre um poço e a ETO deve ser considerado para compor o tempo total da rota a ser cumprida e este, por sua vez, é limitado pela jornada diária L_k. Esse tempo é definido no modelo matemático como t_{ij}.
- **Tempo de atuação de uma UMP k num poço *i*:** a ação de uma UMP *k* sobre um poço *i* qualquer corresponde ao esvaziamento desse poço e o tempo que dura essa ação que consiste em três etapas chamadas de: Tempo de montagem, Tempo de operação e Tempo de desmontagem. Para este trabalho considera-se a soma dos tempos dessas três etapas, que será definido como t'_i (tempo total de atuação de uma UMP *k* num poço *i*).
- **Produção num poço *i*:** o total líquido de óleo a ser extraído num poço *i* é definido no modelo como p_i, sendo o volume medido em m^3.

Os caminhos existentes entre os poços (estradas), assim como os poços e o depósito (ETO) são modelados como um grafo onde os vértices são os poços; o depósito, é o vértice inicial de índice zero (p_0); e os caminhos existentes entre cada vértice são os arcos. A seguir um resumo das definições apresentadas anteriormente:

Dados de entrada:
m – número de UMP's;
n – número total de poços;
p_i – produção de óleo no poço i;
t'_i – tempo de operação (montagem, exploração e desmontagem) no poço i;
L_k – jornada de trabalho definida para uma UMP k;
t_{ij} – tempo de trajeto entre os poços i e j.

Neste trabalho, a modelagem matemática para o PRUMP segue os modelos apresentados em Neves e Didier (2007) e Aloise, Santos e Duhamel (2009). A obtenção da solução ótima através do modelo exato é obtida em duas fases: maximização da produção de óleo (FASE 1), e minimização dos tempos de percurso (FASE 2). Ambos os modelos são descritos a seguir.

2.1. Modelagem matemática do PRUMP

Variáveis de decisão:

$$x_{ik} = \begin{cases} 1, \text{ se o poço i é explotado pela UMP k, para i = 1, ... , n e k = 1, ... , m} \\ 0, \text{ caso contrário} \end{cases} \quad (1)$$

$$r_{ijk} = \begin{cases} 1, \text{ se os poços são consecutivos no roteamento da UMP k, para i, j = 0,..., n e k = 1,..., m} \\ 0, \text{ caso contrário} \end{cases} \quad (2)$$

2.2. FASE 1

$$Maximizar\ P = \sum_{k=1}^{m} \sum_{i=1}^{n} p_i x_{ik} \quad (3)$$

Sujeito às restrições:

$$\sum_{i=1}^{n} t'_i x_{ik} + \sum_{i=0}^{n} \sum_{j=0}^{n} r_{ijk} t_{ij} \leq L_k, \qquad \text{para } k = 1, \ldots, m \quad (4)$$

$$\sum_{j=0}^{n} r_{jik} - x_{ik} = 0, \qquad \text{para } i = 1, \ldots, n \text{ e } k = 1, \ldots, m \quad (5)$$

$$\sum_{j=0}^{n} r_{ijk} - x_{ik} = 0, \qquad \text{para } i = 1, \ldots, n \text{ e } k = 1, \ldots, m \quad (6)$$

$$\sum_{j=1}^{n} r_{i0k} = 1, \qquad \text{para } k = 1, \ldots, m \quad (7)$$

$$\sum_{j=1}^{n} r_{0jk} = 1, \qquad \text{para } k = 1, \ldots, m \quad (8)$$

$$\sum_{k=1}^{m} x_{ik} \leq 1, \qquad \text{para } i = 1, \ldots, n \quad (9)$$

$$\sum_{\substack{j=0 \\ j \neq i}}^{n} y_{ijk} - \sum_{\substack{j=0 \\ j \neq i}}^{n} y_{jik} = x_{ik}, \qquad \text{para } i = 1, \ldots, n \text{ e } k = 1, \ldots, m \quad (10)$$

$$y_{ijk} \leq (l+1) r_{ijk}, \qquad \text{para } i, j = 0, 1, \ldots, n \text{ e } k = 1, \ldots, m \quad (11)$$

$$x_{ik}, r_{ijk} \in \{0,1\} \text{ e } y_{ijk} \geq 0 \ \forall\ i, j, k \text{ com } i \neq j \quad (12)$$

A função objetivo, apresentada na equação 3, maximiza a produção de óleo obtida pelas *m* UMPs num campo de produção com *n* poços em um único dia. O total produzido *P* é apresentado na unidade de medida de volumes (m^3).

As restrições representadas na equação 4 limitam o tempo total de produção das UMPs (tempo de operação mais tempo de trajeto entre os poços) a uma jornada diária atribuída a cada UMP *k*. As restrições nas equações 5 e 6 condicionam os números de entrada e saída de arcos de um poço. Caso ele seja pistoneado por uma UMP *k* ($x_{ik} = 1$), haverá apenas um arco entrando e outro saindo para este poço. Caso contrário, não haverá qualquer arco entrando ou saindo desse poço. As restrições nas equações 7 e 8 limitam a apenas um arco entrando e saindo do depósito (ETO). As restrições na equação 9 configuram apenas uma UMP para cada poço. As restrições nas equações 10 e 11, em atuação conjunta com as restrições nas equações de 5 a 9, ajudam a descartar possíveis subciclos ou sub-rotas inválidas. E as restrições contidas na equação 12 configuram como variáveis binárias x_{ik} e r_{ijk}, e a variável inteira y_{ijk} como não-negativa.

