UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

DESENVOLVIMENTO DE METAHEURÍSTICAS PARA O PROBLEMA DA ÁRVORE GERADORA MÍNIMA GENERALIZADO

DISSERTAÇÃO DE MESTRADO

Fernando de Cristo

Santa Maria, RS, Brasil
2008

DESENVOLVIMENTO DE METAHEURÍSTICAS PARA O PROBLEMA DA ÁRVORE GERADORA MÍNIMA GENERALIZADO

por

Fernando de Cristo

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Engenharia de Produção, Área de Concentração em Tecnologia da Informação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Engenharia de Produção**.

Orientador: Prof. Dr. Felipe Martins Müller

Santa Maria, RS, Brasil

2008

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia de Produção

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

DESENVOLVIMENTO DE METAHEURÍSTICAS PARA O PROBLEMA DA ÁRVORE GERADORA MÍNIMA GENERALIZADO

elaborada por
Fernando de Cristo

como requisito parcial para obtenção do grau de
Mestre em Engenharia de Produção

COMISSÃO EXAMINADORA:

Felipe Martins Müller, Dr.
(Presidente/Orientador)

José Vicente Canto dos Santos, Dr. (UNISINOS)

Olinto Cezar Bassi de Araújo, **Dr.** (UFSM)

Santa Maria, 20 de março de 2008.

Dedico este trabalho, em homenagem póstuma, a
Irfe Teresinha de Cristo, minha mãe.

AGRADECIMENTOS

Meus sinceros agradecimentos a preciosa ajuda recebida de meu orientador Prof. Dr. Felipe Martins Müller que sempre que possível ofereceu valorosas contribuições a este trabalho.

Ao apoio financeiro recebido da CAPES através do programa PIQDTEC.

Agradecimento ao Colégio Agrícola de Frederico Westphalen por conceder a licença de afastamento em tempo integral durante o decorrer do curso de mestrado.

Aos colegas, professores e funcionários do PPGEP-UFSM pelo companheirismo e receptividade.

Em especial agradeço a minha esposa Edinara Filipiak de Cristo pelo companheirismo e apoio incondicional em todos os momentos em que se fez necessário.

RESUMO

O problema da árvore geradora mínima generalizado está presente em várias situações do mundo real, tais como no contexto das telecomunicações, transportes e agrupamento de dados, nas quais uma rede de grupos precisa ser conectada utilizando um nodo de cada grupo. Nesse trabalho é apresentado o projeto e a implementação de um algoritmo de busca tabu com reconexão de caminhos e busca local iterativa para o problema da árvore geradora mínima generalizado e sua variante com pelo menos um vértice por grupo. Nos testes computacionais foram utilizadas 271 instâncias da TSPLIB geradas através dos métodos de agrupamento *Center Clustering* e *Grid Clustering*, e mais 20 instâncias para a extensão do problema com pelo menos um vértice por grupo. Os resultados demonstram a eficiência do algoritmo proposto na obtenção de soluções satisfatórias para os dois problemas.

Palavras-chaves: Problema da Árvore de Geradora Mínima Generalizado. Otimização em Grafos. Metaheurísticas.

ABSTRACT

The generalized minimum spanning tree problem is present in several situations of the real world, such as in the context of the telecommunications, transports and grouping of data, where a net of necessary clusters to be connected using a node of each cluster. In that work it is presented the project and the implementation of an algorithm of tabu search with path relinking and iterated local search for the generalized minimum spanning tree problem and your variant with at least one vertex by group. In the computational tests 271 instances of TSPLIB were used generated through the grouping methods Center Clustering and Grid Clustering, and more 20 instances for the extension of the problem with at least one vertex by group. The results demonstrate the efficiency of the algorithm proposed in the obtaining of satisfactory solutions for the two problems.

Keywords: Generalized Minimum Spanning Tree Problem. Graph Optimization. Metaheuristics.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AGM – Árvore Geradora Mínima

BT+RC+BLI – Busca Tabu com Reconexão de Caminhos e Busca Local Iterativa

BLI – Busca Local Iterativa

BT – Busca Tabu

CE – Conjunto de Elite

E-GMSTP – *Exactly one node per cluster - Generalized Minimum Spanning Tree Problem*

GRASP – *Greedy Randomized Adaptive Search Procedure*

GA – *Genetic Algorithm*

JDK – *Java Development Kit*

L-GMSTP – At Least one node per cluster – Generalized Minimum Spanning Tree Problem

LS – *Local Search*

PAGMG – Problema da Árvore Geradora Mínima Generalizado

PC – *Personal Computer*

PM-PAGMG – Problema da Árvore Geradora Mínima Generalizado com Pelo Menos um Vértice por Grupo

RC – Reconexão de Caminhos

TS – *Tabu Search*

TSPLIB - *Traveling Salesman Problem Library*

SUMÁRIO

1 INTRODUÇÃO

Esta dissertação aborda a utilização de algoritmos heurísticos para a resolução de dois problemas de otimização combinatória e é motivada e justificada pela complexidade dos algoritmos necessários para encontrar uma solução ótima para tais problemas.

A otimização combinatória é um ramo da ciência da computação que estuda problemas de otimização em conjuntos. Em um problema de otimização tem-se uma função objetivo e um conjunto de restrições, ambos relacionados às variáveis de decisão. O problema pode ser de minimização ou de maximização da função objetivo. A resposta para o problema, ou seja, o "ótimo global" é o menor (ou maior) valor possível para a função objetivo para o qual o valor atribuído às variáveis não viole nenhuma restrição. Em alguns casos, chega-se a valores cuja alteração discreta conduz a resultados melhores, mas que não são também o ótimo global, essas soluções são denominadas de ótimo local. Há muitos problemas de otimização, e alguns deles apresentam métodos exatos e eficientes de resolução, outros levam à necessidade de métodos não-exatos, ou seja, métodos heurísticos, uma vez que sua formulação e/ou resolução exatas levariam a uma complexidade elevada, tornando-o intratável computacionalmente. (REEVES, 1993)

O Problema da Árvore Geradora Mínima Generalizado (PAGMG), objeto de estudo neste trabalho, é da classe NP-Difícil, não sendo conhecido nenhum algoritmo exato que o resolva em tempo polinomial. Para demonstrar que esse problema é NP-Difícil, Myung et al. (1995) utilizaram o *node cover problem*, que é conhecidamente NP-Completo, e demonstraram que ele pode ser reduzido polinomialmente para o PAGMG. O PAGMG consiste em encontrar a árvore geradora mínima que interligue todos os grupos de vértices de um grafo ponderado utilizando exatamente um vértice de cada grupo.

Além do PAGMG, também faz parte desta dissertação o desenvolvimento de heurísticas para a solução de uma extensão do PAGMG onde o objetivo é encontrar a árvore de menor custo que interligue todos os grupos passando por pelo menos

um vértice de cada grupo. Este problema foi introduzido por Dror et al. (2000) e por ser uma extensão do PAGMG também é da classe NP-Difícil.

O objetivo desse trabalho é avaliar a aplicabilidade das metaheurísticas desenvolvidas para a resolução do PAGMG e sua extensão, deste ponto em diante denominada pela sigla PM-PAGMG.

O capítulo 2 trata sobre a definição dos problemas, bem como, a revisão de literatura a respeito dos mesmos. O capítulo 3 discorre sobre as heurísticas desenvolvidas. O capítulo 4 apresenta e analisa os resultados obtidos. O capítulo 5 trata das conclusões e sugestões para trabalhos futuros.

2 O PROBLEMA DA ÁRVORE GERADORA MÍNIMA GENERALIZADO

O Problema da Árvore Geradora Mínima Generalizado (PAGMG) foi introduzido por Myung et al. (1995) e trata-se de, em um grafo não direcionado em que os vértices são divididos exaustivamente em grupos de vértices mutuamente exclusivos, localizar a árvore de menor custo a qual inclua exatamente um vértice de cada conjunto de vértices. Neste mesmo artigo Myung et al. demonstra que este é um problema da classe NP-Difícil e a menos que $P = NP$, não há um algoritmo heurístico de tempo polinomial para o PAGMG. Estes autores forneceram quatro formulações de programação inteira e desenvolveram um algoritmo *branch-and-bound*. Eles reportaram a solução exata para instâncias criadas aleatoriamente com até 100 vértices. O PAGMG apresenta inúmeras aplicações práticas na área de transportes, energia, telecomunicações, biologia molecular e agrupamento de dados (HAOUARI e CHAOUACHI, 2006), além de aplicações em problemas de localização de facilidades, em problemas de roteamento, em projeto de circuitos integrados, planejamento da produção e construção de redes de irrigação em ambientes desertos (DROR et al., 2000). Um exemplo de árvore geradora mínima incluindo um vértice de cada grupo pode ser visto na Figura 1.

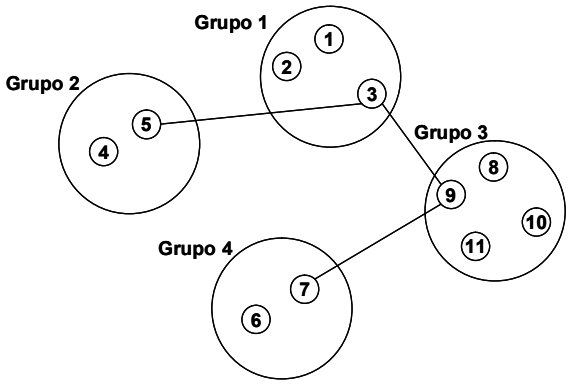


Figura 1 - Problema da Árvore Geradora Mínima Generalizado

Considere que $G = (V, E)$ é um grafo não direcionado dividido em $|K|$ grupos $V_k, k \in K$. Cada arco $e = \{i, j\} \in E$ possui um custo $c_e \in \Re^+$. O PAGMG tem como objetivo encontrar a árvore de custo mínimo incluindo exatamente um vértice de

cada grupo. Há também uma variante do PAGMG onde o problema consiste em encontrar uma árvore de custo mínimo incluindo ao menos um vértice de cada grupo. Este problema foi introduzido por Ihler et al. (1999) como um caso particular do Problema da Árvore de Steiner Generalizado sob o nome "*Class Tree Problem".* Na literatura internacional utilizam-se as siglas *Generalized Minimum Spanning Tree* (GMSTP ou E-GMSTP) para se referir a versão original do problema e "L" de *"at Least"* (L-GMSTP) para se referir a variante com pelo menos um vértice por grupo. Na Figura 2 pode-se visualizar um exemplo de árvore geradora mínima incluindo pelo menos um vértice por grupo.

