Universidade Federal de Ouro Preto - UFOP

Instituto de Ciências Exatas e Biológicas - ICEB

Departamento de Computação - DECOM

DESENVOLVIMENTO DE UMA NOVA METAHEURÍSTICA E APLICAÇÃO EM PROBLEMAS DE OTIMIZAÇÃO

Aluno: Leandro Augusto de Araújo Silva
Matricula: 08.1.4032

Orientador: Marcone Jamilson Freitas Souza

Ouro Preto
8 de dezembro de 2011

Universidade Federal de Ouro Preto - UFOP

Instituto de Ciências Exatas e Biológicas - ICEB

Departamento de Computação - DECOM

DESENVOLVIMENTO DE UMA NOVA METAHEURÍSTICA 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
8 de dezembro de 2011

Resumo

Este trabalho tem seu foco no desenvolvimento de uma metaheurística inovadora, nomeada *Nonggis*, que pode tanto excluir a necessidade de uma estratégia para criação de uma boa solução inicial quanto melhorar uma (boa) solução conhecida. Essa metaheurística trabalha com uma matriz de probabilidades P, em que cada componente P_{ij} é a probabilidade da posição correta/ótima do elemento i da sequência estar na posição j. *Nonggis* pode ser aplicada a uma extensa quantidade de problemas de otimização. Dentre esses, serão mostradas aplicações no problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção (PSUMAA) e o conhecido Problema do Caixeiro Viajante (PCV). Para resolução desses problemas, propõe-se um algoritmo paralelo baseado em Nonggis, em aliança ao Método de 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

Lista de Figuras

Lista de Tabelas

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 $s^\star \in S$, 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 [37].

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

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 dessa classe de problemas, 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. Essas técnicas procuram uma boa solução a um custo computacional aceitável, sem, no entanto, estar capacitadas 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.

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", comumemente 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.

Uma deficiência comum à maioria das metaheurísticas, no entanto, é a necessidade de uma boa solução inicial, pelo menos quando o tempo de processamento é um fator limitante para a apresentação da solução. No presente trabalho, pretendemos apresen-

tar o desenvolvimento de uma nova metaheurística, nomeada *Nonggis*, que pode tanto excluir a necessidade de uma estratégia para criação de uma boa solução inicial quanto melhorar uma (boa) solução conhecida.

Para validá-la, ela será aplicada a dois problemas comuns na área de otimização: ao Problema do Caixeiro Viajante (PCV) e a uma classe de problemas de sequenciamento em uma máquina com penalidades por antecipação e atraso (PSUMAA).

O PCV 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. O PCV é um dos problemas mais estudados tanto por cientistas quanto por matemáticos e pesquisadores de outras áreas. Isso é devido à sua grande aplicabilidade em diversos setores (industriais ou não), e à sua dificuldade de resolução na otimalidade.

De acordo com [11], 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 torna 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.

2 Justificativa

O desenvolvimento de um procedimento heurístico para resolver eficientemente um problema de otimização é de fundamental importância em vários campos da ciência. A importância pode ser destacada no aspecto financeiro relacionado à maior economia que a técnica proporciona quando, por exemplo, ela é aplicada para reduzir os custos de transporte de uma empresa de atividades logísticas. Mas a contribuição não é apenas financeira. Por exemplo, em um problema de designação de escalas de controladores de voo, uma boa técnica pode produzir escalas que estabeleçam um equilíbrio da carga de trabalho entre os diversos controladores de voo, fazendo uma distribuição mais justa da carga de trabalho.

As metaheurísticas atualmente disponíveis, à exceção de *Simulated Annealing*, requerem que uma boa solução inicial esteja disponível quando o fator tempo é limitante da apresentação da solução. Porém, em muitas aplicações, principalmente as de caráter operacional, exigem que as decisões sejam tomadas rapidamente. Assim, é essencial produzir soluções de boa qualidade rapidamente.

3 Objetivos

3.1 Objetivo geral

Este trabalho tem como objetivo principal o desenvolvimento de uma metaheurística eficiente, que não requeira uma boa solução inicial, mostrando sua aplicabilidade, adaptabilidade e eficácia para uma extensa classe de problemas de Otimização.

3.2 Objetivos específicos

- Descrever o funcionamento detalhado da ***Nonggis***, de forma a permitir sua reprodutibilidade;
- Testar a nova metaheurística nos problemas PSUMAA e PCV;
- Comparar os resultados da ***Nonggis*** nesses problemas com os de outros métodos da literatura;
- Disseminar o uso da nova metaheurística;
- Incluir a metaheurística em conhecidos frameworks.

4 Revisão de Literatura

Nesta seção é feita uma revisão de literatura do PSUMAA. Uma revisão do PCV será feita na monografia final.

A produção de bens sob encomenda por meio da filosofia *just-in-time* tornou-se uma opção comumente adotada pelas empresas nos últimos anos. Em virtude disso, um planejamento criterioso da produção se faz necessário, visto que a antecipação ou atraso da produção poderão implicar em custos extras para as empresas. Dentre estes custos, podem ser citados: custos de armazenagem pela a antecipação da produção e custos por multas contratuais devido ao atraso da produção.