2.3. FASE 2

$$Minimizar\ T = \sum_{i=1}^{n} t'_i x_{ik} + \sum_{i=0}^{n} \sum_{j=0}^{n} r_{ijk} t_{ij} \tag{13}$$

Sujeito às restrições:
{restrições das equações de 4 a 12}

$$\sum_{k=1}^{m} \sum_{i=1}^{n} p_i x_{ik} = P^{\star} \tag{14}$$

Na equação 13, o objetivo é minimizar a soma dos tempos dos percursos, dado por *T*. Neste modelo matemático são inclusas as restrições apresentadas nas equações 4 a 12. E na equação 14, a restrição garante que a produção obtida *P* seja igual à produção ótima $P^{\star}$ da fase anterior.

3. ESTRATÉGIAS METAHEURÍSTICAS APLICADAS AO PRUMP

Nesta seção são apresentados detalhes das estratégias utilizadas na implementação do Algoritmo Memético e da técnica *Vocabulary Building*. Na experimentação, são utilizadas duas versões de metaheurísticas evolucionárias: Algoritmo Memético – AM e Algoritmo Memético com *Vocabulary Building* – AM+VB.

3.1. Algoritmo Memético – AM

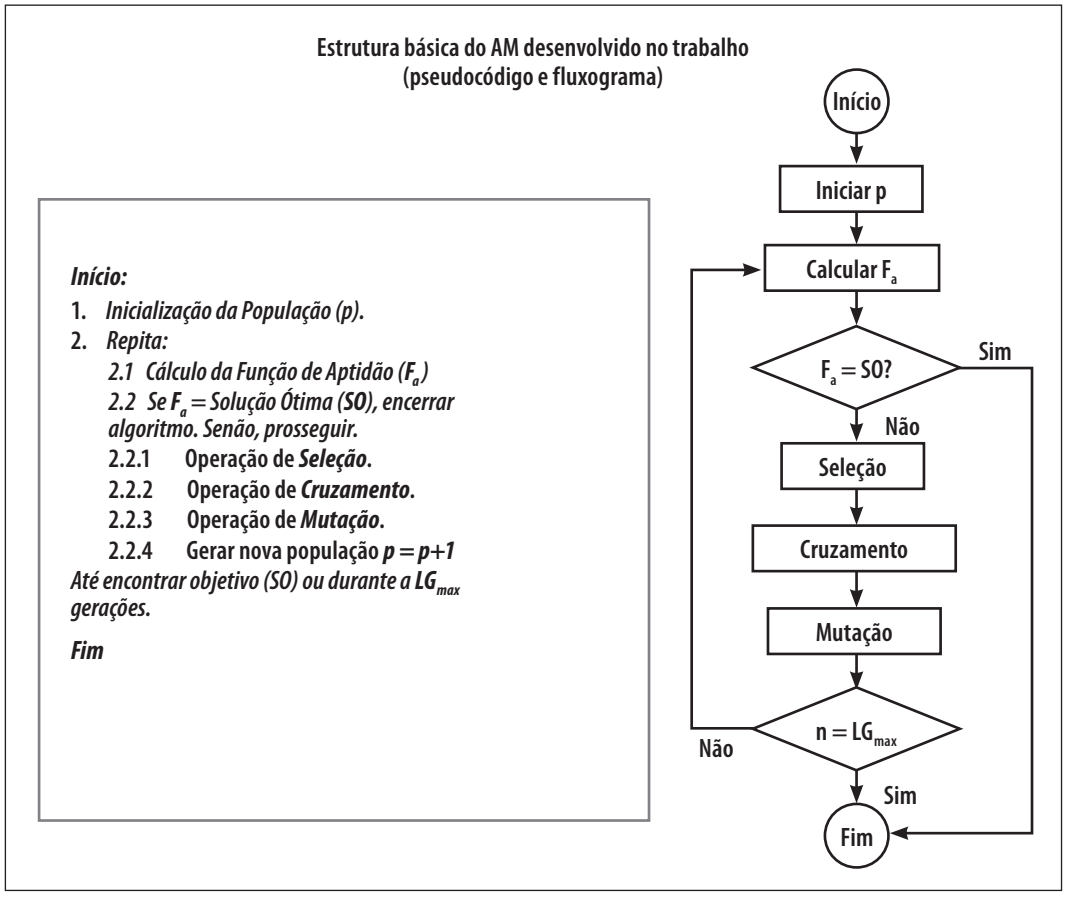
Os Algoritmos Meméticos (AM) pertencem à classe dos algoritmos evolutivos que, além de possuírem os métodos de seleção e operadores genéticos, tais como *crossover* e mutação, encontrados nos algoritmos genéticos, trabalham com o conceito de evolução cultural onde os próprios indivíduos, denominados de agentes, trocam informações culturais entre si adquiridas ao longo de sua existência. Os AMs são amplamente estudados e possuem grande aplicabilidade a vários problemas de otimização combinatória encontrados na literatura, segundo Moscato e Cotta (2003).