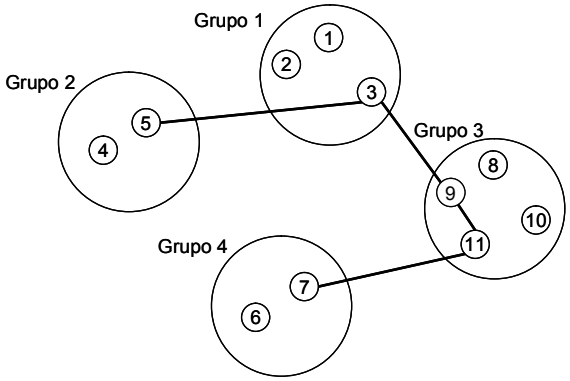


Figura 2 – Problema da Árvore Geradora Mínima Generalizado com pelo menos um vértice por grupo

2.1 Problema da Árvore Geradora Mínima

Dado um grafo conectado, não direcionado, uma árvore geradora deste grafo é um sub-grafo em forma de uma árvore que se conecta a todos os vértices. Um grafo simples pode possuir várias árvores geradoras diferentes. Pode-se associar a cada um dos arcos um peso, o qual é um número representando o quanto desfavorável ele está, e usar isto para associar um peso para a árvore geradora somando os pesos dos arcos naquela árvore geradora. Uma árvore geradora mínima ou árvore geradora de custo mínimo é então uma árvore geradora com custo menor ou igual ao custo de todas as outras árvores geradoras daquele grafo. (CORMEN, 2003)

Na Figura 3 é apresentado um exemplo de árvore geradora mínima para um grafo de sete vértices. Neste exemplo o custo da árvore geradora mínima é de 39,

obtido através da soma dos custos das arestas em negrito, que formam a árvore geradora mínima para este grafo.

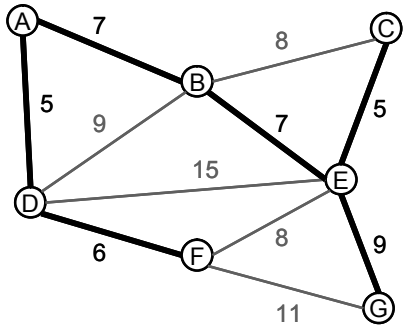


Figura 3 – Árvore Geradora Mínima

Atualmente são, freqüentemente, usados dois algoritmos, algoritmo de Prim (1957) e algoritmo de Kruskal (1956). Todos são algoritmos gulosos, algoritmos que tomam a melhor decisão entre as possíveis a cada passo de sua execução, que rodam em tempo polinomial, então o problema de encontrar tais árvores pertence à classe de complexidade P.

2.1.1 Algoritmo de Prim para o Problema da Árvore Geradora Mínima

O algoritmo de Prim (1957) é um algoritmo em teoria dos grafos que busca uma árvore geradora mínima para um grafo conexo com pesos. O subconjunto S forma uma única árvore, e a aresta adicionada a S é sempre uma aresta de peso mínimo conectando a árvore a um vértice que não esteja na árvore. A árvore começa por um vértice qualquer e cresce até que inclua todos os vértices em V. A cada passo, uma aresta “leve” é adicionada à árvore S, conectando S a um vértice não conectado de $G_S = (V, S)$. De acordo com a descrição anterior, quando o algoritmo termina, as arestas em S formam uma árvore geradora mínima.

A ordem de complexidade para o algoritmo de Prim é $O(|E|\log|V|)$, onde E é o conjunto de arestas e V é o conjunto de vértices do grafo.

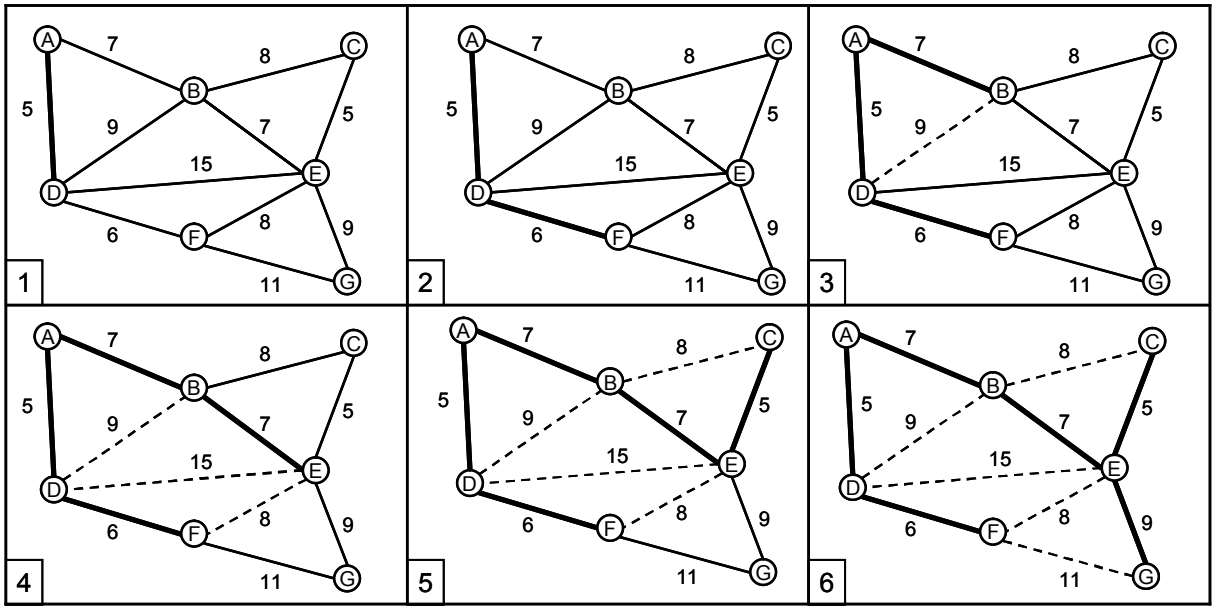


Figura 4 – Algoritmo de Prim para o Problema da Árvore Geradora Mínima

A Figura 4 ilustra a aplicação passo a passo do algoritmo de Prim para obtenção da árvore geradora mínima para o grafo apresentado na Figura 3. No passo 1 o vértice A é selecionado para iniciar a árvore e a partir da análise dos vértices adjacentes a A (B e D), o vértice D que possui a aresta de menor custo (AD=5) até A é selecionado. No passo 2 são analisadas todas as arestas partindo dos vértices A e D e a aresta de menor custo que conduza a um vértice não conectado à árvore é selecionada (DF=6). No passo 3 são analisadas as arestas partindo de A, D ou F, é selecionada aquela com menor custo que leve a um vértice não conectado (AB=7), com a inclusão da aresta AB o uso da aresta DB fica descartado pois a mesma conduz a um vértice já conectado o que formaria um ciclo desrespeitando a definição de árvore. No passo 4 é selecionada entre as arestas BC, BE, DE, FE e FG a aresta BE=7 por ser a de menor custo entre estas, deste modo pode-se descartar as arestas DE e FE por serem formadoras de ciclos. No passo 5 as arestas disponíveis são BC, EC, EG e FG, das quais a que possui menor custo é EC=5 que é adicionada a árvore, com isto fica descartada a aresta BC. No passo 6 entre as arestas restantes EG e FG, é selecionada a aresta EG=9 por possuir custo inferior ao de FG e assim está formada a árvore interligando todos vértices do grafo. O custo final da árvore é dado pela soma das arestas que a compõem que neste caso encontram-se em destaque e tem um custo acumulado de 39. As arestas pontilhadas foram descartadas pelo algoritmo por formarem ciclos.

2.1.2 Algoritmo de Kruskal para o Problema da Árvore Geradora Mínima

O algoritmo de Kruskal (1956) é um algoritmo em teoria dos grafos que busca uma árvore geradora mínima para um grafo conexo com pesos. Isto significa que ele encontra um subconjunto das arestas que forma uma árvore que inclui todos os vértices, onde o peso total, dado pela soma dos pesos das arestas da árvore, é minimizado. Se o grafo não for conexo, então ele encontra uma floresta geradora mínima (uma árvore geradora mínima para cada componente conexo do grafo).

Inicialmente o algoritmo cria uma floresta F (um conjunto de árvores), onde cada vértice no grafo é uma árvore separada. Em seguida cria-se um conjunto S contendo todas as arestas do grafo ordenadas do menor para o maior custo. A partir daí a cada passo do algoritmo remove-se uma aresta de peso mínimo de S até que S esteja vazio. Se a aresta removida conecta duas árvores distintas esta aresta passa a fazer parte da árvore geradora mínima e as duas árvores são combinadas em uma única, caso contrário a aresta é descartada. Ao fim do algoritmo, a floresta tem apenas um componente e forma uma árvore geradora mínima do grafo, se este for conectado.

Com o uso de uma estrutura de dados aceitável, o algoritmo de Kruskal pode ser executado em tempo $O(|E|\log|V|)$, onde E é o conjunto de arestas e V é o conjunto de vértices do grafo.

A Figura 5 ilustra a aplicação passo a passo do algoritmo de Kruskal para obtenção da árvore geradora mínima do grafo apresentado na Figura 3. Inicialmente, todas as arestas são ordenadas em ordem crescente de custo, sendo $S \leftarrow \{AD, CE, DF, AB, BE, BC, FE, BD, EG, FG, DE\}$. No passo 1 é inserida a primeira aresta (AD=5) interligando os vértices A e D em uma única árvore de custo 5. No passo 2 a segunda aresta é inserida (CE=5) e forma uma segunda árvore a qual também possui custo 5. No passo 3 a terceira aresta é inserida (DF=6), esta aresta une o vértice F na árvore formada por A e D sendo que o custo desta aumenta para 11. No passo 4 a quarta aresta é inserida (AB=7) acrescentando o vértice B a árvore formada pelos vértices A, D e F, com o custo desta subindo para 18. No passo 5 a quinta aresta (BE=7) é inserida unindo a árvore de vértices A, B, D e F e a árvore de vértices C e E com um custo total de 30. No passo 6 a sexta aresta (BC), a sétima

(EF) e a oitava (BD) são descartadas, por unirem vértices pertencentes a uma mesma árvore e a nona aresta (EG) é então inserida e o custo da árvore passa para 39, as arestas restantes não serão utilizadas pois a árvore geradora mínima já esta completa.

