Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM

DESENVOLVIMENTO DE NOVA META-HEURÍSTICA
E APLICAÇÃO EM PROBLEMAS DE OTIMIZAÇÃO

Aluno: Leandro Augusto de Araújo Silva
Matricula: 08.1.4032

Orientador: Marcone Jamilson Freitas Souza

Ouro Preto
25 de setembro de 2011

Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM

DESENVOLVIMENTO DE NOVA META-HEURÍSTICA E APLICAÇÃO EM PROBLEMAS DE OTIMIZAÇÃO

Proposta de monografia apresentada ao curso de Bacharelado em Ciência da Computação, Universidade Federal de Ouro Preto, como requisito parcial para a conclusão da disciplina Monografia I (BCC390).

Aluno: Leandro Augusto de Araújo Silva
Matricula: 08.1.4032

Orientador: Marcone Jamilson Freitas Souza

Ouro Preto
25 de setembro de 2011

Resumo

Este trabalho tem seu foco no desenvolvimento de uma meta-heurística inovadora – nomeada Nonggis (Non-greedy good Initial Solution), que exclui a necessidade de uma estratégia gulosa para criação de uma boa solução inicial –, a qual possa ser utilizada em uma extensa quantidade de problemas de otimização. Essa meta-heurística trabalha com uma matriz de probabilidades P , onde P_{ij} é a probabilidade da posição correta/ótima do elemento i ser j . Dentre esses, serão abordados o problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção (PSUMAA) e o conhecido Problema do Caixeiro Viajante (TSP). Para resolução dos problemas, propõe-se um algoritmo paralelo baseado em Nonggis, em aliança à Descida em Vizinhança Variável para a geração da solução inicial, Busca Tabu para refinamento desta solução e, por fim, a Reconexão por Caminhos como estratégia de pós-otimização. Após a coleta, serão comparadas tanto a qualidade de soluções iniciais com a de outras heurísticas, quanto de soluções finais de algoritmos da Literatura.

Palavras-chave: Nonggis. Caixeiro Viajante. Sequenciamento em uma Máquina. GRASP. Descida em vizinhança variável. Busca Tabu. Reconexão por Caminhos. Processamento Paralelo. Threads.

Sumário

1	Introdução	1
1.1	PSUMAA	1
1.2	PCV	1
2	Justificativa	2
3	Objetivos	3
3.1	Objetivo geral	3
3.2	Objetivos específicos	3
4	Metodologia	4
4.1	Nonggis	4
4.2	Busca Tabu	6
4.3	Reconexão por Caminhos	7
4.4	Algoritmo Proposto	9
5	Cronograma de atividades	10

Lista de Figuras

1	Funcionamento da Matriz Tabu	6
2	Matriz Tabu - Realocação Progressiva	7
3	Matriz Tabu - Realocação Regressiva	7

Lista de Tabelas

1	Cronograma de Atividades.	10
---	-----------------------------------	----

1 Introdução

Muitos problemas práticos são modelados da seguinte forma: Dado um conjunto S de variáveis discretas s (chamadas soluções) e uma função objetivo $f : S \leftarrow R$, que associa cada solução $s \in S$ a um valor real $f(s)$, encontre a solução $s^* \in S$, dita ótima, para a qual $f(s)$ tem o valor mais favorável (valor mínimo, no caso de o programa ter como objetivo a minimização de f , ou valor máximo, no caso de o problema ter como objetivo a maximização de f [7].

Grande parte desses problemas são combinatórios, sendo classificados na literatura como NP-Completo, e assim, ainda não existem algoritmos que os resolvam em tempo polinomial. Nos problemas da classe NP-Completo, não é possível garantir que a rota de custo mínimo seja encontrada em tempo polinomial. Assim, no pior caso, todas as possíveis soluções devem ser analisadas.