A estrutura básica do Algoritmo Memético desenvolvido para o PRUMP é apresentada na figura 1.

Inicialmente, o AM cria aleatoriamente uma população *P* com base nos dados de entrada (parâmetros do problema). O passo seguinte é o cálculo da função de aptidão (F_a) de cada indivíduo da população inicialmente criada. O valor de F_a é comparado ao valor objetivo (SO), caso este seja conhecido, e se forem iguais, o algoritmo encerra, atingindo seu objetivo. Caso contrário, o algoritmo prossegue para um processo de evolução que consiste em várias etapas, encerrando

quando se chega ao objetivo ou um número máximo de gerações (LG_{max}) é atingido. Durante cada etapa do processo evolutivo, operações genéticas (seleção, cruzamento e mutação) são realizadas nos cromossomos da população, e ao final é esperada uma nova população com maior valor F_a que a anterior, definindo assim a evolução genética.

Figura 1 – Estrutura básica do AM desenvolvido no trabalho.



Fonte: Elaborado pelos autores

A estrutura de cromossomo adotada para o PRUMP possui uma estrutura semelhante à de uma matriz, mas cada linha pode ter um tamanho diferente. No entanto, cada linha se trata de um vetor dinâmico, onde para cada vetor é alocado um tamanho relativo ao número de poços visitados pela UMP correspondente. Portanto, o tamanho ou quantidade de colunas para cada linha podem se diferenciar, pois cada UMP pode visitar uma quantidade igual, maior ou menor de poços em relação às outras UMPs. O número *m* de UMPs é fornecido como parâmetro de entrada, dessa forma tem-se o número de linhas do cromossomo definido. Em momento de execução do AM, uma nova coluna para dada linha é alocada, quando um novo poço é inserido na rota de uma UMP. Cada linha contém os números dos poços a serem visitados, e cada UMP inicia do depósito e retorna ao mesmo, ou seja, são considerados os percursos até o depósito. As figuras 2 e 3, demonstram, respectivamente, um exemplo de representação de cromossomo com 3 UMPs, para uma instância de 15 poços e seu grafo com as rotas traçadas.

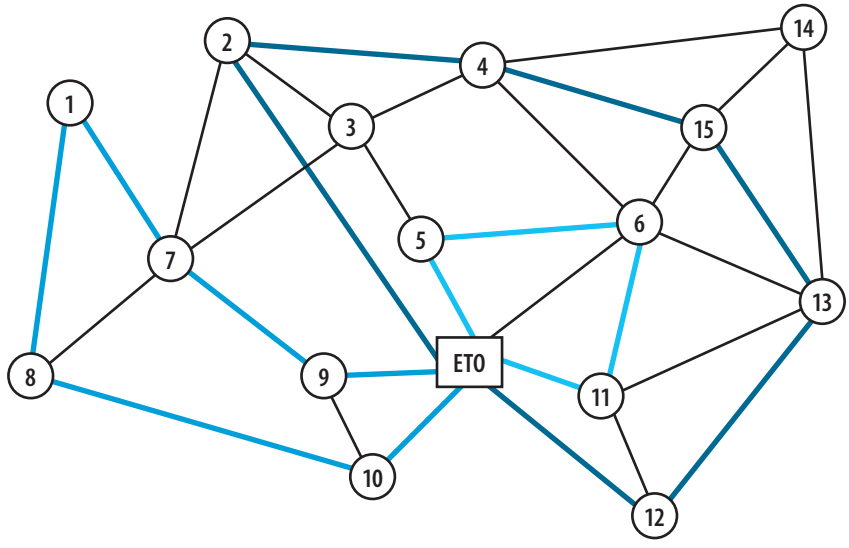
Figura 2 – Exemplo de Cromossomo para 3 UMPs.

UMP 1	9	7	1	8	10
UMP 2	5	6	11		
UMP 3	2	4	15	13	12

Fonte: Elaborado pelos autores

Cada cromossomo é alimentado com os genes que correspondem aos números de cada poço disponível para pistoneio. Foi desenvolvido no AM, outro vetor dinâmico para cada cromossomo, que contém a numeração de cada poço que ainda não pertence à rota de alguma UMP. No momento em que é alocado um novo poço, numa determinada rota do cromossomo de uma UMP, a numeração relativa a esse poço é excluída do vetor de poços disponíveis. Dessa forma, sabem-se quais poços ainda não foram pistoneados pelas UMPs.

Figura 3 – Grafo com rotas para 3 UMPs.



Fonte: Elaborado pelos autores

A função de aptidão é responsável por avaliar cada cromossomo de uma população. Cada cromossomo contém as informações sobre a produção de óleo obtida numa rota por cada UMP e a soma dos tempos de operação e de percurso da mesma. O objetivo do PRUMP é obter para cada UMP uma produção máxima possível, não ultrapassando a sua jornada L_k. Com base nesse objetivo a função de aptidão desenvolvida para avaliar cada cromossomo irá classificar os cromossomos com base em seu valor de produção por rota, mas sendo válidos aqueles que possuírem tempos de operação e de trajeto não superior às suas jornadas L_k. Esta função é definida a seguir:

$$F_a = \sum_{k=1}^{m} \sum_{i=1}^{n} p_i x_{ik}, \qquad \text{para } k = 1, \ldots, m \tag{15}$$

Na equação 15, a função F_a força a seleção de cromossomos com mais produção. Este é o critério a ser considerado durante o ordenamento dos cromossomos por valor de *fitness*.