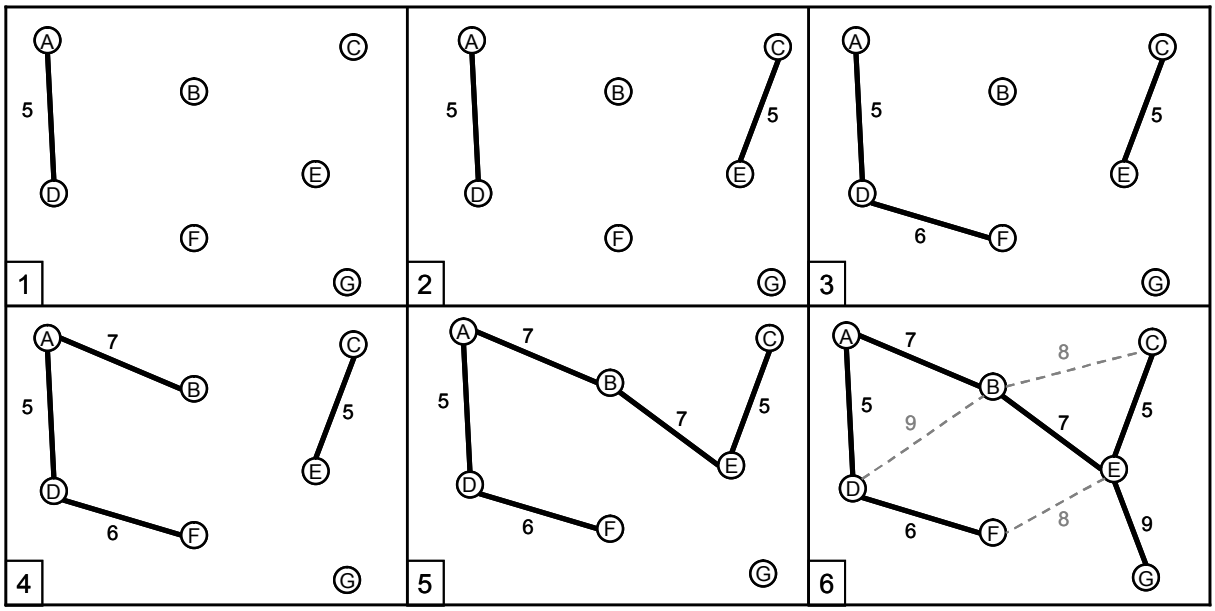


Figura 5 – Algoritmo de Kruskal para o Problema da Árvore Geradora Mínima

2.2 Trabalhos Relacionados

Como já relatado anteriormente o PAGMG foi introduzido por Myung et al. (1995). No artigo os autores, primeiramente, conceituam o problema e comprovam seu grau de complexidade computacional através da redução do problema proposto para o já sabidamente NP-Completo *Node Cover Problem.* São desenvolvidas duas formulações matemáticas para este problema. Os autores implementam um procedimento para buscar valores de limite inferior (valor igual ou menor ao valor ótimo), uma heurística para produzir uma solução inicial e um algoritmo de *Branch and Bound* para encontrar a solução ótima. Estes algoritmos foram testados em dois conjuntos de instâncias, sendo o primeiro composto por 140 instâncias utilizadas anteriormente em estudos relacionados ao problema do caixeiro viajante e adaptadas para o PAGMG e o segundo grupo de instâncias foi criado pelos próprios autores através de dados gerados aleatoriamente. Segundo os autores, independentemente do tipo de instância testada, à medida que o número de grupos

ou de vértices vai sendo incrementado o problema vai se tornando mais complexo de ser resolvido.

Dror et al. (2000) propôs uma variante do problema original onde se usa pelo menos um vértice por grupo para formar a árvore geradora, ao invés de exatamente um vértice por grupo como proposto em Myung et al. (1995). Além da definição, formulações matemáticas e cálculos a respeito da complexidade da nova variante do PAGMG, os autores apresentam três heurísticas construtivas, uma busca local e um algoritmo genético desenvolvidos para solucionar o PM-PAGMG. No mesmo trabalho foi desenvolvido ainda um conjunto de 20 instâncias para testes das heurísticas e do algoritmo genético desenvolvidos pelos autores.

Feremans et al. (2001) em seu artigo aponta erros nas formulações propostas anteriormente por Dror et al. (2000) e apresenta uma nova formulação para o PAGMG. Em sua tese de doutorado Feremans (2002) estuda várias formulações para o PAGMG a fim de encontrar a mais adequada para o desenvolvimento de um algoritmo exato. Também desenvolve uma heurística do tipo Busca Tabu para encontrar valores de limite superior (valor maior ou igual ao valor ótimo) para o método exato. Utilizando seu algoritmo exato Feremans testou um conjunto de 169 instâncias, originalmente desenvolvidas para uma generalização do problema do caixeiro viajante por Fischetti et al. (1995), nas quais, para 150 delas o algoritmo proposto encontrou o valor ótimo em um tempo inferior a duas horas. Para as outras 19 instâncias foram relatados os valores de limite superior encontrados pelo método exato dentro do tempo limite de duas horas de processamento proposto nos testes do trabalho em questão. Em sua tese Feremans definiu as siglas E-GMSTP para o PAGMG com exatamente um vértice por grupo e L-GMSTP para o PM-PAGMG, extensão do problema com ao menos um vértice por grupo, em inglês "*at least*". Ainda neste trabalho a autora executou testes com 20 instâncias geradas por Dror et al. (2000) obtendo o valor ótimo para todas as instâncias através de uma versão adaptada do algoritmo exato desenvolvido para o E-GMSTP.

Ghosh (2003), em seu artigo descreve a implementação de seis metaheurísticas para o PAGMG, sendo duas versões de busca tabu e quatro de Busca em Vizinhança Variável. A primeira busca tabu segue uma linha de desenvolvimento mais tradicional conforme proposta originalmente por Glover (1986), a segunda implementa um mecanismo adicional de penalidades o qual penaliza os vértices mais utilizados em trocas e que gerem soluções com custo

superior ao da solução corrente. As seis metaheurísticas desenvolvidas foram testadas em 36 instâncias com distâncias euclidianas, todas com valor ótimo já conhecido, sendo que, a busca tabu com uso de penalidades obteve o resultado ótimo para 30 destas instâncias, se mostrando superior aos demais métodos implementados no trabalho. Por outro lado, neste mesmo estudo Ghosh aponta que em instâncias maiores sua versão de busca tabu com penalidades embora seja mais rápida do que os demais métodos retorna soluções de qualidade inferior às obtidas pelas implementações de Busca em Vizinhança Variável.

Golden et al. (2005) desenvolvem duas heurísticas para o PAGMG, uma busca local e um algoritmo genético. As heurísticas são submetidas às mesmas instâncias testadas por Feremans et al. (2001), sendo que o algoritmo genético obteve o valor ótimo para 149 instâncias e a busca local para 145 instâncias de um total de 150. No tocante ao tempo de execução a heurística de busca local demonstra em geral ser bem superior ao algoritmo genético. Testes também são realizados em 19 instâncias para as quais o valor ótimo não é conhecido, nestas instâncias novamente é comprovada a superioridade do algoritmo genético em relação à busca local em termos de qualidade de solução, sendo que, em 10 das 19 instâncias o algoritmo genético obteve melhores resultados do que a busca local e a mesma não conseguiu superar o algoritmo genético em nenhuma das instâncias testadas.

Rocha e Alvarenga (2005) apresentam uma metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP) para o PM-PAGMG com pelo menos um vértice por grupo. Trata-se de uma implementação de GRASP puro onde a primeira fase da heurística consiste em gerar uma solução inicial através de um método construtivo guloso aleatorizado, na segunda etapa é aplicada uma heurística de busca local sob a solução gerada pelo método construtivo, a cada iteração se uma solução melhor que a solução incumbente for encontrada então a solução incumbente é atualizada. Para utilização na primeira fase do GRASP foi desenvolvida uma versão adaptada do algoritmo de Kruskal (1956). Na segunda fase foi utilizada uma busca local similar a utilizada por Golden et al. (2005). Em testes com 10 das 20 instâncias geradas por Dror et al. (2000) o algoritmo proposto obteve bons resultados.

Ferreira et al. (2006) apresentam duas versões da metaheurística GRASP para o PAGMG uma GRASP pura e outra com o uso de Memória Adaptativa, ambas

utilizando quatro diferentes métodos de construção e um método de busca local. Na versão com memória adaptativa esta memória é utilizada no sentido de manter um conjunto de soluções de elite que contempla as quatro melhores soluções encontradas até o momento com o intuito de selecionar entre estas uma para realizar o procedimento de reconexão de caminhos. Ferreira et al. (2006) expõe resultados alcançados com 37 instâncias de pequeno porte, todas com valor ótimo conhecido, onde o algoritmo pelos autores desenvolvido atinge o valor ótimo em todos os casos com tempos bem inferiores aos do algoritmo exato de Feremans et al. (2001). Neste mesmo artigo os autores descrevem ainda testes realizados com outras 12 instâncias de médio e grande porte das quais o valor ótimo não é conhecido.

Wang et al. (2006) descreve a implementação de uma heurística busca tabu e um algoritmo heurístico de relaxação, sendo este último utilizado para produzir valores de limitante inferior para as instâncias testadas. Nos testes a metaheurística busca tabu desenvolvida alcançou o valor ótimo para 152 de um total de 194 instâncias de teste. Em relação aos demais trabalhos este diferencia-se pela aplicação de uma lista restrita de elementos candidatos durante as iterações da busca tabu.

Harouari et al. (2006) relata o desenvolvimento de um algoritmo genético baseado no proposto em Dror et. al. (2000). Este algoritmo alcançou excelentes resultados em instâncias para o PM-PAGMG. Neste artigo são feitas varias comparações de desempenho com outras heurísticas, sendo que, o algoritmo genético se mostra superior na maioria dos casos.

Em Öncan et al. (2007), os autores descrevem uma heurística busca tabu baseada em atributos para o PAGMG. Para esta heurística busca tabu baseada em atributos utilizou-se novas estratégias de vizinhança. Em virtude dos bons resultados obtidos em instâncias de pequeno porte, um conjunto estendido de instâncias de teste TSPLIB (REINELT, 1991) foi gerado. Além disso, uma adaptação do algoritmo de busca tabu foi proposta para a variante do PAGMG na qual pelo menos um vértice de cada grupo deve ser incluído na árvore. Os resultados obtidos pelos autores em sua maioria superam os trabalhos publicados até então tanto em nível de qualidade de solução como em nível de tempo computacional.