O problema de sequenciamento em uma máquina visando a minimização das penalidades por antecipação e atraso da produção (PSUMAA), trata um dos casos mais simples dos problemas de planejamento da produção no contexto da filosofia *just-in-time*. Entretanto, este problema é difícil de ser resolvido na sua otimalidade com tempos computacionais aceitáveis, devido ao fato de pertencer à classe NP-difícil [9, 39].

O PSUMAA com datas de entrega comum tem sido estudado em vários trabalhos. Em [17] os autores produziram um estudo unificado sobre datas comuns de entrega relacionadas a problemas de sequenciamento em uma máquina e em máquinas paralelas. De acordo com o autor, a filosofia *just-in-time* foi fator determinante no estudo das datas de entrega em problemas de sequenciamento. Segundo esse autor, o trabalho [21] foi o ponto de partida do estudo. Também foram revisados alguns modelos para atribuição da data de entrega e apresentados várias formulações para a função objetivo dos problemas de sequenciamento em uma máquina e em máquinas paralelas, além de algumas propriedades para datas comuns de entrega.

[22] propôs um algoritmo utilizando a metaheurística Busca Tabu para resolver o PSUMAA com data comum de entrega, penalidades pela antecipação e atraso da produção e sem permitir tempo ocioso entre as tarefas. As estruturas de vizinhanças utilizadas foram: troca entre tarefas adjacentes, troca entre duas tarefas quaisquer e inserção de uma tarefa à frente de outra tarefa qualquer. Segundo o autor, experimentos demonstraram que a vizinhança baseada na inserção de tarefas produziram resultados melhores que a vizinhança com troca entre duas tarefas quaisquer e que a troca entre duas tarefas foi melhor que a troca envolvendo tarefas adjacentes. Contudo, movimentos híbridos baseados nas vizinhanças de inserção e troca entre duas tarefas quaisquer produziram melhores soluções. O método proposto utilizou duas técnicas para explorar o espaço de soluções. Na primeira, chamada *job space solution*, as soluções visitadas não satisfazem a propriedade *V-shaped*; enquanto na segunda, denominada *early/tardy solution space*, as soluções visitadas satisfazem a essa propriedade.

[24] e [25] trataram o PSUMAA com datas comuns de entrega, sem permitir tempo ocioso de máquina. O primeiro autor decompõe o problema em dois subproblemas com uma estrutura mais simples, de forma que o limite inferior do problema é a soma dos limites inferiores desses dois subproblemas. O limite inferior de cada subproblema é determinado por relaxação lagrangiana. Um algoritmo *branch-and-bound* é apresentado e usado para resolver instâncias de até 50 tarefas, dobrando a dimensão de problemas que podiam ser resolvidos na otimalidade com algoritmos exatos até aquela data. O autor propôs, também, procedimentos heurísticos baseados em busca local para resolver problemas de dimensões mais elevadas. No segundo trabalho é apresentado um algoritmo *branch-and-bound* que faz uso de procedimentos para determinar limites in-

feriores e superiores fortes. Regras de dominância são usadas para tentar eliminar nós não promissores na árvore de busca. É analisado o desempenho do algoritmo para resolver problemas de até 50 tarefas.

[4] trataram o PSUMAA com datas comuns de entrega e propuseram um gerador de problemas-teste, com o qual geraram um total de 280 instâncias. Foram apresentadas duas heurísticas para resolver estes problemas. Segundo os autores, o estudo também teve a intenção de utilizar os problemas-teste gerados para futuras comparações de desempenho com diferentes metodologias para resolução dos problemas.

[10] resolveram problemas do PSUMAA com datas comuns de entrega [4] por meio de três métodos: Algoritmo Evolutivo, *Simulated Annealing* – SA e uma versão aperfeiçoada da heurística *Threshold Accepting*, sendo esta última uma variante de *Simulated Annealing*. Segundo os autores, o método *Threshold Accepting* foi mais eficiente na obtenção de melhores soluções. O mesmo problema foi tratado por [20], os quais apresentaram métodos baseados nas metaheurísticas Busca Tabu e Algoritmos Genéticos, bem como hibridizações destas. Nesse trabalho, o autor aponta uma melhor qualidade das soluções do Algoritmo Genético sobre a Busca Tabu, bem como uma similaridade dos resultados do Algoritmo Genético com o Algoritmo Híbrido.

[40] trata do PSUMAA com datas comuns de entrega das tarefas e com tempo de *setup* de cada tarefa incluído em seu tempo de processamento e independente da sequência de produção. O problema é resolvido pelo algoritmo *Recovering Beam Search* – RBS, que é uma versão aperfeiçoada do algoritmo *Beam Search* – BS. Este, por sua vez, consiste em um algoritmo *branch-and-bound* em que somente os w nós mais promissores de cada nível da árvore de busca são retidos para ramificação futura, enquanto os nós restantes são podados permanentemente. Para evitar decisões equivocadas com respeito à poda de nós que conduzam à solução ótima, o algoritmo RBS utiliza uma fase de recobrimento que busca por soluções parciais melhores que dominem aquelas anteriormente selecionadas. O método proposto foi comparado com os de [10, 20], alcançando melhores resultados.

[28] focaram no PSUMAA com datas comuns de entrega e tempo de preparação da máquina dependente da sequência de produção. Os autores apresentam um procedimento *branch-and-bound* que é capaz de resolver na otimalidade, em tempos aceitáveis, problemas-teste de até 25 tarefas, o que representava um avanço porque, até aquela data, os algoritmos exatos para essa classe de problemas eram capazes de resolver somente problemas-teste de até 8 tarefas.