Os métodos de seleção adotados para esse trabalho são o "elitismo" e "*ranking*". Dessa forma, os melhores cromossomos (10% deles) são mantidos (copiados para a próxima geração) por elitismo e, na parte restante, é aplicado o método *ranking* para que seja efetuada a operação de cruzamento entre os cromossomos selecionados.

A operação de cruzamento ocorre da seguinte maneira: pares de cromossomos são selecionados para reprodução através do método de seleção "*Ranking*", onde serão criados novos cromossomos (filhos) com base nas informações genéticas contidas nos cromossomos pais. As sequências dos genes em cada cromossomo representam as rotas desenvolvidas para cada UMP. Nesta operação, são definidos dois setores aleatoriamente em rotas distintas de cada par de cromossomos. Então, os genes selecionados são trocados entre si, formando dois novos cromossomos. A cada inserção dos genes num novo cromossomo, um teste de verificação é feito em busca de corrigir possíveis repetições de rotas num mesmo cromossomo, o que violaria a restrição de que cada poço só é visitado por uma única UMP. Se houver algum gene inserido que seja um mesmo já existente no cromossomo, ele será substituído por outro gene no vetor de poços disponíveis associado ao cromossomo.

A operação de Mutação seleciona aleatoriamente alguns cromossomos e aplica uma ligeira modificação em suas estruturas. Isto contribui para a diversidade da população e fuga de ótimos locais, maior abrangência do espaço de busca. Num cromossomo, uma rota é escolhida ao acaso e dela é removido um gene que será inserido numa outra rota do mesmo cromossomo. Dessa forma, tem-se o tamanho das rotas de cada UMP alterado. Outra forma desta operação é trocar algum gene por algum dos outros guardados no vetor de poços disponíveis associado ao cromossomo.

Para realizar a busca local no AM proposto, foi necessário definir as estruturas de vizinhança para que se possa partir para o melhoramento do valor de *fitness* dos indivíduos. Foram definidas duas estruturas de vizinhança: *insereGene()* e *otimizaRota()*. A primeira realiza para cada indivíduo a inserção de um gene de maior produção proveniente do vetor de poços não pistoneados, visando obter um aumento na produção de óleo de cada indivíduo. Na segunda, após a realização da primeira busca local, há uma reestruturação das rotas entre genes (poços), aplicada individualmente para melhorar (minimizar) o valor do custo da rota.

O método *insereGene()* recebe como parâmetro o indivíduo selecionado aleatoriamente e substitui o gene proveniente do vetor de poços disponíveis pelo gene de menor produção no indivíduo. Cada gene está associado a um poço visitado, que por sua vez possui um valor de produção associado. Antes da efetiva substituição de genes, é verificada a distância do gene para cada um dos genes do indivíduo a receber a cópia. Caso essa distância ultrapasse o tempo de jornada de trabalho para alguma UMP, logo é descartado e selecionado o próximo melhor gene do vetor de poços disponíveis. Após a inserção, o método *otimizaRota()* refaz as ligações entre cada gene em busca de melhorar o custo com as rotas, mantendo os mesmos genes.

Este processo de busca local melhora consideravelmente o *fitness* de cada cromossomo. Logo após a execução dos métodos de busca local, inicia-se uma nova iteração do algoritmo. É aplicado o método de Avaliação de *fitness* em cada cromossomo e logo após é aplicado o método de seleção, que irá guardar 10% dos melhores cromossomos para a próxima geração. Dessa forma, com a aplicação dos métodos de busca local e de seleção, somando-se aos métodos que diversificam toda a população, há garantias de soluções melhores a cada iteração do algoritmo.

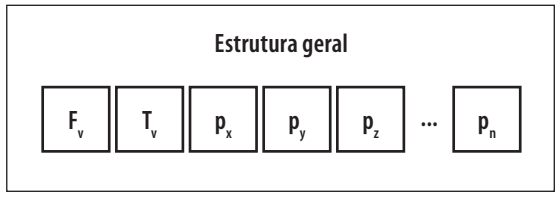
3.2. *Vocabulary Building*

A *Vocabulary Building* (VB) trata-se de uma técnica de otimização e foi idealizada por Glover (1992), dentro do contexto de *Path Relinking*. Trabalhos mais recentes acerca da VB encontram-se: em Guedes e Aloise (2006), que aplicaram VB num Algoritmo Memético – AM para o Problema do Caixeiro Viajante Assimétrico; em Soares (2008), que utilizou VB num algoritmo Busca Tabu para o Problema de Atribuição de Localidades em Anéis DH/SONET; em Silva (2008) que aplicou VB num Algoritmo Memético e Genético para o mesmo problema anteriormente; e em Neto (2009), que aplicou VB num Algoritmo Memético para o PCV Assimétrico.