Com base nos estudos feitos a partir dos trabalhos anteriormente relacionados optou-se por seguir uma estratégia de desenvolvimento do trabalho

que contemplou a inserção de novas técnicas em metaheurísticas que se demonstraram boas para aplicação nos dois problemas tratados nesta dissertação. Sendo assim na seção seguinte estão descritas as heurísticas propostas para ambos os problemas.

3 METAHEURÍSTICAS DESENVOLVIDAS

Na busca de soluções para os problemas anteriormente expostos foram desenvolvidas várias heurísticas e metaheurísticas, sendo todas avaliadas em diversos testes com várias instâncias para os problemas. Destas, algumas foram selecionadas por terem apresentado melhores resultados durante os testes computacionais realizados para participarem de um estudo mais aprofundado onde pode-se fazer um refinamento maior dos parâmetros necessários a tais métodos.

Inicialmente, optou-se por seguir duas linhas de desenvolvimento, implementando um algoritmo genético e uma busca tabu para o PAGMG, conforme descrito em Cristo et al. (2007c). A proposta de algoritmo genético baseou-se no estudo de outros trabalhos relacionados como Dror et al. (2000) e Golden et al. (2005). O algoritmo genético não apresentou os resultados esperados, e então decidiu-se iniciar o estudo de alguns métodos adicionais para incrementar o seu desempenho. Partiu-se então para o estudo do artigo de Ferreira et al. (2006) onde constatou-se os bons resultados obtidos através da aplicação da técnica de reconexão de caminhos sobre a metaheurística GRASP. Sendo assim decidiu-se buscar maiores informações a respeito desta técnica através do estudo de Glover et al. (2000). Depois de realizado o estudo visando buscar a maneira mais adequada de implementar a reconexão de caminhos, desenvolveu-se uma nova versão do algoritmo genético a qual contava então com este mecanismo. A aplicação da reconexão de caminhos sob o algoritmo genético propiciou uma sensível melhoria nos resultados, sendo que, nos testes realizados alcançou-se o valor ótimo para 32 das 34 instâncias utilizadas, enquanto que sem a reconexão de caminhos o algoritmo genético conseguiu atingir o valor ótimo somente para 21 das 34 instâncias testadas.

Na outra linha de desenvolvimento constatou-se situação idêntica, onde a adição da técnica de reconexão de caminhos a metaheurística busca tabu proporcionou também uma melhora nos resultados. Implementou-se três variações de busca tabu. Primeiramente, desenvolveu-se uma busca tabu simples, com qual atingiu-se o valor ótimo para 29 entre 34 instâncias e um desvio das soluções em

relação ao valor ótimo de 0,12%. Passou-se então a uma segunda fase no desenvolvimento onde foi incorporada a reconexão de caminhos ao final do processo de busca tabu, sendo realizada entre as 10 melhores soluções encontradas durante a busca. Para esta nova versão conseguiu-se atingir o ótimo também para 29 das 34 instâncias, porém com um desvio médio em relação ao valor ótimo de 0,06%. Buscando-se melhorar ainda mais o processo de busca e ampliar a capacidade do método de reconexão de caminhos, decidiu-se por torná-lo iterativo, ou seja, executá-lo a cada iteração da busca tabu. Com esta nova dinâmica de execução conseguiu-se atingir o ótimo para 32 entre as 34 instâncias utilizadas nestes testes com um desvio médio em relação ao valor ótimo de 0,04%. Neste ponto o desenvolvimento das duas primeiras versões de busca tabu foi interrompido para o aprimoramento da terceira versão, na qual se obteve os melhores resultados.

Após um extenso trabalho de refinamento do código e ajuste de parâmetros das metaheurísticas, produziu-se uma publicação com os resultados preliminares deste estudo (veja Cristo et al. (2007c)), a qual já foi referida anteriormente. Feitas estas melhorias nas metaheurísticas propostas chegou-se ao resultado de 32 entre 34 instâncias resolvidas na otimalidade para o algoritmo genético e todas as 34 instâncias resolvidas na otimalidade para a busca tabu, nesta mesma publicação foram utilizados dados de Ferreira et al. (2006) para um comparativo entre os resultados para as instâncias testadas.

Levando em consideração os bons resultados obtidos nesta fase, optou-se por prosseguir o desenvolvimento das heurísticas e depois de mais alguns testes conforme relatado em Cristo et al. (2007b) e Ferreira et al. (2007), decidiu-se interromper os trabalhos de desenvolvimento do algoritmo genético, uma vez que os resultados obtidos favoreciam a implementação da busca tabu, e o desenvolvimento em duas frentes tomava muito tempo e impossibilitava uma maior dedicação a um dos métodos visando aprofundar-se e obter melhores resultados. Após a interrupção no desenvolvimento do algoritmo genético houve uma dedicação exclusiva dos esforços de desenvolvimento e testes em melhorar a metaheurística busca tabu. Com isto chegou-se a novos resultados relatados em Cristo et al. (2007a), no qual em testes para 150 instâncias da TSPLIB (REINELT, 1991) geradas através do método de agrupamento *Center Clustering* e *Grid Clustering* testadas por Feremans et al. (2001) atingiu-se o valor ótimo para 146 instâncias.

Nas subseções a seguir são descritas todas as heurísticas utilizadas neste trabalho para solução do PAGMG. Trata-se de cinco métodos construtivos, duas buscas locais e uma metaheurística do tipo busca tabu com reconexão de caminhos e busca local iterativa. Para o PM-PAGMG foi desenvolvida uma heurística adicional que tem por objetivo inserir vértices adicionais em soluções contendo apenas um vértice por grupo, na tentativa de minimizar o custo das mesmas.

3.1 Construtivo C1

A idéia central deste método construtivo consiste em gerar uma solução a partir da seleção dos vértices com menor distância média em relação aos seus vizinhos mais próximos de outros grupos. A Figura 6 traz o pseudocódigo para o construtivo C1, sendo G um grafo ponderado, g e g' agrupamentos de vértices pertencentes a G, S o conjunto de vértices da solução e T a árvore geradora mínima que interliga os vértices presentes em S.

1. **para todo** $g \in G$ **faça**
2. $d_{\min} \leftarrow \infty$
3. **para todo** $v_i \in g$ **faça**
4. $d \leftarrow 0$
5. **para todo** $g' \in G \setminus g$ **faça**
6. $d' \leftarrow \infty$
7. **para todo** $v_j \in g'$ **faça**
8. **se** $d' > custo(v_i, v_j)$ **então**
9. $d' \leftarrow custo(v_i, v_j)$
10. **fim se**
11. **fim para**
12. $d \leftarrow d + d'$
13. **fim para**
14. **se** $d_{\min} > d$ **então**
15. $d_{\min} \leftarrow d$
16. $S[g] \leftarrow v_i$
17. **fim se**
18. **fim para**
19. **fim para**
20. $T \leftarrow AGM(S)$

Figura 6 – Pseudocódigo da heurística construtiva C1

Inicialmente, visita-se cada grupo $g \in G$ e para cada vértice $v_i \in g$ o algoritmo encontra o vértice $v_j \in g'$ mais próximo de v_i, sendo que, $g' \in G \setminus g$. Então d acumula os custos das arestas $e(v_i, v_j)$ selecionadas. Se o custo médio d for inferior ao menor custo encontrado até então, $d_{\min}$ é atualizado e v_i é inserido na solução S na posição g. Após percorrer todos os g grupos a solução estará completa e então executa-se o algoritmo de árvore geradora mínima para encontrar as arestas para compor a árvore T. Na Figura 7 pode-se observar uma representação gráfica do funcionamento do método descrito neste tópico.

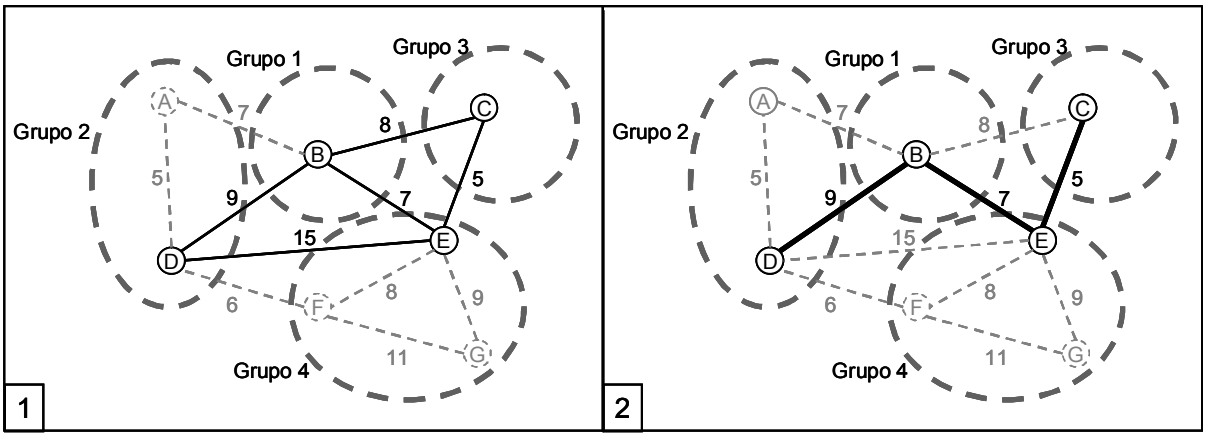


Figura 7 – Representação gráfica da execução do construtivo C1

No primeiro passo são selecionados os vértices que irão compor a solução S através do cálculo do custo médio d entre o vértice e o vizinho mais próximo de cada um dos outros grupos. Nesta e em todas as outras heurísticas desenvolvidas neste trabalho, sempre que dois vértices não possuem uma aresta entre eles, é atribuído um M para esta interligação, que é maior do que o custo de qualquer árvore geradora do grafo. Tem-se como exemplo o grupo dois, no qual para o vértice A $d = 7 + M + M$ e para o vértice D $d = 9 + M + 6$, sendo assim, D é selecionado por possuir um custo médio inferior a A. No segundo passo é executado o algoritmo de árvore geradora mínima $AGM(S)$ que seleciona os arcos e calcula o custo de T. O algoritmo utilizado para o calculo da árvore geradora mínima encontra-se descrito na seção 2.1.1. O custo final da árvore é de 21 ficando próximo ao custo da árvore ótima para o PAGMG neste grafo que é 19.

3.2 Construtivo C2

Este método construtivo foi desenvolvido conforme apresentado em Ferreira et al. (2006). Como primeiro passo, são selecionados os vértices para compor a solução, para isto, é selecionado um vértice de cada grupo que possua a menor distância média em relação a todos os vértices dos demais grupos. Na Figura 8 é apresentado o pseudocódigo para esta heurística.