1.1 PSUMAA

De acordo com [1], o estudo do Problema de Sequenciamento de tarefas em uma Única Máquina envolvendo penalizações pela Antecipação e Atraso da produção é mais recente do que estudos voltados para problemas onde o objetivo envolve uma função não-decrescente do instante de conclusão do processamento da tarefa, tais como: tempo médio de fluxo, soma ponderada de atrasos e *makespan*. Nestes problemas, o atraso na conclusão das tarefas gera os custos mais elevados. Com o advento da adoção da filosofia just-in-time por muitas empresas, a penalização pela antecipação da produção tornou-se parte deste problema. Esta penalização se justifica pelo fato de que, concluir uma tarefa antecipadamente, pode resultar em custos financeiros extras pela necessidade antecipada de capital e/ou espaço para armazenamento e/ou de outros recursos para manter e gerenciar o estoque.

1.2 PCV

O Problema do Caixeiro Viajante é um dos mais estudados tanto por cientistas quanto por matemáticos e pesquisadores de outras áreas. Esse problema consiste em, dados n vértices e m arestas – cada qual com um peso w e ligando dois vértices quaisquer –, encontrar o ciclo de menor peso que “passe” por todos os vértices. Há bastantes aplicabilidades (industriais ou não), haja visto que alguns problemas são transformáveis ao TSP; o mais famoso desses, o problema de encontrar a menor rota entre cidades.

2 Justificativa

É possível dar uma certa “inteligência” a um método de enumeração, utilizando por exemplo as técnicas *branch-and-bound* ou *branch-and-cut*, de forma a reduzir o número de soluções a analisar no espaço de soluções. Com isto, pode ser possível resolver problemas de dimensões mais elevadas. Entretanto, dada a natureza combinatória do problema, pode ser que, no pior caso, todas as soluções tenham que ser analisadas. Este fato impede o uso exclusivo destes métodos, dito exatos, dado o tempo proibitivo de se encontrar a solução ótima. Portanto, em problemas desta natureza, o uso de métodos exatos se torna bastante restrito. Por outro lado, na prática, em geral, é suficiente encontrar uma “boa” solução para o problema, ao invés do ótimo global, o qual, para esta classe de problemas, somente pode ser encontrado após um considerável esforço computacional.

Esse é o motivo pelo qual os pesquisadores têm concentrado esforços na utilização de heurísticas para solucionar problemas deste nível de complexidade. Definimos **heurística** como sendo uma técnica inspirada em processos intuitivos que procura uma boa solução a um custo computacional aceitável, sem, no entanto, estar capacitada a garantir sua otimalidade, bem como garantir quão próximo está da solução ótima. O desafio é produzir, em tempo reduzido, soluções tão próximas quanto possível da solução ótima. Muitos esforços têm sido feitos nesta direção e heurísticas muito eficientes foram desenvolvidas para diversos problemas. Entretanto, a maioria das heurísticas desenvolvidas é muito específica para um problema particular, não sendo eficientes (ou mesmo aplicáveis) na resolução de uma classe mais ampla de problemas.

E é aí que entra a **Nonggis**: somente a partir da década de 1980 intensificaram-se os estudos no sentido de se desenvolver procedimentos heurísticos com uma certa estrutura teórica e com caráter mais geral, sem prejudicar a principal característica destes, que é a flexibilidade. Essa meta tornou-se mais realista a partir da reunião de conceitos das áreas de Otimização e Inteligência Artificial, viabilizando a construção das chamadas melhores estratégias ou dos métodos “inteligentemente flexíveis”, comumente conhecidos como **metaheurísticas**. Esses métodos, situados em domínios teóricos ainda pouco explorados pela literatura, possuem como característica básica estruturas com uma menor rigidez que as encontradas nos métodos clássicos de otimização sem, contudo, emergir em uma flexibilidade caótica.

Dentre os procedimentos enquadrados como metaheurísticas que surgiram ao longo das últimas décadas, destacam-se: Algoritmos Genéticos (AGs), Redes Neurais, Simulated Annealing (SA), Busca Tabu (BT), GRASP, VNS, Colônia de Formigas, etc.

As duas primeiras metaheurísticas fundamentam-se em analogias com processos naturais, sendo que os AGs são procedimentos inspirados em princípios da evolução natural. O SA explora uma possível analogia com a termodinâmica, enquanto a BT faz uso de uma memória flexível para tornar o processo de busca mais eficaz.