O PSUM com penalidade por atraso e data de entrega distinta foi estudado por [3], os quais utilizaram a metaheurística Busca Tabu para resolvê-lo. Para gerar a solução inicial os autores utilizaram vários métodos, entre eles as heurísticas *earliest due date* – EDD e *weighted shortest processing time* – WSPT. A BT desenvolvida usou uma estrutura de vizinhança híbrida com movimentos de troca e realocação; porém, os autores utilizaram uma estratégia para criação de uma lista de candidatos, se valendo de algumas propriedades do problema, para não permitir movimentos que não levariam a boas soluções. Foi também utilizado um tempo tabu (*tabu tenure*) dinâmico para evitar a ciclagem.

[23] aplicaram Algoritmos Genéticos – AG ao PSUMAA com datas de entrega distintas. Para determinar a data ótima de início do processamento de cada tarefa da sequência produzida pelo AG, desenvolveram um algoritmo específico, de complexidade polinomial, que explora as características do problema.

[6] tratou o PSUMAA com datas distintas de entrega por meio de um algoritmo *branch-and-bound*. É analisado o desempenho desse algoritmo para resolver problemas com até 45 tarefas. Também foi desenvolvido um esquema para delimitar o cálculo de diferentes limites inferiores baseados no procedimento de eliminação de sobreposição de uma sequência *just-in-time*. Propriedades e teoremas sobre procedimento de eliminação de sobreposição são apresentados.

[18] propuseram dois algoritmos, um baseado em GRASP e outro na heurística *space-based search*, para a resolução do problema de sequenciamento em uma máquina considerando a existência de data de entrega para cada tarefa e tempo de preparação de máquina dependente da sequência de produção, tendo como objetivo a minimização do tempo total de atraso. O algoritmo GRASP proposto foi dividido em três fases: Construção, Refinamento e Reconexão por Caminhos. A fase de refinamento do método GRASP baseou-se nas heurísticas *Variable Neighborhood Search* – VNS e *Variable Neighborhood Descent* – VND. Segundo os autores, a contribuição está em uma nova função custo para a fase de construção, uma nova variação do método VND para a fase de refinamento e uma fase de Reconexão por Caminhos usando diferentes vizinhanças.

[19] tratou o PSUMAA com datas de entrega distintas, não sendo permitida a existência de tempo ocioso entre as tarefas. O autor propõe um método híbrido que combina heurísticas de busca local (regras de despacho, método da descida e *Simulated Annealing*) e um algoritmo evolucionário.

[39] apresentaram um algoritmo baseado na metaheurística Busca Tabu – TS para resolver o PSUMAA com janelas de entrega distintas. Para cada sequência de tarefas gerada pela Busca Tabu é acionado um procedimento de complexidade polinomial para determinar a data ótima de início do processamento de cada tarefa da sequência. Este último procedimento é uma adaptação daquele proposto em [23]. Nesta adaptação, consideram-se janelas de entrega no lugar de datas de entrega.

[15] desenvolveram um modelo de programação linear inteira mista para o PSUMAA com janelas de entrega e tempo de preparação dependente da sequência de produção (PSUMAA-JP). O modelo de desenvolvido foi utilizado para resolver na otimalidade problemas de até 12 tarefas. Esta modelagem serviu para comparar os resultados obtidos por um algoritmo heurístico baseado em GRASP, ILS e VND, também proposto pelos autores. Para cada sequência gerada pela heurística, é acionado um algoritmo para determinar a data ótima de conclusão do processamento de cada tarefa. Tal algoritmo foi adaptado de [39], e inclui no tempo de processamento de uma tarefa o tempo de preparação da máquina, já que quando ele é acionado, já se conhece a sequência de produção. O algoritmo desenvolvido pelos autores foi capaz de encontrar todas as soluções ótimas conhecidas.

[34] desenvolveu dois modelos de programação linear inteira mista para o PSUMAA-JP. Dentre as duas novas formulações propostas nesse trabalho, uma delas se trata de um aperfeiçoamento do modelo proposto por [15], e a outra, uma formulação indexada no tempo. Foi proposto também, um algoritmo heurístico de duas fases baseado em GRASP e VND. Segundo o autor, a formulação indexada no tempo possibilitou a obtenção de um maior número de soluções ótimas, bem como um menor esforço computacional gasto na resolução dos problemas. Já o algoritmo heurístico obteve tempos computacionais inferiores aos resultados da literatura e se mostrou uma metodologia bastante competitiva com as outras existentes.

[33] propôs uma Algoritmo Genético Adaptativo para resolução do PSUMAA-JP.

A população inicial é gerada a partir da metaheurística GRASP. Para cada indivíduo gerado gerado é utilizado algoritmo de tempo polinomial [15] para determinar a data ótima de início do processamento de cada tarefa na sequência dada. São utilizados cinco operadores de cruzamento e um mutação. O algoritmo também utiliza periodicamente a Reconexão por Caminhos com o objetivo de encontrar soluções intermediárias de boa qualidade.