1. **para todo** $g \in G$ **faça**
2. $d_{\min} \leftarrow \infty$
3. **para todo** $v_i \in g$ **faça**
4. $d \leftarrow 0$
5. **para todo** $g' \in G \setminus g$ **faça**
6. **para todo** $v_j \in g'$ **faça**
7. $d \leftarrow d + custo(v_i, v_j)$
8. **fim para**
9. **fim para**
10. **se** $d_{\min} > d$ **então**
11. $d_{\min} \leftarrow d$
12. $S[g] \leftarrow v_i$
13. **fim se**
14. **fim para**
15. **fim para**
16. $T \leftarrow AGM(S)$

Figura 8 – Pseudocódigo da heurística construtiva C2

Para selecionar os vértices participantes da solução o algoritmo percorre cada um dos grupos $g \in G$ e seleciona um vértice v_i que possua a menor distância média em relação aos vértices v_j pertencentes a $g' \in G \setminus g$. O vértice com a menor distância média é armazenado na solução S na posição g. Após selecionar um vértice v_i para cada $g \in G$ é executado o algoritmo de árvore geradora mínima $AGM(S)$. Então a árvore geradora mínima formada pelos vértices de S é armazenada em T. A Figura 9 traz a representação gráfica da execução do método em dois passos para uma instância com 7 vértices divididos em 4 grupos.

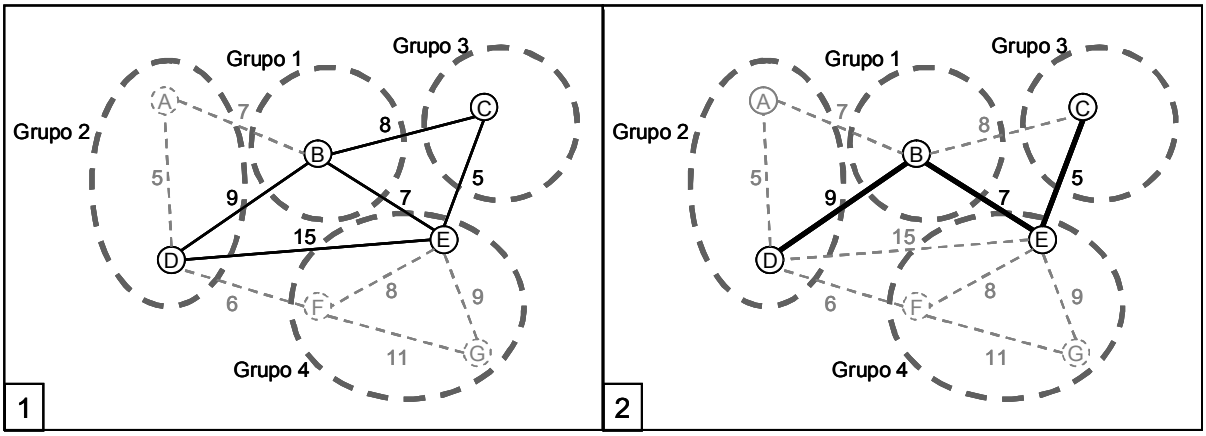


Figura 9 – Representação gráfica da execução do construtivo C2

Na etapa um são selecionados os vértices v_i para compor S, conseqüentemente os demais vértices bem como os arcos que se ligam a eles são deixados de lado pelo algoritmo. Tem-se como exemplo o grupo dois no qual para o vértice A $d = 7 + M + M + M + M$ e para o vértice D $d = 9 + M + 15 + 6 + M$, sendo assim, D é selecionado por possuir um custo médio inferior a A. No segundo passo é calculada a árvore geradora mínima utilizando os vértices e os arcos restantes. Para isto é executado o algoritmo descrito na seção 2.1.1. A árvore final possui um custo 21 que é muito próximo do custo ótimo do PAGMG neste grafo que é de 19.

3.3 Construtivo C3

Esta heurística foi proposta primeiramente no trabalho de Ferreira e Ochi (2007) diferencia-se em relação às demais pelo fato de que após um vértice ser selecionado, somente os arcos que conduzem a este vértice são considerados para o cálculo da distância média em relação a este grupo. Na Figura 10 apresenta-se o pseudocódigo para esta heurística. O algoritmo inicia com $G' \leftarrow \phi$ a cada ciclo um vértice v_i é selecionado e seu grupo g é inserido em G'. Assim para cada $g' \in G \setminus g$ se $g' \in G'$ então a distância média é calculada em relação à $S[g']$, caso contrário o cálculo é feito em relação a todos os vértices v_j de g'. Após a seleção dos vértices v_i é executado o algoritmo de árvore geradora mínima $AGM(S)$ para selecionar os arcos e calcular o custo da árvore T formada pelos vértices de S.

1. $G' \leftarrow \phi$
2. **para todo** $g \in G$ **faça**
3. $d_{\min} \leftarrow \infty$
4. **para todo** $v_i \in g$ **faça**
5. $d \leftarrow 0$
6. **para todo** $g' \in G \setminus g$ **faça**
7. **se** $g' \in G'$ **então**
8. $d \leftarrow d + custo(v_i, S[g'])$
9. **senão**
10. **para todo** $v_j \in g'$ **faça**
11. $d \leftarrow d + custo(v_i, v_j)$
12. **fim para**
13. **fim se**
14. **fim para**
15. **se** $d_{\min} > d$ **então**
16. $d_{\min} \leftarrow d$
17. $S[g] \leftarrow v_i$
18. $G' \leftarrow g$
19. **fim se**
20. **fim para**
21. **fim para**
22. $T \leftarrow AGM(S)$

Figura 10 – Pseudocódigo para o construtivo C3

Na Figura 11 pode-se ver a representação gráfica em dois passos para a execução do algoritmo aqui descrito.

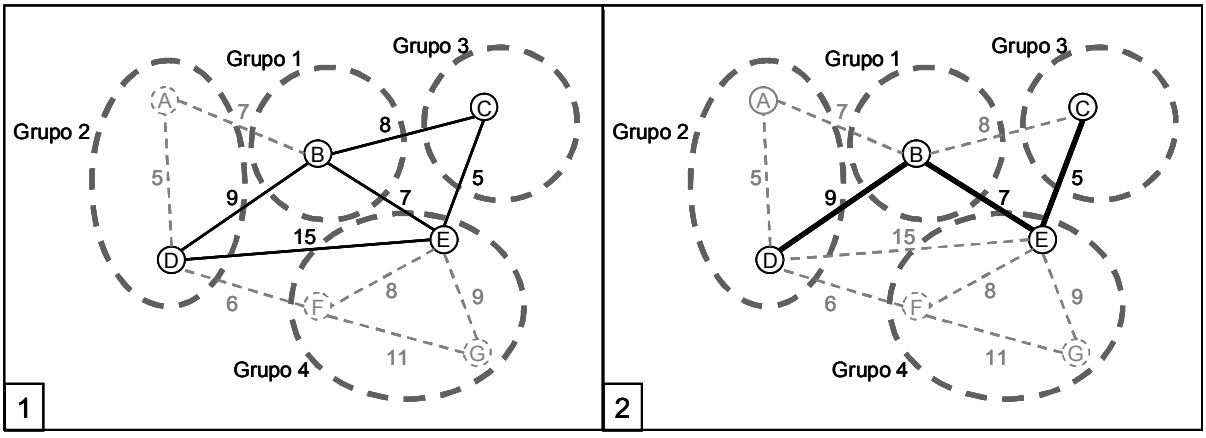


Figura 11 – Representação gráfica da execução do construtivo C3

No primeiro passo ocorre a seleção dos vértices. Esta seleção é feita percorrendo-se em ordem aleatória todos os grupos do grafo e selecionando um vértice de cada grupo para compor a solução. Como exemplo podemos supor uma ordem de varredura dos grupos na seqüência 3, 2, 1 e 4. No grupo 3 o vértice C é

selecionado por ser o único do grupo, seguindo para o grupo 2, onde o vértice D é selecionado por possuir um custo $d = 9 + M + 15 + 6 + M$ enquanto A possui um custo superior $d = 7 + M + M + M + M$. No grupo 1 o selecionado é B por ser único no grupo. No grupo 4 tem-se E com $d = 7 + 15 + 5$, F com $d = M + 6 + M$ e G com $d = M + M + M$, assim o vértice E é selecionado por apresentar o menor custo médio. Com a seleção dos vértices concluída agora o passo seguinte é calcular a árvore geradora mínima. Para isto utilizou-se o algoritmo descrito na seção 2.1.1. O custo da solução é de 21, muito próximo do custo ótimo para o PAGMG neste grafo que é 19.

Os três algoritmos construtivos descritos até aqui, são baseados na seleção dos vértices através do cálculo da distância média e posterior construção da árvore geradora mínima. Os três métodos, apesar de similares, apresentaram resultados bem diversificados durante os testes sendo que todos se mostraram melhores para determinadas instâncias do que outros, o que reforçou a idéia de utilizar mais de um método construtivo.

3.4 Construtivo C4

Esta heurística consiste em uma adaptação do algoritmo de Prim (1957), originalmente desenvolvido para o problema da árvore geradora mínima, para o PAGMG. Esta adaptação do algoritmo consiste em incluir apenas um vértice de cada grupo na solução. Para iniciar este algoritmo é necessário estabelecer um vértice inicial. Algumas maneiras para seleção do vértice inicial foram implementadas, como: iniciar pelo primeiro vértice, por um vértice aleatório e pelo vértice central, ou seja, o vértice com menor custo médio em relação a todos os demais vértices. Esta última proposição atingiu melhores resultados nos testes computacionais que as demais e foi a selecionada para a implementação.

A Figura 12 exibe o pseudocódigo para o algoritmo.