3 Objetivos

3.1 Objetivo geral

- Este trabalho tem como objetivo principal a disseminação e popularização dessa nova meta-heurística, mostrando suas altas aplicabilidade, adaptabilidade e eficácia para uma extensa classe de problemas de Otimização. Além disso, permitir que pessoas, mesmo leigas e/ou totalmente aquém de determinado problema, sejam capazes de apresentar uma boa solução para o mesmo.
- Incluir a meta-heurística em conhecidos frameworks.

3.2 Objetivos específicos

- Mostrar o funcionamento da *Nonggis* em problemas diferentes, mostrando quando criar a matriz de probabilidades de determinada maneira.
- Realizar testes da nova meta-heurística através do PSUMAA.
- Realizar testes da nova meta-heurística através do PCV.
- Comparar os resultados da *Nonggis* com outras meta-heurísticas (GRASP, por exemplo) e soluções gulosas conhecidas.

4 Metodologia

4.1 Nonggis

Função auxiliar *CriaMatrizDeProbabilidades*:

Algoritmo 1: CriaMatrizDeProbabilidades

Entrada: *solAtual*, *n*, *denominador*

Saída: *matProb*[][]

```
1 início
2   para  $i = 1 .. n$  faça
3     pos0 = i
4     matProb[sol[i]][pos0] = 1
5     para  $j = 1 .. \frac{n}{2}$  faça
6       ant  $\leftarrow (posIni + (i - 1) + n) \% n$ 
7       se ant == 0 então
8         | ant  $\leftarrow n$ 
9       fim
10      att  $\leftarrow (posIni + i + n) \% n$ 
11      se att == 0 então
12        | ant  $\leftarrow n$ 
13      fim
14      matProb[sol[i]][att]  $\leftarrow \frac{matProb[sol[i]][ant]}{denominador}$ 
15      att  $\leftarrow (posIni - i + n) \% n$ 
16      se att == 0 então
17        | att  $\leftarrow n$ 
18      fim
19      matProb[sol[i]][att]  $\leftarrow \frac{matProb[sol[i]][ant]}{denominador}$ 
20    fim
21  fim
22  return matProb
23 fim
```

Nonggis (para um problema de minimização qualquer):

Parâmetros:

$\alpha \Rightarrow$ Valor mínimo do denominador;

$\beta \Rightarrow$ Número máximo de soluções de mesmo valor objetivo consecutivas (critério de convergência);

$\gamma \Rightarrow$ Número máximo de iterações com determinado denominador;

$\kappa \Rightarrow$ Valor do incremento do denominador;

É importante ressaltar que quando o denominador é igual a 1, a solução é construída de forma totalmente aleatória e quanto maior seu valor, maior a probabilidade de que sua posição a ser selecionada seja a corrente – para exemplificar: com $n = 3$ e *denominador* = 2 e *pos0* = 2, a probabilidade de que a posição selecionada será a #2 é de 50%, enquanto a de que seja quaisquer das outras será de 25%.