[36] desenvolveram um algoritmo híbrido sequencial de três fases baseado em GRASP, VND, Busca Tabu e Reconexão por caminhos para o PSUMAA-JP. Na primeira fase foi utilizado um procedimento baseado em GRASP para gerar a solução inicial. As soluções geradas neste método são refinadas por um procedimento baseado em VND. Na segunda fase, a melhor solução construída pelo procedimento GRASP é então refinada por um procedimento baseado em Busca Tabu. Durante a execução da Busca Tabu, um conjunto de soluções denominado conjunto elite é formado. Na terceira e última fase, como estratégia de pós-otimização é aplicado o procedimento Reconexão por Caminhos aos pares de solução do conjunto elite.

Para uma revisão abrangente de vários estudos sobre problemas de sequenciamento com tempos de preparação, apontamos os trabalhos de [1, 2].

Com relação à utilização de algoritmos paralelos, de nosso conhecimento, nenhuma aplicação ainda foi desenvolvida para tratar o problema em questão. Contudo, alguns trabalhos que utilizam algoritmos paralelos aplicados a outros problemas são citados a seguir.

[31] apresentaram um *framework* que implementa a abstração *MapReduce* para a linguagem C++. A utilização deste *framework* torna o desenvolvimento de aplicações paralelas mais simples, uma vez que não é requerido ao desenvolvedor nenhum conhecimento do mecanismo de paralelização do problema em estudo. Para validar o *framework* proposto, foi desenvolvido um algoritmo paralelo para resolver o Problema do Caixeiro Viajante – PCV. Segundo os autores, os resultados comprovaram a eficiência desta ferramenta.

[30] desenvolveram um algoritmo paralelo para resolver o problema de planejamento operacional de lavra com alocação dinâmica de caminhões. O algoritmo desenvolvido combina a paralelização do *Iterated Local Search* – ILS com os procedimentos GRASP e *Variable Neighborhood Descent*. Foram utilizados oito movimentos para explorar o espaço de soluções. Os resultados produzidos foram comparados aos de um modelo de programação linear inteira mista resolvido pelo otimizador CPLEX e um procedimento heurístico sequencial baseado em ILS, GRASP e VND [8]. Segundo os autores, houve uma melhoria na qualidade das suas soluções finais quando comparadas às obtidas pelo algoritmo sequencial. Houve também melhora em relação às soluções encontradas pelo CPLEX.

[12] desenvolveram um algoritmo memético paralelo aplicado ao problema de sequenciamento em uma máquina com datas de entrega distintas e tempo de preparação da máquina dependente da produção. Foram revisados alguns modelos clássicos de algoritmos evolutivos paralelos e a estrutura geral dos algoritmos meméticos. O algoritmo proposto foi baseado em Algoritmos Evolutivos Paralelos Globais – AEPG. Nessa implementação, utilizou-se programas mestre-escravo. Foram realizados experimentos computacionais com 8 problemas-teste, sendo 4 com 71 tarefas e outras 4 com 100. Observou-se um ganho de desempenho com a utilização de mais processadores.

[38] resolveram o problema de roteirização com reabastecimento e entregas dire-

tas (*Inventory Routing Problem with Direct Deliveries* – IRPDD). Foram consideradas algumas simplificações do modelo, mas segundo os autores a aplicabilidade não foi descaracterizada. Utilizou-se sistemas computacionais paralelos de baixo custo, também conhecidos como *beowulf clusters* e a reposição do estoque gerenciada pelo fornecedor (*Vendor Managed Inventory* – VMI). Foi proposto um algoritmo genético paralelo baseado em ilhas com operadores de migração. O algoritmo foi implementado em Fortran e MPI e testado num *beowulf cluster* de 6 nós. Foram obtidas acelerações quase lineares no tempo de processamento.

[29] descreveram estratégias simples para paralelização de algoritmos baseados em GRASP. Nesse trabalho, os autores categorizaram a estratégia de paralelização de duas formas: *multiple-walk independent-thread* e *multiple-walk cooperative-thread*. Foi proposto um algoritmo híbrido baseado em GRASP e Reconexão por Caminhos. O algoritmo proposto foi testado com 3 problemas-teste da literatura utilizando-se as duas estratégias de paralelização citadas anteriormente. Segundo os autores, a estratégia *multiple-walk cooperative-thread* levou a melhores resultados neste algoritmo híbrido, tornando-o mais rápido e robusto.

[27] apresentou um algoritmo de busca local paralelo para computar uma árvore de *Steiner* em um plano bidimensional. Como vantagem deste algoritmo, não se tem o *overhead* gerado por parte da comunicação entre os processos. Segundo os autores, a busca local paralela permitiu resolver problemas maiores e melhorar a qualidade das soluções finais.

[5] propuseram a utilização de metaheurísticas e computação paralela para a resolução de um problema real de roteamento de veículos com frota heterogênea, janelas de tempo e entregas fracionadas. Nesse problema, a demanda dos clientes pode ser maior que a capacidade dos veículos. A solução do problema consiste na determinação de um conjunto de rotas econômicas que devem atender a necessidade de cada cliente respeitando várias restrições descritas no trabalho. O algoritmo proposto utilizou uma adaptação da heurística construtiva proposta por [7] como solução inicial e para refinamento a heurística Meta-RaPS [26]. Posteriormente, implementa-se um algoritmo genético paralelo que é resolvido com o auxílio de um *cluster* de computadores, com o objetivo de explorar novos espaços de soluções. A estratégia adotada para o algoritmo genético paralelo é o de ilhas com a utilização de operadores de migração. Os resultados obtidos demonstram que a heurística construtiva básica apresenta resultados satisfatórios para o problema, mas pode ser melhorada substancialmente com o uso de técnicas mais sofisticadas. A aplicação do algoritmo genético paralelo de múltiplas populações proporcionou um redução no custo total da operação na ordem de 10%, em relação à heurística construtiva e 13%, quando comparada às soluções utilizadas originalmente pela empresa.