A técnica VB tem por objetivo guardar os melhores vocábulos numa espécie de *pool* (vetor dinâmico contendo vocábulos) que simula uma memória adaptativa. Os melhores indivíduos de uma população podem possuir boas soluções parciais que se aplicadas a outras podem melhorar seus respectivos *fitness*. Um vocábulo para o PRUMP representa então um determinado trecho de uma rota para uma UMP que é boa em termos de tamanho mínimo e possui um alto valor de produção total dos poços presentes nesse trecho da rota.

Portanto, em cada vocábulo é importante registrar além da sequência da sub-rota, seu valor de produção (F_v – soma das produções dos poços presentes na sub-rota) e seu valor de custo (T_v – soma dos tempos de operação e de percurso entre os poços). Seguindo essa ideia, uma estrutura de vocábulo para o PRUMP é apresentada na figura 4.

Figura 4 – Estrutura de um vocábulo para o PRUMP.



Fonte: Elaborado pelos autores

Na figura 4, o valor de *fitness* do vocábulo (F_v) é representado na primeira posição (da esquerda para a direita) do vetor que representa um vocábulo. Este valor é calculado sendo a soma das produções dos poços presentes na sub-rota, como aplicado na equação 15, para cálculo de *fitness* de um cromossomo. Na segunda posição, o (T_v) representa o valor do tamanho daquela sub-rota, sendo a soma dos tempos de percurso e de operação. T_v é o segundo critério para ordenamento dos vocábulos, do menor para o maior, quando o F_v de dois ou mais vocábulos são iguais. As demais posições guardam os números dos poços. A sequência da rota é definida pela ordem (da esquerda para a direita) iniciando e retornando à ETO.

Inicialmente o *Pool* de vocábulos está vazio. Após a avaliação da população, é aplicada aos 10% melhores indivíduos a técnica VB, pois os mesmos devem possuir bons trechos. O método *registraVocabulo()* tem por função verificar em cada cromossomo os melhores trechos e guardá-los no *Pool*. Este método recebe como entrada o cromossomo a ser verificado.

É interessante manter o *Pool* vocábulos não muito grande. Dessa forma pode-se facilmente inserir os vocábulos nas outras estruturas cromossômicas que apresentem menores valores de *fitness*. Portanto, o tamanho do vocábulo é definido até a metade do tamanho da rota completa para uma UMP. E seu tamanho mínimo é 2, ou seja, uma rota entre dois poços i e j. A seguir é apresentado na figura 5 um pseudocódigo para o método *registraVocabulo()*.

Neste método, o tamanho da atual rota completa da UMP é conhecido para que a partir dele possa ser calculado o tamanho máximo do vocábulo. Considerando os limites, aleatoriamente é definido o tamanho do vocábulo a ser procurado na rota da UMP. Se por exemplo, o tamanho definido para o vocábulo for 3, dentro da rota completa da UMP será procurada a melhor sequência de três poços que possui o melhor valor de *fitness*. Logo, esse vocábulo é adicionado no *Pool*. Este processo se repete para cada rota contida no cromossomo. Neste trabalho foi definido um tamanho máximo para o pool de até 50 vocábulos. Portanto, quando houver já preenchido todo o *pool* de vocábulos, o que apresentar menor *fitness* será substituído por um novo vocábulo de *fitness* maior.

Figura 5 – Pseudocódigo do método registraVocabulo().

```
Método registraVocabulo
        Entrada: cromossomo s ∈ S
        Saída: Pool de vocábulos atualizado
início
        para i=0 até N_UMPs faça
                tam_vocabulo ← tam(s(UMP i))/2
                tam_vocabulo ← random(tam_vocabulo)
                vocabulos ← verificaVocabulo(tam_vocabulo, UMP i)
                adicionaVocabulosNoPool(vocabulos)
        fim-para
fim
```

Fonte: Elaborado pelos autores

Após a atualização do *Pool* de vocábulos, entra em execução o método *insereVocabulos()* que irá inserir vocábulos num conjunto específico de cromossomos da população. São escolhidos para inserção de vocábulos os 50% cromossomos com menor valor de *fitness*. O objetivo é inserir bons trechos de rotas para as UMPs desses cromossomos. Esse método recebe como parâmetro a lista de vocábulos (*pool*) e o cromossomo a ser avaliado. O pseudocódigo do método *insereVocabulos()* é apresentado na figura 6.

O método *insereVocabulos()* irá analisar cada rota do cromossomo procurando encontrar uma sub-rota de mesmo tamanho do vocábulo para que seja substituída por este último. A sub-rota a ser substituída será a que oferecer menor valor de *fitness* para o cromossomo. Outra condição para inserção do vocábulo na rota é que na mesma não haja poços duplicados após a inserção. É feita uma verificação no método *comparaVoc()* antes da inserção. Caso a inserção implique em duplicação de poços na rota, um próximo vocábulo é selecionado da lista. Se não houver implicações na inserção da rota, a mesma contribuirá para o melhoramento do *fitness* para aquela rota (ganho de produtividade e diminuição dos tempos de percurso e de operação sobre os poços).