Algoritmo 2: Nonggis

Entrada: $solAtual$
Saída: $solOtimizada$

```
1 início
2    $min \leftarrow calculaFO(solAtual)$ 
3    $nIterSemMelhora \leftarrow 0$ 
4    $nIterConv \leftarrow 0$ 
5    $denominador \leftarrow \alpha$ 
6    $ultRes \leftarrow \infty$ 
7    $matProb \leftarrow CriaMatrizDeProbabilidades(solAtual, n, denominador)$ 
8   repita
9      $LCR \leftarrow CriaListaDeCandidatosRestritos()$  // Nesse momento, são adicionados
10    todos os elementos a tal lista
11     $solAuxiliar \leftarrow \emptyset$ 
12    enquanto  $LCR \neq \emptyset$  faça
13       $elemenSort \leftarrow$  escolha aleatoriamente um elemento de  $LCR$ 
14       $LCR = LCR - \{elemenSort\}$ 
15       $posSort \leftarrow$  sorteie, via roleta não determinística, uma posição através do vetor
16       $matProb[elemenSort]$ 
17       $solAuxiliar[posSort] = elemenSort$ 
18      elimine a coluna  $posSort$  de  $matProb$ 
19    fim
20     $result \leftarrow calculaFO(solAuxiliar)$ 
21    se  $result < min$  então
22       $min \leftarrow result$ 
23       $solOtimizada \leftarrow solAuxiliar$ 
24       $nIterSemMelhora \leftarrow 0$ 
25       $denominador \leftarrow \alpha$ 
26       $matProb \leftarrow CriaMatrizDeProbabilidades(solOtimizada, n, denominador)$ 
27    fim
28    senão
29       $nIterSemMelhora \leftarrow nIterSemMelhora + 1$ 
30    fim
31    se  $nIterSemMelhora == \gamma$  então
32       $denominador \leftarrow denominador + \kappa$ 
33       $matProb \leftarrow CriaMatrizDeProbabilidades(solOtimizada, n, denominador)$ 
34    fim
35    se  $result \neq ultRes$  então
36       $ultRes \leftarrow result$ 
37       $nIterConv \leftarrow 0$ 
38    fim
39    senão
40       $nIterConv \leftarrow nIterConv + 1$ 
41    fim
42 até  $nIterConv > \beta$ 
43 return  $solOtimizada$ 
44 fim
```

4.2 Busca Tabu

O procedimento Busca Tabu ([3]), conhecido na literatura como *Tabu Search* – TS, inicia sua execução a partir da solução retornada pelo procedimento Nonggis. Esse método funciona da seguinte maneira: a cada iteração o espaço de soluções é explorado alternando entre as vizinhanças N^T e N^R , nesta ordem. Assim, na primeira iteração são aplicados movimentos de troca, na segunda movimentos de realocação e assim por diante. Após todos os vizinhos serem analisados, é aceito o melhor vizinho que não seja tabu, ou se tabu, que atenda a condição de aspiração. Para atender a esta condição, o vizinho considerado tabu deve ser melhor avaliado que a melhor solução existente até então segundo a função de avaliação, isto é, adota-se a condição de aspiração por objetivo. O vizinho escolhido é, então, aceito como a solução corrente. Esta escolha não implica necessariamente em uma solução melhor que a solução corrente, uma vez que o melhor vizinho não tabu da solução corrente pode ser uma solução pior. Esta condição de piora permite que o método consiga escapar de um ótimo local e explorar soluções em áreas ainda não analisadas.

Um atributo do melhor vizinho da solução corrente segundo a vizinhança N^T (ou N^R) é armazenado em uma Matriz Tabu T , de dimensão $n \times n$, sendo n o número de elementos. O método define de forma variável o tempo p em que uma solução permanecerá tabu, estando neste prazo proibido o retorno a esse vizinho. Esse prazo variável de proibição p é selecionado aleatoriamente no intervalo $[0, 9 \times \text{PrazoTABU} \leq p \leq 1, 1 \times \text{PrazoTABU}]$, sendo PrazoTABU um parâmetro do método. Assim, escolhido um vizinho, o valor de p é escolhido aleatoriamente no intervalo, sendo armazenado na Matriz Tabu o valor p somado ao número da iteração corrente do método.

Para exemplificar o funcionamento da Matriz T , considere a sequência $v = \{2, 3, 4, 1, 5\}$ com $n = 5$. Logo, T é representada pela matriz 5×5 (Figura 1(a)). Considere também um vizinho $v' = \{2, 5, 4, 1, 3\}$, resultado da troca entre E_2 e E_5 , ou seja, troca realizada entre os elementos 3 e 5. Este vizinho é caracterizado pelo par de elementos E_2 e E_3 de v ; assim, na posição (3,4) da matriz é armazenado um valor P que representa o tempo tabu p da solução somado ao número da iteração corrente. A Figura 1(b) mostra a matriz após a troca. [4]

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

(a) Matriz Tabu Inicial

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	P	0
4	0	0	0	0	0
5	0	0	0	0	0

(b) Matriz Tabu após a troca

Figura 1: Funcionamento da Matriz Tabu

Na realocação de elementos, analisando primeiro a realocação progressiva, considere novamente $v = \{2, 3, 4, 1, 5\}$ e o vizinho $v'' = \{3, 4, 1, 2, 5\}$. Este vizinho é o resultado de uma realocação do elemento E_1 (#2) para a posição $j = 4$. O atributo que caracteriza

esta solução é representado pelo par E_1 e E_2 de v ; desta maneira, a posição (2, 3) será preenchida com P , tal como anteriormente. A matriz resultante é mostrada na Figura 2.