1. $d_{\min} \leftarrow \infty$
2. $V' \leftarrow \phi$
3. $G' \leftarrow \phi$
4. $E \leftarrow \phi$
5. $E' \leftarrow \phi$
6. **para todo** $v_i \in V$ **faça**
7. $d \leftarrow 0$
8. **para todo** $v_j \in V$ **faça**
9. **se** $i \neq j$ **então** $d \leftarrow d + custo(v_i, v_j)$
10. **fim para**
11. **se** $d < d_{\min}$ **então**
12. $d_{\min} \leftarrow d$
13. $v \leftarrow v_i$
14. $g \leftarrow grupo(v)$
15. **fim se**
16. **fim para**
17. $V' \leftarrow V' \cup v$
18. $G' \leftarrow G' \cup g$
19. $d \leftarrow 0$
20. **para todo** $g \in G$ **faça**
21. $d_{\min} \leftarrow \infty$
22. **para todo** $v_i \in V'$ **faça**
23. **para todo** $g' \in G \setminus G'$ **faça**
24. **para todo** $v_j \in g'$ **faça**
25. **se** $d_{\min} > custo(v_i, v_j)$ **então**
26. $d_{\min} \leftarrow custo(v_i, v_j)$
27. $v \leftarrow v_i$
28. $v' \leftarrow v_j$
29. **fim se**
30. **fim para**
31. **fim para**
32. **fim para**
33. $E \leftarrow E \cup v$
34. $E' \leftarrow E' \cup v'$
35. $V' \leftarrow V' \cup v'$
36. $G' \leftarrow G' \cup grupo(v')$
37. $d \leftarrow d + d_{\min}$
38. **fim para**

Figura 12 – Pseudocódigo para o construtivo C4

Depois de selecionado o vértice inicial v é inserido no conjunto V' e seu grupo g é inserido em G'. Em cada ciclo o algoritmo busca o vértice v_j pertencente

a $g' \in G \setminus G'$ com menor $custo(v_i, v_j)$, sendo que, $v_i \in V'$. Se o $custo(v_i, v_j)$ é menor que $d_{\min}$ então $d_{\min}$ recebe $custo(v_i, v_j)$, v recebe v_i e v' recebe v_j. Ao final de cada ciclo do algoritmo o vértice inicial v de cada arco da árvore é armazenado em E, o vértice final em E', o custo d é incrementado em $d_{\min}$, o vértice v' é incluído no conjunto V' e o $grupo(v')$ é incluído em G'. Ao final da execução do algoritmo os arcos que formam árvore geradora mínima estão armazenados em E e E', respectivamente origem e destino, os vértices da solução estão armazenados em V' e o custo em d.

A Figura 13 traz a representação gráfica da execução do algoritmo C4 em uma instância com 7 vértices divididos em 4 grupos. No primeiro passo é selecionado o vértice inicial E por possuir a menor distância média $d = M + 7 + 5 + 15 + 8 + 9$ em relação aos demais vértices do grafo, desta forma, sendo considerado o vértice central do grafo.

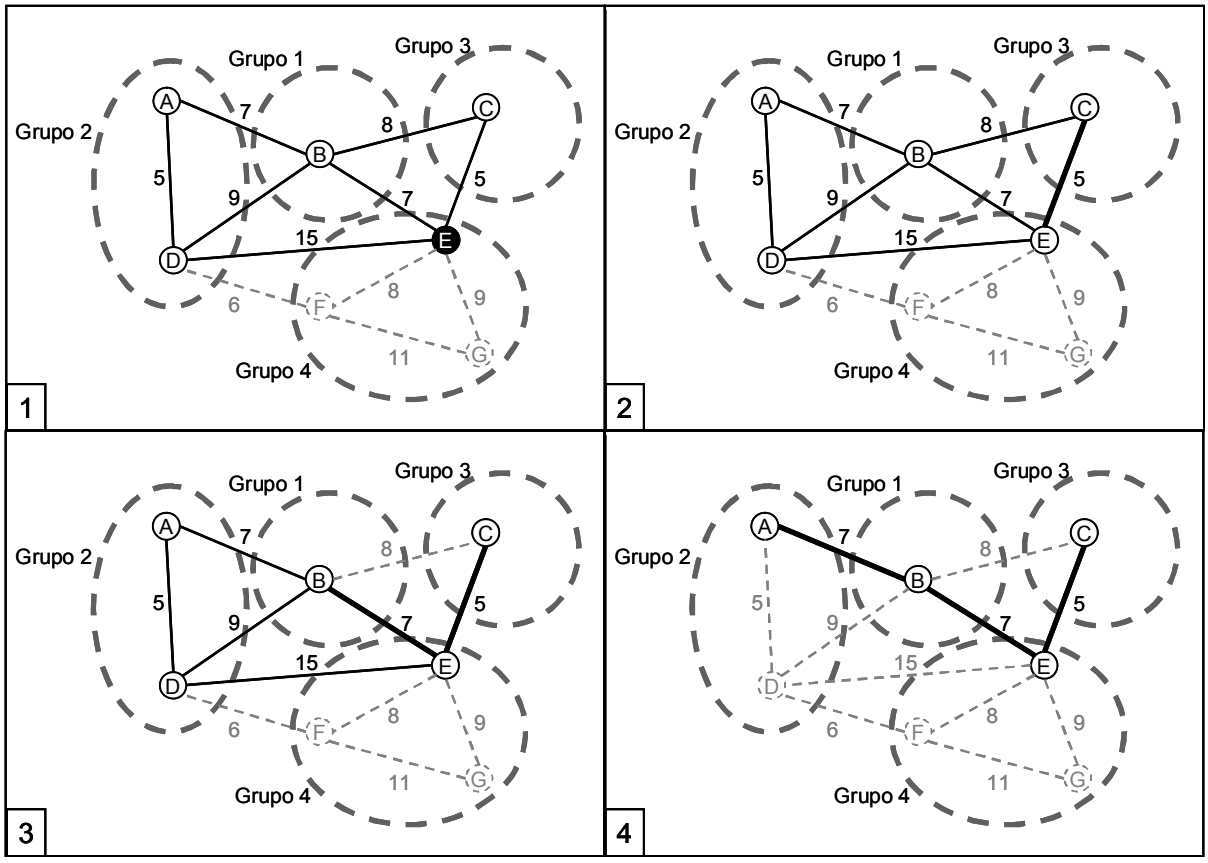


Figura 13 – Representação gráfica da execução do construtivo C4

No segundo passo escolhe-se a aresta de custo mínimo que interligue o vértice inicial a um grupo não visitado, as arestas disponíveis são BE=7, CE=5 e DE=15, sendo selecionada CE por possuir o menor custo. No terceiro passo é

selecionada uma aresta que interligue um dos vértices já visitados, C e E, a um dos vértices não visitados A, B ou D, a menor aresta BE=7 é selecionada. No quarto passo repete-se o procedimento executado no terceiro, estando disponíveis os vértices A e D, sendo A selecionado por apresentar a aresta de menor custo AB=7 para um vértice já inserido na árvore e finaliza-se a execução do algoritmo. A cada passo executado os demais vértices pertencentes ao grupo do vértice selecionado são excluídos dos passos seguintes do algoritmo. O custo final da solução é 19 sendo este o valor ótimo para esta instância.

Este método se mostrou eficaz para algumas instâncias embora, no geral nos testes realizados neste trabalho tenha um desempenho inferior aos métodos apresentados anteriormente. Outro aspecto interessante deste método é que após a aplicação das heurísticas de melhoramento ele supera os demais métodos para várias instâncias.

3.5 Construtivo C5

Esta heurística consiste em uma adaptação do algoritmo de Kruskal (1956) para o PAGMG. Adaptações como esta já foram utilizadas em outros trabalhos como Golden et al. (2005) e Ferreira et al. (2006). Para este construtivo é necessária a aplicação de um método de ordenação sob as arestas. Para tanto foram desenvolvidos alguns algoritmos clássicos de ordenação *selection sort*, *bubble sort*, *merge sort* e *quick sort*, sendo este último o que apresentou melhores resultados médios em relação ao tempo de ordenação para as instâncias analisadas. Depois de classificados em ordem crescente de custo, os arcos vão sendo inseridos um a um formando uma ou mais árvores que ao final do algoritmo se unem para formar uma única árvore geradora. Durante a inserção dos arcos é necessário tomar o cuidado de não visitar mais de um vértice do mesmo grupo e também que a inserção de um arco interligando dois vértices já visitados não ocasione a formação de um ciclo. A Figura 14 traz o pseudocódigo para o construtivo C5.

1. $ordenar(E)$
2. $T \leftarrow \phi$
3. $c \leftarrow K$
4. **para todo** $g \in G$ **faça**
5. $\quad \gamma[g] \leftarrow 0$
6. **fim para**
7. **enquanto** $c > 1$ **faça**
8. $\quad e(u,v) \leftarrow proximaAresta(E)$
9. $\quad$ **se** $!geraCiclos(e)$ **então**
10. $\quad\quad$ **se** $(\gamma[g_u] = u$ **ou** $\gamma[g_u] = 0)$ **e** $(\gamma[g_v] = v$ **ou** $\gamma[g_v] = 0)$ **então**
11. $\quad\quad\quad T \leftarrow T \cup e$
12. $\quad\quad\quad c \leftarrow c - 1$
13. $\quad\quad\quad$ **se** $\gamma[g_u] = 0$ **então**
14. $\quad\quad\quad\quad \gamma[g_u] \leftarrow u$
15. $\quad\quad\quad$ **fim se**
16. $\quad\quad\quad$ **se** $\gamma[g_v] = 0$ **então**
17. $\quad\quad\quad\quad \gamma[g_v] \leftarrow v$
18. $\quad\quad\quad$ **fim se**
19. $\quad\quad$ **fim se**
20. $\quad$ **fim se**
21. **fim enquanto**

Figura 14 - Pseudocódigo para o construtivo C5

As arestas E do grafo são dispostas em ordem crescente. A cada passo uma aresta $e(u,v) \in E$ é selecionada. Caso a inserção de e não gere ciclo e no grupo a que u pertence ainda não tenha sido selecionado nenhum vértice ou u seja o vértice selecionado no grupo para compor a árvore e a mesma coisa acontecendo com v, então a aresta e é inserida na árvore. O algoritmo termina quando forem inseridas $K-1$ arestas na árvore interligando todos os g grupos.

A Figura 15 traz uma ilustração da execução deste construtivo para uma instância com 7 vértices divididos em 4 grupos. A execução para esta instância possui quatro passos. No primeiro é feita a ordenação das arestas e neste momento as arestas interligando vértices do mesmo grupo são descartadas. Resultando numa lista ordenada contendo respectivamente as arestas CE=5, DF=6, AB=7, BE=7, BC=8, BD=9 e DE=15. No segundo passo é selecionada a primeira aresta da lista interligando os vértices C e E a qual é inserida na árvore, com isto sendo descartada a segunda aresta da lista por não interligar o grupo 4 através do vértice E. No terceiro passo é selecionada a aresta AB a qual forma uma segunda árvore. No

quarto e último passo a aresta BE é inserida interligando as duas árvores para formar a árvore geradora mínima entre os vértices selecionados com custo 19.