5 Metodologia

As seções seguintes detalham a metaheurística proposta.

5.1 Nonggis

A metaheurística Nonggis, ilustrada pelo Algoritmo 2, tem como função-chave a criação de uma matriz de probabilidades, descrita pelo Algoritmo 1.

Algoritmo 1: CriaMatrizDeProbabilidades

Entrada: sol, n, $denominador$
Saída: $matProb[][]$

```
1  início
2      para i = 1 .. n faça
3          total ← 0
4          para j = 1 .. n faça
5              matProb[sol[i]][j] ← (denominador^distancia(i,j))^-1
                   // Essa distância varia de acordo com o problema.  Por exemplo, no
                   PSUMMA a distância entre o primeiro e o último elemento de uma
                   solução é n, enquanto no PCV, essa distância é 1.
6              total ← total + matProb[sol[i]][j]
7          fim
8          //Normalização dos resultados
9          para j = 1 .. n faça
10             matProb[sol[i]][j] ← matProb[sol[i]][j]×100 / total
11         fim
12     fim
13     return matProb
14 fim
```

Para um problema de minimização qualquer, considere os seguintes parâmetros para a Nonggis:

$\alpha \Rightarrow$ Valor mínimo do denominador;
$\beta \Rightarrow$ Número máximo de soluções de mesmo valor objetivo consecutivas (critério de convergência);
$\omega \Rightarrow$ Número máximo de iterações com determinado denominador;
$\kappa \Rightarrow$ Valor do incremento do denominador;
$\theta \Rightarrow$ Percentual máximo que uma solução pode ser pior que a melhor corrente para que o denominador não seja incrementado. Este parâmetro evita que muitas soluções potencialmente ruins sejam exploradas.

Algoritmo 2: Nonggis

```
   Entrada: solAtual
   Saída: solOtimizada
 1 início
 2     exectype ← 1
 3     φ ← ∞
 4     enquanto exectype ≤ 2 faça
 5         min ← calculaFO(solAtual)
 6         nIterSemMelhora ← 0
 7         nIterConv ← 0
 8         denominador ← α
 9         ultRes ← ∞
10         matProb ← CriaMatrizDeProbabilidades(solAtual, n, denominador)
11         se exectype == 2 então
12             InvertaProbabilidades(matProb)
13         fim
14         repita
15             LCR ← CriaListaDeCandidatosRestritos()    // Nesse momento, são adicionados todos os
               elementos a tal lista
16             solAuxiliar ← ∅
17             enquanto LCR ≠ ∅ faça
18                 elemenSort ← escolha aleatoriamente um elemento de LCR
19                 LCR = LCR − {elemenSort}
20                 posSort ← sorteie, via roleta não determinística, uma posição através do vetor
                   matProb[elemenSort]
21                 solAuxiliar[posSort] = elemenSort
22                 elimine a coluna posSort de matProb
23             fim
24             result ← calculaFO(solAuxiliar)
25             se result < min então
26                 exectype ← 1
27                 min ← result
28                 solOtimizada ← solAuxiliar
29                 nIterSemMelhora ← 0
30                 denominador ← α
31                 matProb ← CriaMatrizDeProbabilidades(solOtimizada, n, denominador)
32             fim
33             senão
34                 nIterSemMelhora ← nIterSemMelhora + 1
35             fim
36             se (nIterSemMelhora == ω) OR (result > θ×min/100) então
37                 denominador ← denominador + κ
38                 matProb ← CriaMatrizDeProbabilidades(solOtimizada, n, denominador)
39                 se exectype == 2 então
40                     InvertaProbabilidades(matProb)
41                 fim
42             fim
43             se result ≠ ultRes então
44                 ultRes ← result
45                 nIterConv ← 0
46             fim
47             senão
48                 nIterConv ← nIterConv + 1
49             fim
50         até (nIterConv > β) OR (denominador > φ)
51         se exectype == 1 então
52             φ ← denominador
                       // Isso implica que φ é igual ao denominador no qual a solução convergiu
53         fim
54         exectype ← exectype + 1
55     fim
56     return solOtimizada
57 fim
```

5.1.1 Funcionamento

Na prática, a metaheuristíca funciona da seguinte forma:

1. Gere uma solução aleatória;
2. Atribua 1 ao denominador;
3. Crie a matriz de probabilidades da seguinte forma, através de um único parâmetro (denominador): há uma linha / um vetor para cada elemento; o peso da posição corrente do elemento na solução será 1; seja d a distância entre a posição avaliada e a corrente, as demais posições possuem peso $1/denominador^d$;
4. Sorteie um elemento, rode a roleta e o insira na posição que a mesma parar; remova a coluna da referida posição da matriz e continue o procedimento até que a solução seja formada. Se essa solução for melhor que a melhor encontrada até então, retorne a 2;
   Se ela não for de melhora, verifique se o número máximo de soluções sem melhora para um mesmo denominador foi atingido ou se a função objetivo dela é maior $pMax$ vezes a função objetivo da melhor solução, em caso positivo de quaisquer das duas, incremente o denominador e retorne a 3;
5. Se as últimas m soluções tiverem mesmo valor de função objetivo, inicie a fase de atenuação, da seguinte forma: $mDen =$ denominador no qual a solução convergiu;
6. Atribua 1 ao denominador;
7. Crie a matriz de probabilidades tal como em 3, mas, ao invés de passar o denominador como parâmetro da função, passe 1/denominador;
8. Sorteie um elemento, rode a roleta e o insira na posição que a mesma parar; remova a coluna da referida posição da matriz e continue o procedimento até que a solução seja formada. Se essa solução for melhor que a melhor encontrada até então, retorne a 2;
   Se ela não for de melhora, verifique se o número máximo de soluções sem melhora para um mesmo denominador foi atingido ou se a função objetivo dela é maior $pMax$ vezes a função objetivo da melhor solução, em caso positivo de quaisquer das duas, incremente o denominador e verifique se o mesmo se tornou maior que $mDen$, se sim, o resultado do procedimento é a melhor solução encontrada até então, senão retorne a 7;

Repare 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 a posição a ser selecionada seja a corrente – para exemplificar: com $n = 3$ e $denominador = 2$ e $pos0 = 2$, a probabilidaede que a posição selecionada seja a #2 é de 50%, enquanto a de que seja qualquer outra será de 25%.

É importante ressaltar o acréscimo de uma fase de atenuação, a qual é incluída após a fase de intensificação e por meio da inversão da matriz de probabilidades.

Atente também que a distãncia através da qual a matriz de probabilidades é construída varia de acordo com o problema. Por exemplo, no PSUMMA a distância entre o primeiro e o último elemento de uma solução é n, enquanto no PCV, essa distância é 1.

5.2 Busca Tabu

O procedimento Busca Tabu ([14]), 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 $[0,9 \times PrazoTABU \leq p \leq 1,1 \times PrazoTABU]$, 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. [16]

	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

(a) Realocação Regressiva 1

	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

(b) Realocação Regressiva 2

Figura 3: Matriz Tabu - Realocação Regressiva

5.3 Reconexão por Caminhos

A técnica Reconexão por Caminhos ou Religação de Caminhos, conhecida na literatura inglesa como *Path Relinking*, foi proposta por [13] como uma estratégia de intensificação para explorar trajetórias que conectavam soluções elite obtida pelo método Busca Tabu ou *Scatter Search* [14]. 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 [35]:

- 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-otmização 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 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 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 [35]:

- 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 [32] 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.

5.4 Algoritmo Proposto

O algoritmo proposto para resolução de um problema combinatório é apresentado pelo Algoritmo 3.

Algoritmo 3: AlgoritmoProposto

Entrada: n
Saída: $solOtimizada$
1 **início**
2 $\quad solOtimizada \leftarrow constroiSolucaoAleatoria(n)$
3 $\quad solOtimizada \leftarrow Nonggis(SolOtimizada)$
4 $\quad solOtimizada \leftarrow BuscaTabu(SolOtimizada)$
5 $\quad solOtimizada \leftarrow ReconexaoPorCaminhos(SolOtimizada)$
6 $\quad$ return $solOtimizada$
7 **fim**

Logo o algoritmo proposto é uma combinação da Nonggis com Busca Tabu e Reconexão por Caminhos.

Os procedimentos Busca Tabu e Reconexão por Caminhos necessitam de uma boa solução inicial tanto para ganho de tempo quanto de qualidade. Como se espera isso de uma solução advinda rapidamente da metaheurística criada, ao final há grande probabilidade que uma solução, no mínimo, próxima da ótima seja encontrada em um tempo inferior ao da maioria dos algoritmos conhecidos da literatura. O que é de extrema valia sobretudo quando essas soluções têm que ser encontradas quase imediatamente.

6 Situação atual da proposta de monografia

A Tabela 1 contém o cronograma de atividades que havia sido proposto para o desenvolvimento do projeto em curso.

Atividades	**Ago**	**Set**	**Out**	**Nov**	**Dez**
Revisão de literatura	X	X			
Estudo técnicas heurísticas	X	X			
Implementação de formulações		X	X	X	
Testes		X	X	X	X
Paralelização do Nonggis			X	X	
Redação do projeto de monografia			X	X	X
Apresentação do Trabalho					X

Tabela 1: Cronograma de Atividades para a disciplina BCC390.

6.1 Revisão de Literatura

Como os resultados parciais são somente do PSUMAA, sua revisão de literatura foi realizada dentro do tempo previsto e uma seção para tal foi adicionada. O que falta é revisar a literatura quanto ao PCV, tarefa a ser executada em BCC391 - Monografia II.

6.2 Estudo de Técnicas Heurísticas

As técnicas heurísticas vem sendo por mim realizado desde meu segundo período no curso, quando me matriculei na disciplina Inteligência Computacional para Otimização, lecionada pelo professor Frederico Gadelha, quando foram ensinadas várias heurísticas (entre elas, GRASP, Busca Tabu, *Path Relinking* e *Simulated Annealing*). Concomitantemente comecei a iniciação científica com o professor Marcone Freitas e tive que me aprofundar no tema. Posteriormente cursei a disciplina Computação Evolutiva, também lecionada pelo professor Frederico, onde foram ensinados algoritmos genéticos e evolutivos (os quais também se tratam de técnicas heurísticas). Ademais já estou envolvido no meu terceiro projeto de iniciação científica, sendo todos relacionados a metaheurísticas.