Figura 6 – Pseudocódigo do método *insereVocabulos()*

```
Método insereVocabulos
        Entrada: Pool de vocábulos, cromossomo s ∈ S
        Saída: Cromossomo s atualizado
início
        para i=0 até N_UMPs faça
                para j=0 até tam_pool faça
                        tam_voc Pool(j).getTamVocabulo()
                        boolean melhor ← comparaVoc(Pool(j), tam_voc)
                        if melhor então
                                insereVocabuloNaRota()
                fim-para
        fim-para
fim
```

Fonte: Elaborado pelos autores

O método *insereVocabulos()* prossegue verificando para outras rotas das outras UMPs do cromossomo uma possível inserção de vocábulos que possa promover um aumento de *fitness*.

A técnica VB tem promovido a melhora das soluções no decorrer do processamento do AM. Sua grande vantagem é guardar "trechos" de boas soluções para que a partir destas se possam construir outras boas soluções para o problema proposto.

4. TESTES COMPUTACIONAIS E RESULTADOS OBTIDOS

Foi desenvolvido um algoritmo memético fundamentado nas definições apresentadas na subseção 3.1. O objetivo deste algoritmo é obter soluções para o PRUMP considerando uma frota k de UMPs. A implementação do AM proposto para resolução do PRUMP se deu através do uso da linguagem de programação C++ por ser mais rápida em tempo de processamento e por suportar o paradigma Orientação a Objetos, que possibilita uma fácil e eficiente implementação baseando-se em comportamentos e atributos de entidades contidas no problema abordado do presente trabalho.

No AM, foi implementado um módulo de busca local além de outro módulo para aplicação da técnica *Vocabulary Building*.

Os parâmetros utilizados para o AM foram:

- Tamanho da População(M): 30 cromossomos
- Taxa de *Crossover*: 80%
- Taxa de Mutação: 10%
- Número de Gerações: 200

Para validação do modelo e do algoritmo proposto, instâncias do PRUMP foram utilizadas. Cada instância simula um campo de extração de petróleo contendo *n* poços disponíveis para pistoneio. Portanto, são parâmetros de entrada para os métodos de resolução:

- *n*: número de poços;
- *m*: número de UMPs;
- $L_k[m]$: vetor de jornadas de trabalhos de cada UMP;
- *p[n]*: vetor de valores de produção dos poços disponíveis para pistoneio;
- *t[n]*: vetor de tempos de operação dos poços disponíveis para pistoneio;
- $t_{ij}[n+1, n+1]$; matriz de distâncias (tempos de percursos) entre cada poço, incluindo a ETO.

As instâncias foram submetidas a dois tipos de métodos de resolução: método exato e métodos heurísticos. Para resolução através de um método exato foi utilizado o *software* ILOG CPLEX 11.210 (versão *full* licenciada para a Universidade Federal do Rio Grande do Norte) que possui algoritmos exatos para tratar problemas de otimização de programação linear inteira mista. Métodos de resolução exatos proporcionam, num determinado tempo relacionado ao tamanho da instância, uma solução ótima global. Para resolução através de métodos heurísticos, é aplicado o AM desenvolvido em duas categorias: Algoritmo Genético com busca local (Algoritmo Memético – AM) e Algoritmo Memético com a técnica *Vocabulary Building* (AM+VB). Os resultados obtidos através de métodos heurísticos são próximos da solução ótima ou, em alguns casos, é igual ao ótimo obtido através de um método exato. A vantagem que se tem ao usar métodos heurísticos trata-se do tempo de processamento computacional que é bem menor quando comparado ao de um método exato. Os resultados de cada um dos métodos são comparados em termos de tempo de processamento e solução obtida.

As instâncias trabalhadas dividem-se em três grupos: instâncias testes com 11 poços (ITP11), instâncias com 99 poços (IP99) e instâncias com 200 poços (IP200), que simulam um campo real. As instâncias com 11 poços foram criadas pelo autor do presente trabalho com objetivos de testes de validação do modelo (formulação matemática para o PRUMP) e do AM proposto. As instâncias com 99 e 200 poços foram adaptadas dos trabalhos de dissertação de mestrado de Neves (2000) e Barros (2001).

Todos os testes (métodos exatos e heurísticos) foram realizados numa mesma máquina. As configurações do computador utilizado são:

- Sistema Operacional *Linux* (*Ubuntu* versão 9.10, *kernel* 2.6.31-22)
- CPU *QUAD CORE* (4 núcleos) com freqüência de 2.4 GHZ
- Memória RAM DDR2 com capacidade para 2GB

Os resultados, exatos e heurísticos, dos experimentos realizados para as instâncias de 11 poços (ITP11) são apresentados na tabela 1.

Tabela 1 – Resultados exatos e heurísticos das instâncias ITP11.