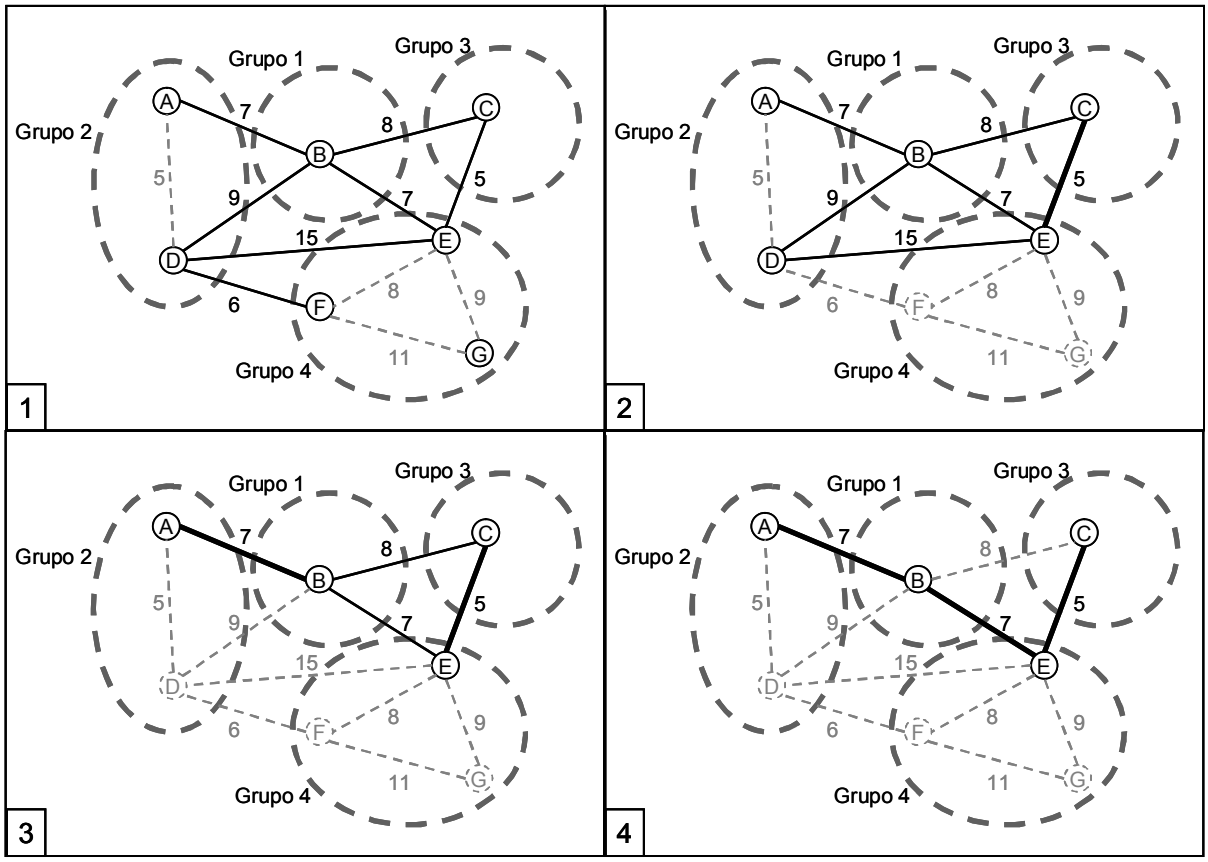


Figura 15 – Representação gráfica da execução do construtivo C5

Este método construtivo, apesar de retornar na maioria das vezes o melhor resultado entre os cinco desenvolvidos apresentou resultados geralmente inferiores aos demais após a aplicação dos métodos de melhoramento, descritos a seguir nas seções 3.6 e 3.7. Este fato se deve aos três primeiros métodos possuírem uma característica de buscar selecionar os vértices mais adequados para uma solução de boa qualidade para depois construírem a árvore, ao contrário de C4 e C5. Sendo assim é bem provável que na maioria das vezes, para as buscas locais desenvolvidas, que funcionam através do princípio de substituição de vértices destas soluções iniciais, é mais fácil chegar a soluções de melhor qualidade partindo de uma solução com vértices pré-selecionados.

3.6 Busca Local BL1

Esta heurística de busca local procura encontrar melhorias envolvendo a troca de um vértice da solução por outro do mesmo grupo. A cada passo da heurística é feita uma varredura em toda a vizinhança da solução que envolva a troca de um vértice. A cada melhoria encontrada a solução corrente é atualizada. A busca é interrompida quando não é mais possível encontrar nenhuma solução vizinha com valor de função objetivo inferior ao da solução corrente. Estratégias similares de busca em vizinhança foram adotadas nos trabalho de Golden et al. (2005) e Ferreira et al. (2006).

Por tratar-se de uma heurística de busca local, a mesma exige como entrada uma solução inicial S a partir da qual a busca terá início. A solução inicial S consiste em uma lista contendo um vértice de cada grupo do grafo, que podem ser obtidos através da aplicação de um método construtivo ou escolhidos aleatoriamente. Na Figura 16 é possível visualizar o pseudocódigo para a heurística de busca local BL1.

```
1.  T ← AGM(S)
2.  c_min ← custo(T)
3.  m ← 1
4.  enquanto m = 1 faça
5.      m ← 0
6.      para todo g ∈ G faça
7.          para todo v ∈ g faça
8.              a ← S[g]
9.              S[g] ← v
10.             T' ← AGM(S)
11.             c ← custo(T')
12.             se c < c_min então
13.                 m ← 1
14.                 T ← T'
15.                 c_min ← c
16.             senão
17.                 S[g] ← a
18.             fim se
19.         fim para
20.     fim para
21. fim enquanto
```

Figura 16 - Pseudocódigo para a busca local BL1

Inicialmente a árvore geradora mínima de S é armazenada em T e o custo da mesma em $c_{\min}$. Então a variável de controle m é inicializada e a busca tem início. A cada passo da busca um vértice de S é substituído por outro do mesmo grupo, caso a nova árvore T' possua um custo c inferior a $c_{\min}$ então T é atualizada para T' e c para $c_{\min}$. Caso contrário, a substituição é desfeita retornando o vértice substituído a S no lugar do vértice substituto.

A Figura 17 apresenta uma representação gráfica para a heurística de busca local BL1. No primeiro passo a heurística recebe uma solução inicial com custo 23 e então estabelece uma ordem aleatória para visitar os grupos (Ex.: 3, 4, 1 e 2). No grupo 3 não há vértices disponíveis para trocas, sendo assim, o algoritmo segue para o grupo 4. Neste grupo são possíveis duas trocas, a primeira é substituir o vértice F pelo vértice E gerando uma nova solução com custo 21 e a segunda é substituir o vértice F por G gerando uma solução com custo $M+17$. O segundo quadro ilustra a troca do vértice F do grupo 4 pelo vértice E do mesmo grupo proporcionando a primeira melhoria na solução fazendo seu custo cair de 23 para 21. Este processo se repete para os grupos 1 (no qual não há trocas possíveis) e 2 no qual a substituição do vértice D pelo vértice A faz o custo da solução cair para 19. No terceiro quadro pode-se visualizar a solução final encontrada pela heurística de busca local BL1. O processo continua até que todos os grupos tenham sido visitados sem encontrar novas melhorias.

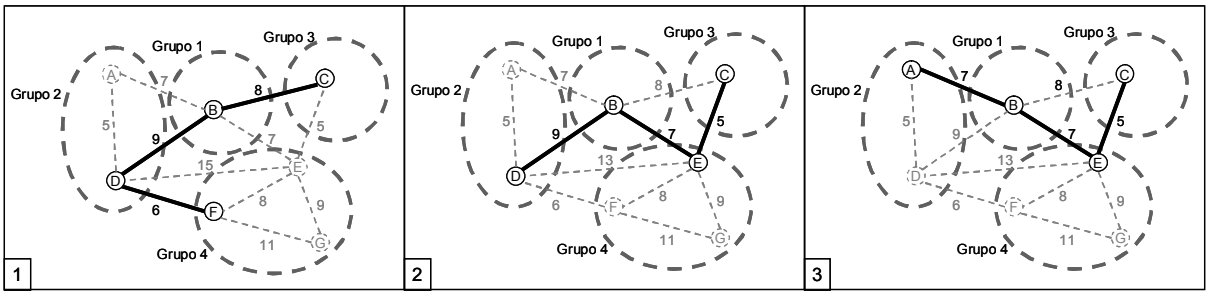


Figura 17 – Representação gráfica da execução da busca local BL1

3.7 Busca Local BL2

A busca local BL2 se diferencia da heurística BL1 por executar somente o melhor movimento envolvendo a troca de um vértice por outro de mesmo grupo a

cada iteração da busca. Sendo assim, a busca local procura a cada passo a solução que cause o maior ganho em termos do valor de função objetivo, tendo encontrado tal solução esta passa a ser a nova solução corrente e uma nova iteração é iniciada. A busca é interrompida quando não houver na vizinhança soluções melhores do que a corrente. A Figura 18 apresenta o pseudocódigo para a heurística de busca local BL2.

```
1.  T ← AGM(S)
2.  c_min ← custo(T)
3.  m ← 1
4.  enquanto m = 1 faça
5.      m ← 0
6.      para todo g ∈ G faça
7.          para todo v ∈ g faça
8.              a ← S[g]
9.              S[g] ← v
10.             T' ← AGM(S)
11.             c ← custo(T')
12.             se c < c_min então
13.                 m ← 1
14.                 v' ← v
15.                 g' ← g
16.                 c_min ← c
17.             fim se
18.             S[g] ← a
19.         fim para
20.     fim para
21.     se m = 1 então
22.         S[g'] ← v'
23.         T ← AGM(S)
24.     fim se
25. fim enquanto
```

Figura 18 – Pseudocódigo para a busca local BL2

A Figura 19 apresenta a representação gráfica da execução da busca local BL2 em uma instância com 7 vértices divididos em 4 grupos, onde, são realizadas duas iterações da busca até que a mesma seja interrompida ao encontrar um ótimo local.