O que tive que estudar nesses dois meses para melhoria da metaheurística criada foram algoritmos probabilísticos, através tanto de fontes bibliográficas quanto da disciplina Projeto e Análise de Algoritmos.

6.3 Implementação de Formulações

Tal como planejado, a implementação da Nonggis se inciou em meados de Setembro e, ainda no corrente mês, já havia uma versão funcionando. Como já possuía uma implementação do PSUMAA, bastou integrá-la ao mesmo para que os primeiros resultados foram obtidos – ainda não muito bons, mas promissores.

No decorrer dos meses, algumas alterações foram realizadas para melhoria dos resultados e, entre as principais, estão os ajustes dos parâmetros da metaheurística outrora apresentados e a adição de um novo, que não permita que soluções potencialmente ruins sejam abundantemente exploradas. Com essas alterações, tanto os resultados quanto o tempo para obtê-los foram reduzidos.

Uma outra alteração feita foi a adição de uma fase de atenuação, que deve estar presente em metaheurísticas.

A implementação do PCV está em processo, já com uma versão da Nonggis funcionando. Todavia as demais metaheurísticas propostas (Busca Tabu e *Path Relinking*) ainda não foram nele implementadas, o que será feito na disciplina BCC391 - Monografia II.

6.4 Testes

Para o PSUMAA, a implementação está terminada e os testes foram realizados sobre o mesmo. As instâncias utilizadas foram as mesmas abordadas no programa de iniciação científica. Essas instâncias foram criadas por [15], tendo em vista a metodologia descrita em [39] e [28]. Os resultados obtidos são comparados com aqueles gerados por outros algoritmos da literatura.

6.4.1 Ambiente de Desenvolvimento

Sistema Operacional: Ubuntu 11.10
IDE: NetBeans 7.0.1
Compilador: GCC 4.6.1

6.4.2 Resultados

Os testes estão ainda em execução e os resultados serão apresentados parcialmente na apresentação e, em "definitivo" na versão final do Relatório de Atividades.

6.5 Paralelização da metaheurística

A idéia de como paralelizar a metaheurística está formada; contudo não é uma tarefa trivial, haja visto que alguns algoritmos sequer são paralelizáveis ou usar paralelismo nos mesmos inclusive piora o tempo de execução.

A proposta inicial de paralelização da metaheurística, a ser feita na disciplina BCC 391 – Monografia II, segue o Algoritmo 4.

Algoritmo 4: PropostaParalelizacao

Entrada: n, $nProcessors$
Saída: $solOtimizada$

```
1  início
2      para i = 1 .. nProcessors faça
3          solParc[i] ← constroiSolucaoAleatoria(n)
4          solParc[i] ← Nonggis(SolParc[i])
5      fim
6      solOtimizada ← melhorSolucao(solParc)
                  // melhorSolucao é uma função auxiliar que, ao receber um vetor de
       soluções, retorna aquela que possui melhor função objetivo
7      solOtimizada ← BuscaTabu(SolOtimizada)
8      solOtimizada ← ReconexaoPorCaminhos(SolOtimizada)
9      return solOtimizada
10 fim
```

7 Cronograma de atividades – BCC391 - Monografia II

A Tabela 2 contém o cronograma de atividades proposto para a disciplina BCC391 em 2012/1.

Atividades	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**
Revisão de literatura	X	X			
Implementação de formulações	X	X			
Testes	X	X	X		
Paralelização do Nonggis			X	X	
Redigir a Monografia			X	X	X
Apresentação do Trabalho				X	X

Tabela 2: Cronograma de Atividades para a Disciplina BCC391.

8 Trabalhos Futuros

Como trabalhos futuros:

- Aplicar a metaheurística a outros problemas conhecidos da literatura;
- Compará-la a outras metaheurísticas.

Referências

[1] A. Allahverdi, J. N. D. Gupta, and T. Aldowaisan. A review of scheduling research involving setup considerations. *OMEGA*, 27:219–239, 1999.

[2] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032, 2008.

[3] Ü. Bilge, M. Kurtulan, and F. Kirac. A tabu search algorithm for the single machine total weighted tardiness problem. 176:1423–1435, 2007.

[4] D. Biskup and M. Feldmann. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. 28:787–801, 2001.

[5] G. D. Campos and H. T. Y. Yoshizaki Pand . P. Belfiore. Algoritmo genético e computação e paralela para problemas de roteirização de veículos com janelas de tempo e entregas fracionadas. In *Gestão e Produção*, volume 13, pages 271–281, 2006.

[6] P. C. Chang. A branch and bound approach for single machine scheduling with earliness and tardiness penalties. 37(10):133–144, 1999.

[7] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. In *Operations Research*, volume 12, pages 568–581, 1964.

[8] I. M. Coelho, S. Ribas, and M. J. F. Souza. Um algoritmo baseado em grasp, iterated local search para a otimização do planejamento operacional de lavra. In *Anais do XI Encontro de Modelagem Computacional*, Volta Redonda, 2008.