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	P	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

Figura 2: Matriz Tabu - Realocação Progressiva

Analisando agora a realocação regressiva, considere o primeiro caso, no qual dada a solução corrente $v = \{2, 3, 4, 1, 5\}$ e o vizinho $v''' = \{4, 2, 3, 1, 5\}$ resultado da realocação do elemento E_3 (elemento 4) para a posição $j = 1$ com $E_i < E_n$. Desta maneira, a solução é caracterizada por E_3 e E_4 de v ; assim, o valor P é armazenado na posição (4, 1) (Figura 3(a)). Para o segundo caso, considere $v = \{2, 3, 4, 1, 5\}$ e o vizinho $v^{iv} = \{5, 2, 3, 4, 1\}$ resultado da realocação do elementos E_5 (elemento 5) para a posição $j = 1$ com $E_i = E_n$. O atributo que representará a solução será formado por E_4 e E_5 de v . Assim, será gravado na posição (1, 5) o valor de P (Figura 3(b)).

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	P	0	0	0	0
5	0	0	0	0	0

	1	2	3	4	5
1	0	0	0	0	P
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

(a) Realocação Regressiva 1

(b) Realocação Regressiva 2

Figura 3: Matriz Tabu - Realocação Regressiva

4.3 Reconexão por Caminhos

A técnica Reconexão por Caminhos ou Religação de Caminhos, conhecida na literatura inglesa como Path Relinking, foi proposta por [2] como uma estratégia de intensificação para explorar trajetórias que conectavam soluções elite obtida pelo método Busca Tabu ou Scatter Search [3]. Esta busca por soluções de melhor qualidade consiste em gerar e explorar caminhos no espaço de soluções partindo de uma ou mais soluções elite e levando a outras soluções elite. Para tal finalidade, são selecionados movimentos que introduzem atributos das soluções guia na solução corrente. Desse modo, a Reconexão por Caminhos pode ser vista como uma estratégia que objetiva incorporar atributos de soluções de boa qualidade, favorecendo a seleção de movimentos que as contenham.

A Reconexão por Caminhos pode ser aplicada segundo duas estratégias básicas [6]:

- Reconexão por Caminhos aplicada como uma estratégia de pós-otimização entre todos os pares de soluções elite;
- Reconexão por Caminhos aplicada como uma estratégia de intensificação a cada ótimo local obtido após a fase de busca local.

A Reconexão por Caminhos como pós-otimização é aplicada a pares (s_1, s_2) de soluções pertencentes a um *Grupo Elite* – limitado por *TamConjElite* – criado durante a exploração do espaço de soluções. Este conjunto está, inicialmente, vazio. Cada solução obtida ao final de um procedimento anterior – no caso, Busca Tabu – é considerada como uma candidata a ser inserida no conjunto elite, desde que ela seja melhor que a solução de pior qualidade desse conjunto e apresente um percentual mínimo de diferença em relação a cada solução do conjunto elite (*PercDif*). Essa estratégia é adotada para evitar que o conjunto elite contenha soluções muito parecidas. Se o conjunto estiver vazio, a solução é simplesmente inserida no conjunto. Se o conjunto elite já possui *TamConjElite* soluções e a solução corrente é candidata a ser inserida neste conjunto, então esta substitui a solução de pior qualidade.

O algoritmo inicia computando a diferença simétrica $\text{DELTA}(s_1, s_2)$ entre s_1 e s_2 , resultando no conjunto de movimentos que deve ser aplicado a uma delas, dita solução inicial, para alcançar a outra, dita solução guia. A partir da solução inicial, o melhor movimento ainda não executado de $\text{DELTA}(s_1, s_2)$ é aplicado à solução corrente até que a solução guia seja atingida. A melhor solução encontrada ao longo desta trajetória é considerada como candidata à inserção no conjunto elite e a melhor solução já encontrada é atualizada.