Instâncias ITP11	CPLEX			AM			AM+VB		
	P* (m^3)	TCPU (s)	T* (min)	P* (m^3)	TCPU (s)	T (min)	P* (m^3)	TCPU (s)	T (min)
1 UMP (L1=480 min)	18	0.02	466	18	15.01	466	18	22.00	466
1 UMP (L1=960 min)	35	0.08	955	35	16.99	955	35	24.78	955
2 UMPs (L1=480 min L2=480 min)	33	0.34	911	33	21.66	911	33	22.55	911
2 UMPs (L1=480 min L2=960 min)	46	0.09	1322	46	19.09	1322	46	23.76	1322
3 UMPs (L1=480 min L2=480 min L3=480 min)	46	0.79	1372	46	26.12	1372	46	34.90	1372

Fonte: Elaborado a partir dos resultados obtidos pelos autores.

De acordo com o observado na tabela 1, todos os métodos heurísticos (AM e AM+VB) atingiram a solução ótima. Seus tempos de processamento (TCPU) ocorreram todos em menos de 1 minuto. É observado, na maioria dos casos, que os tempos de execução do AM são menores quando comparados aos tempos de execução do AM+VB. Isto se deve à adição da técnica VB discutida na subseção 3.2. Estes tempos, quando comparados aos tempos do método exato, são bem maiores, sendo, portanto preferível utilizar para essas instâncias (ITP11) o método exato, que possibilitou um resultado ótimo em menos de 1 segundo.

Vale ressaltar que as instâncias ITP11 são pequenas, contendo apenas 11 poços. Para instâncias maiores, os métodos heurísticos propostos no presente trabalho são mais eficazes quando considerados seus tempos de processamento, como apresentado na tabela 2 a seguir, para as instâncias IP99.

Tabela 2 – Resultados exatos e heurísticos das instâncias IP99.

Instâncias IP99	CPLEX			AM			AM+VB		
	P* (m^3)	TCPU (s)	T* (min)	P* (m^3)	TCPU (s)	T (min)	P* (m^3)	TCPU (s)	T (min)
1 UMP (L1=480 min)	14	6.05	420	14	846.00	420	14	972.13	420
1 UMP (L1=960 min)	28.70	93.37	946	28.70	901.86	946	28.70	953.44	946
2 UMPs (L1=480 min L2=480 min)	27.70	161.9	940	27.70	876.97	940	27.70	954.19	940
2 UMPs (L1=480 min L2=960 min)	42.70	648.16	1410	38.7	814.32	1358	38.7	916.62	1320
2 UMPs (L1=960 min L2=960 min)	56.50	39601	1888	53	863.71	1792	56.50	1025.00	1888
3 UMPs (L1=480 min L2=480 min L3=480 min)	37.70	4408.3	1356	37	996.74	1284	37	1163.03	1364
3 UMPs (L1=960 min L2=960 min L3=960 min)	81*	466867	2848*	71.5	1082.00	2700	75	1235.39	2796

Fonte: Elaborado a partir dos resultados obtidos pelos autores

Novamente é observado que a técnica VB melhora o valor da produção, mesmo que consuma um pouco mais de tempo, quando comparado ao AM. Destacam-se os resultados obtidos através do AM+VB, onde de sete instâncias, quatro atingiram a solução ótima (nas 1ª, 2ª, 3ª e 5ª instâncias), e para as demais atingiram *gaps* num intervalo de 1,86% a 9,37%.

Na tabela 2, percebe-se que os tempos de execução (TCPU) obtidos do método exato são melhores que os tempos das metaheurísticas para as instâncias de até duas UMPs, mas com tempos diferentes de jornada de trabalho L_k (na 4ª instância). Mas a partir da 5ª instância os TCPU obtidos para o método exato são muito grandes, variando de 1 hora para mais de 5 dias. O melhor resultado de produção obtida (P) para a 7ª instância foi de 81m^3, sendo este um valor aproximativo da solução ótima, com *gap* de 1,36%. A não obtenção da solução ótima para essa instância através do método exato deve-se a problemas de "estouro de memória" ocorrido durante a execução. Portanto, a execução teve que ser abortada e foi registrada a melhor solução encontrada. Na fase de minimização, ainda para a 7ª instância, os tempos obtidos para cada UMP não são ótimos. São os tempos T_k realizados por cada UMP, considerando o valor de produção ótima P = 81m^3. De mesma forma, houve problemas com estouro de memória na obtenção dos tempos mínimos para essa instância.

Entretanto, os tempos (TCPU) obtidos a partir da 5ª instância, através das execuções das metaheurísticas desenvolvidas para o PRUMP, são bem menores quando comparados com os tempos obtidos através do método exato para essas mesmas instâncias. Portanto, para instâncias maiores, que condizem com a realidade trabalhada pelas empresas de exploração de petróleo, algoritmos aproximativos respondem melhor que os algoritmos exatos, principalmente pelo fator tempo de execução computacional. De acordo com a tabela 2, o tempo máximo para uma metaheurística obter um resultado próximo do ótimo, foi de 1235,39 segundos, ou seja, 20 minutos e meio, para a 7ª instância. Para essa mesma instância, o método exato obteve uma resposta, próxima do ótimo, em 466867 segundos, ou seja, em mais de cinco dias.

A tabela 3 apresenta os resultados para as instâncias contendo 200 poços (IP200):

Tabela 3 – Resultados exatos e heurísticos das instâncias IP200.