[9] J. Du and J. Y. T. Leung. Minimizing total tardiness on one machine is np-hard. 15:483–495, 1990.

[10] M. Feldmann and D. Biskup. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. 44:307–323, 2003.

[11] 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.

[12] V. J. Garcia, A. S. Mendes, P. M. França, and P. A. Moscato. Algoritmo memético paralelo aplicado a problemas de sequenciamento em máquina simples. In *Anais do XXXIII Simpósio Brasileiro de Pesquisa Operacional – XXXIII SBPO*, pages 871–981, Campos do Jordão, 2001.

[13] F. Glover. Tabu search and adaptative memory programming – advances, applicantions and challenges. In *Interfaces in Computer Sciences and Operations Research*, pages 1–75, 1996.

[14] F. Glover and M. Laguna. Tabu search. *Boston: Kluwer Academic Publishers*, 1997.

[15] A. C. Gomes Jr., C. R. V. Carvalho, P. L. A. Munhoz, and M. J. F. Souza. Um método heurístico híbrido para a resolução do problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção. In *Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional, SBPO*, pages 1649–1660, Fortaleza, 2007.

[16] F. A. Almeida Gonçalves, M. J. Freitas Souza, and S. R. de Souza. Sequenciamento em uma máquina: Otimização heurística via multiprocessamento paralelo. 2010.

[17] V. Gordon, J.-M. Proth, and C. Chu. A survey of the state-of-the-art of common due date assignment and scheduling research. 139:1–25, 2002.

[18] S. R. Gupta and J. S. Smith. Algorithms for single machine total tardiness scheduling with sequence dependent setups. 175:722–739, 2006.

[19] R. M. Hallah. Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. 34:3126–3142, 2007.

[20] C. M. Hino, D. P. Ronconi, and A. B. Mendes. Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. 160:190–201, 2005.

[21] J. R. Jackson. Scheduling a production line to minimize maximum tardiness. Technical report, UCLA, Management Science Research Project, 1955.

[22] R. J. W. James. Using tabu search to solve the common due date early/tardy machine scheduling problem. 24(3):199–208, 1997.

[23] C. Y. Lee and J. Y. Choi. A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. 22:857–869, 1995.

[24] F. Li. Single machine earliness and tardiness scheduling. 96:546–558, 1997.

[25] C. F. Liaw. A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. 26:679–693, 1999.

[26] R. J. Moraga, G. E. Whitehouse, G. W. Depuy, B. C. Neyveli, and S. M. Kuttuva. Solving the vehicle routing problem using the meta-raps approach. In *International Conference on Computers and Industrial Engineering*, pages 1–6, Montreal, 2001.

[27] R. B. MUhammad. A parallel local search algorithm for euclidean steiner tree problem. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, International Conference on and Self-Assembling Wireless Networks, International Workshop on, IEEE Computer Society*, volume 0, pages 157–164, Los Alamitos, 2006.

[28] G. Rabadi, M. Mollaghasemi, and G. C. Anagnostopoulos. A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. 31:1727–1751, 2004.

[29] M. G. C. Resende and C. C. Ribeiro. *Parallel metaheuristics: a new class of algorithms*, chapter 14, *Parallel Greedy Randomized Adaptive Search Procedures*, pages 315–344. John Wiley and Sons, 2005.

[30] S. Ribas, I. M. Coelho, M. J. F. Souza, and D. Menotti. Parallel iterated local search aplicado ao planejamento operacional de lavra. In *Anais do XLI Simpósio Brasileiro de Pesquisa Operacional – XLI SBPO*, 2009.

[31] S. Ribas, M. H. de P. Perche, I. M. Coelho, P. L. A. Munhoz, M. J. F Souza, and A. L. L. Aquino. Grasp, tabu search and path relinking for solving total earliness/tardiness single machine scheduling problem with distinct due windows and sequence-dependent setups. In *Anais do IX Congresso Brasileiro de Redes Neurais / Inteligência Computacional – IX CBRN*, Ouro Preto, 2009.

[32] 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.

[33] F. F. Ribeiro. Um algoritmo genético adaptivo para a resolução do problema de sequenciamento em um máquina com penalização por antecipação e atraso da produção. Master's thesis, Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, 2009.

[34] B. F. Rosa. Heurísticas para o problema de seqüenciamento em uma máquina com penalidades por antecipação e atraso da produção. Master's thesis, Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, 2009.

[35] 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.

[36] M. J. F Souza, L. S. Ochi, F. A. C. A Gonçalves, and P. H. V. Penna. Grasp, tabu search and path relinking for solving total earliness/tardiness single machine scheduling problem with distinct due windows and sequence-dependent setups. In *Proceedings of the XXIX CILAMCE*, volume 1, pages 1–15, Maceió, 2008.

[37] M. J. Freitas Souza. *Inteligência Computacional para Otimização*.

[38] N. Standerski. Aplicação de algoritmos genéticos paralelos a problemas de grande escala vmi - vendor managed inventory. In *Anais do XXXV Simpósio Brasileiro de Pesquisa Operacional – XXXV SBPO*, Natal, 2003.

[39] G. Wan and B. P. C. Yen. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. 142:271–281, 2002.

[40] K.-C. YING. Minimizing earliness-tardiness penalties for common due date single-machine scheduling problems by a recovering beam search algorithm. 55:494 – 502, 2008.