Diversas alternativas têm sido consideradas e combinadas em implementações recentes [6]:

- Explorar duas trajetórias potencialmente diferentes, usando s_1 como solução inicial e depois s_2 no mesmo papel;
- Explorar apenas uma trajetória, usando s_1 ou s_2 como solução inicial e não percorrer a trajetória completa de s_1 até s_2 , mas sim apenas parte dela (Reconexão por Caminhos Truncada).

Para a escolha da alternativa mais apropriada, deve-se ter um compromisso entre tempo de processamento e qualidade da solução. Em [5] foi observado que a exploração de duas trajetórias potencialmente diferentes para cada par (s_1, s_2) de soluções consome, aproximadamente, o dobro do tempo de processamento necessário para explorar apenas uma delas, com um ganho marginal muito pequeno em termos de qualidade de solução. Também foi observado pelos autores que, se apenas uma trajetória deve ser investigada, melhores soluções tendem a ser obtidas quando a Reconexão por Caminhos usa como solução inicial a melhor solução dentre s_1 e s_2 (Esta é a chamada Reconexão por Caminhos Regressiva - Backward Path Relinking). Como a vizinhança da solução inicial é explorada com muito mais profundidade do que aquela da solução guia, usar como solução inicial a melhor dentre s_1 e s_2 oferece mais chances ao algoritmo de investigar mais detalhadamente a vizinhança da solução mais promissora. Pela mesma razão, as melhores soluções são normalmente encontradas próximas da solução inicial, permitindo que o procedimento de Reconexão por Caminhos seja interrompido após algumas iterações, antes de a solução guia ser alcançada.

4.4 Algoritmo Proposto

Algoritmo 3: AlgoritmoProposto

Entrada: n

Saída: $solOtimizada$

1 início

2 $solOtimizada \leftarrow constroiSolucaoAleatoria(n)$

3 $solOtimizada \leftarrow Nonggis(solOtimizada)$

4 $solOtimizada \leftarrow BuscaTabu(solOtimizada)$

5 $solOtimizada \leftarrow ReconexaoPorCaminhos(solOtimizada)$

6 return $solOtimizada$

7 fim

5 Cronograma de atividades

A Tabela 1 contém o cronograma de atividades proposto para a elaboração da monografia.

Atividades	Ago	Set	Out	Nov	Dez
Revisão de literatura	X	X			
Estudo técnicas heurísticas	X	X			
Implementação formulações		X	X	X	
Testes		X	X	X	X
Paralelização do Nonggis			X	X	
Redigir a Monografia			X	X	X
Apresentação do Trabalho					X

Tabela 1: Cronograma de Atividades.

Referências

- [1] M. F. França Filho. *GRASP e Busca Tabu aplicados a problemas de programação de tarefas em máquinas paralelas*. PhD thesis, UNICAMP, Brazil, Departamento de Engenharia de Sistemas, 2007.
- [2] F. Glover. Tabu search and adaptative memory programming – advances, applications and challenges. In *Interfaces in Computer Sciences and Operations Research*, pages 1–75, 1996.
- [3] F. Glover and M. Laguna. Tabu search. *Boston: Kluwer Academic Publishers*, 1997.
- [4] F. A. Almeida Gonçalves, M. J. Freitas Souza, and S. R. de Souza. Sequenciamento em uma máquina: Otimização heurística via multiprocessamento paralelo. 2010.
- [5] C. C. Ribeiro, E. Uchoa, and R. F. Werneck. A hybrid grasp with perturbations for the steiner problem in graphs. *INFORMS Journal on Computing*, pages 14:228–246, 2002.
- [6] I. C. M. Rosseti. *Estratégias sequenciais e paralelas de GRASP com reconexão por caminhos para o problema de síntese de redes a 2-caminhos*. PhD thesis, PUC-RJ - Pontifícia Universidade Católica do Rio de Janeiro, Brazil, 2003.
- [7] M. J. Freitas Souza. *Inteligência Computacional para Otimização*.