Instâncias IP200	CPLEX			AM			AM+VB		
	P* (m^3)	TCPU (s)	T* (min)	P* (m^3)	TCPU (s)	T (min)	P* (m^3)	TCPU (s)	T (min)
1 UMP (L1=480 min)	22	3114.43	479.26	22	1043.60	479.26	22	1081.15	479.26
1 UMP (L1=960 min)	44*	21017.16	960*	39	1005.00	920	42	1196.52	935
2 UMPs (L1=480 min L2=480 min)	43.60*	70273.88	957.25*	41.20	1342.30	915	41.85	1466.00	942
2 UMPs (L1=960 min L2=960 min)	88*	92705.53	1908.5*	81	1290.10	1639.2	87.4	1418.71	1830
3 UMPs (L1=480 min L2=480 min L3=480 min)	65.20*	140717.31	1428.25*	62.35	1518.80	1396	63	1683.44	1410
3 UMPs (L1=960 min L2=960 min L3=960 min)	128.20*	312592.80	---	119.60	1657.00	2564	121.30	1757.82	2752

Fonte: Elaborado a partir dos resultados obtidos pelos autores

Os resultados apresentados na tabela 3 reforçam a afirmativa de que os métodos heurísticos conseguem achar uma boa solução em tempo menor, quando comparados aos tempos obtidos pelo método exato. De um total de seis instâncias contendo 200 poços, foi somente possível achar a solução ótima para a primeira instância, devido a problemas de estouro de memória na máquina e estagnação de seus processadores. Porém, os *gaps* obtidos variam de 2,00 % a 5,00 %.

5. CONCLUSÕES

Neste trabalho, foram realizados estudos para solucionar o Problema de Roteamento de várias Unidades Móveis de Pistoneio – PRUMP com frota de 1 (uma) a 3 (três) UMPs através das Metaheurísticas Algoritmo Memético – AM e Algoritmo Memético com a técnica *Vocabulary Building* – AM+VB, com o objetivo de promover a maximização da produção de óleo em campos petrolíferos terrestres.

Foram desenvolvidas na linguagem de programação C++ as metaheurísticas propostas e as mesmas foram aplicadas às instâncias criadas especialmente para fins de testes. Outras instâncias foram obtidas através de trabalhos de dissertação correlatos que simulam o problema real. Resultados satisfatórios foram obtidos com o uso das metaheurísticas.

Através dos resultados obtidos, chegou-se à conclusão de que para instâncias maiores, os métodos heurísticos possibilitam um menor tempo de resposta para uma solução quando comparados com o método exato.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ALOISE, D. J.; SANTOS, A. C.; DUHAMEL, C. Formulações multiobjetivos para um problema de coleta de petróleo em campos maduros. In: XLI Simpósio Brasileiro de Pesquisa Operacional, Porto Seguro **Anais...**, p.1143-1154, 2009.

BARROS, C. A. **Uma aplicação de grasp na otimização do emprego da unidade móvel de pistoneio**. Master's thesis, Universidade Federal do Rio Grande do Norte, Departamento de Informática, Natal, Brasil. 2001.

GLOVER, F. New Ejection Chain and Alternating Path Methods for Traveling Salesman Problems. *In*: **Computer Science and Operations Research: New Developments in Their Interfaces** [editado por R. Sharda, O. Balci e S. Zenios], Cambridge: Elsevier, 449-509. 1992.

GUEDES, A. B. da C.; ALOISE. D. J. **Um algoritmo memético para o problema do caixeiro viajante assimétrico:** Uma abordagem baseada em vocabulary building. Sociedade Brasileira de Pesquisa operacional – SBPO. 2006.

MOSCATO, P.; COTTA, C. Una introduccion a los algoritmos memeticos. **Revista Iberoamericana de Inteligencia Artificial**, v. 19, p. 131–148. 2003.

NETO, J. S. S. **Aplicação Das Técnicas Path-Relinking e Vocabulary Building na Melhoria de Performance do Algoritmo Memético para o Problema do Caixeiro Viajante Assimétrico**. Dissertação de Mestrado – Programa de Pós-Graduação em Engenharia de Produção, UFRN. 2009.

NEVES, J. A. **Uma aplicação de Algoritmo Genético na Otimização do Emprego da Unidade Móvel de Pistoneio**. Dissertação de Mestrado, Universidade Federal do Rio Grande do Norte, Departamento de Informática e Matemática Aplicada, Natal, Brasil. 2000.

NEVES, J. A.; DIDIER, M. A. C. Modelagem de Problemas em Programação Linear Inteira. *In*: VII Encontro Regional de Matemática Aplicada – ERMAC, 2007, Recife, PE. **Anais...** Recife: SBMAC, 2007.

SILVA, A. C. G. **Busca Heurística Através de Algoritmo Genético e Memético com Construção de Vocábulos para o Problema de Atribuição de Localidades a Anéis SONET**. Dissertação de Mestrado– Programa de Pós-Graduação em engenharia de Produção, UFRN. 2008.

SOARES, W. K. da S. **Heurísticas Usando Construção de Vocabulário Aplicadas ao Problema da Atribuição de Localidades a Anéis em Redes SONET/SDH**. Dissertação de Mestrado – Programa de Pós-Graduação em Engenharia de Produção, UFRN. 2008.