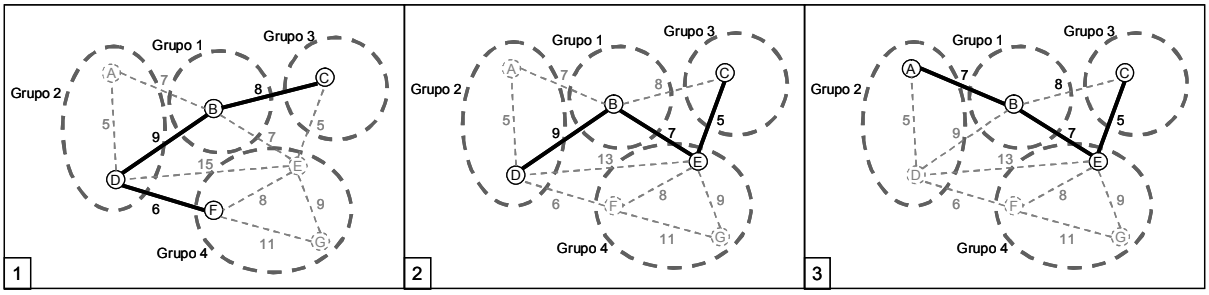


Figura 19 - Representação gráfica da execução da busca local BL2

Na vizinhança da solução inicial $S[BDCF] = 23$ exibida no quadro 1 da Figura 19 tem-se as soluções $S[BDCE] = 21$, $S[BDCG] = M + 17$ e $S[BACF] = M + 15$ das quais a primeira apresenta o menor custo entre as três sendo este inferior ao custo da solução corrente, neste caso a solução é atualizada. Partindo então da nova solução corrente que é exibida no quadro 2 da Figura 19 tem-se como soluções vizinhas $S[BDCF] = 23$, $S[BDCG] = M + 17$ e $S[BACE] = 19$ sendo esta última a possuir o menor custo na vizinhança. Como o custo desta solução é inferior ao da solução corrente ela é atualizada. Verificando a vizinhança da solução exibida no quadro 3 é possível verificar que não há mais movimentos de melhoria disponíveis, sendo encerrada portanto a busca por ter atingido um ótimo local.

Para a maioria das instâncias BL2 produziu melhores resultados do que BL1, por outro lado na grande maioria das vezes BL2 consumiu um tempo de processamento substancialmente maior do que BL1. Foi definido como estratégia de busca executar uma vez cada um dos construtivos e aplicar sobre cada um deles as duas buscas locais propostas. Ao final têm-se dez soluções iniciais geradas e a melhor delas é selecionada para iniciar a metaheurística.

3.8 Reconexão de Caminhos RC

Esta técnica é detalhada no artigo de Glover et al. (2000), no qual os autores descrevem como fazer a sua implementação e possíveis aplicações. A idéia central da reconexão de caminhos é partir de uma solução origem em direção a uma de destino explorando um caminho de soluções intermediárias, dentre as quais possivelmente poderá haver uma solução melhor do que ambas.

Neste trabalho a técnica de reconexão de caminhos é utilizada em conjunto com a busca tabu, com vistas a melhorar o desempenho desta metaheurística na solução do PAGMG. No exemplo da Figura 20, a partir da comparação entre a solução de origem S_o e a solução de destino S_d são identificadas três diferenças entre elas, as quais encontram-se em destaque na figura. Sendo assim, um a um são substituídos estes três vértices de S_d que diferem dos encontrados em S_o gerando cada um uma nova solução intermediária S_i. Entre as três soluções intermediárias a melhor delas é selecionada e passa a ser a nova solução corrente S'. Restam então dois vértices que diferem entre S' e S_d, através da substituição destes vértices em S' são geradas duas novas S_i. A cada passo todas as S_i são avaliadas e a S_i de menor custo é utilizada como nova S'. Se em algum passo S' possuir custo inferior a melhor solução encontrada até o momento S, então, S é atualizada para S'. O número total de S_i geradas é igual a $dif^2 - \left(\frac{dif^2 - dif}{2} \right) - 1$, onde dif se refere ao número de vértices que diferem entre S_o e S_d. A cada passo a geração e seleção se repete até atingir S_d.

Na Figura 20 pode-se visualizar a representação gráfica da execução do método de reconexão de caminhos, as S' são indicadas por setas pretas, as S_i por setas cinza e S encontra-se em destaque. Partindo de S_o são geradas três soluções intermediárias (S_i) S_1, S_2 e S_3 com custos de 22, 17 e 18, respectivamente. Pelo fato de S_2 possuir o menor custo entre as três S_i está é selecionada para dar continuidade ao método tornando-se a nova S'. Na segunda iteração do método são geradas duas soluções intermediárias S_4 e S_5 com custos de 14 e 20, respectivamente. Sendo S_4 a S_i de menor custo ela passa a ser a nova S'. A cada atualização em S' seu custo é comparado com o custo de S, caso S' possua um custo inferior a S esta é atualizada.

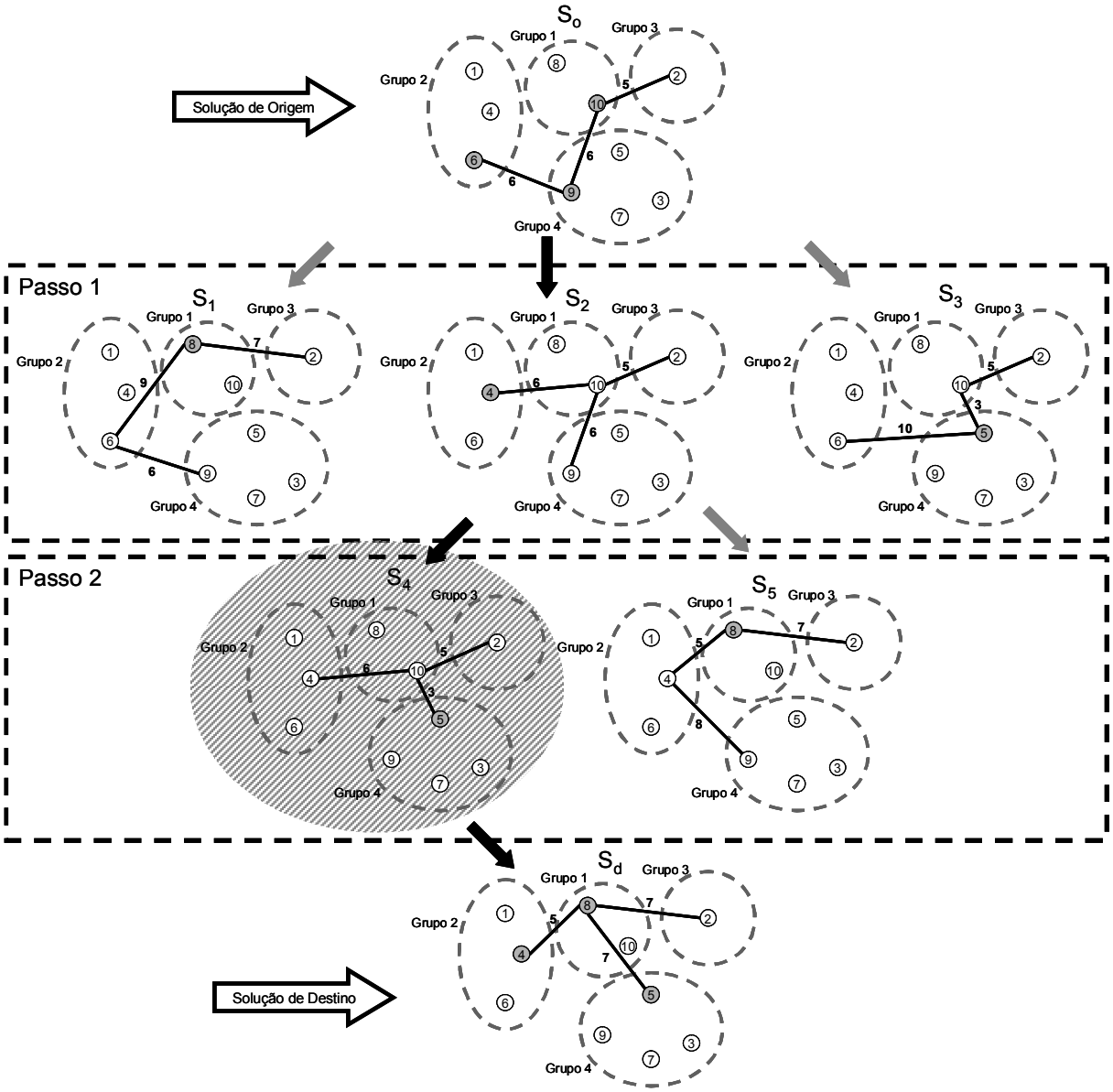


Figura 20 – Representação gráfica da execução da reconexão de caminhos

Na metaheurística proposta a reconexão de caminhos é executada a cada iteração entre a solução corrente e a solução do conjunto de elite que maximize dif, caso a solução corrente possua um custo inferior a $cMax$. Dentre as duas S_o recebe a de menor custo e S_d a outra. O valor de $cMax$ fica contido no intervalo $[cMin + 10\%, cMin + 100\%]$, sendo $cMin$ o custo da solução incumbente, ou seja, a melhor solução encontrada até o momento pela metaheurística. A variação de $cMax$ dentro deste intervalo se dá de acordo com o número de iterações sem atualização do conjunto de soluções de elite, ou seja, quanto mais difícil atualizar o conjunto de elite, maior será o valor de $cMax$.

3.9 Busca Local Iterativa BLI

Este procedimento foi originalmente proposto e brevemente descrito em Lourenço et al. (2001), posteriormente, o mesmo foi mais bem detalhado em Lourenço et al. (2002). Os autores deixam claro nos trabalhos anteriormente citados que quatro são os fatores determinantes para que a técnica seja bem sucedida. São eles: geração da solução inicial, algoritmo de busca local, procedimento de perturbação e função de aceitação. A escolha correta dos algoritmos para cada um destes papéis é de fundamental importância para atingir as metas desejadas. Na Figura 21 pode-se visualizar o pseudocódigo do algoritmo de busca local iterativa (BLI) disponível em Lourenço et al. (2001) e Lourenço et al. (2002).

1. $S_0 \leftarrow GerarSolucaoInicial(\ )$
2. $S^* \leftarrow BuscaLocal(S_0)$
3. **repita**
4. $\quad S' \leftarrow Perturbacao(S^*, historico)$
5. $\quad S^{*\prime} \leftarrow BuscaLocal(S')$
6. $\quad S^* \leftarrow CriterioDeAceitacao(s^*, s^{*\prime}, historico)$
7. **até que** $CondicaoAtingida$

Figura 21 – Pseudocódigo para Busca Local Iterativa Adaptado de Lourenço et al. (2001)

Neste trabalho a busca local iterativa foi utilizada como uma ferramenta adicional para ampliar o potencial da metaheurística. A técnica é acionada quando é constatada uma estagnação no processo de melhoria da solução incumbente, ou seja, quando a busca fica presa a um ótimo local por um período equivalente a 100 iterações então aplica-se a BLI sob o conjunto de soluções de elite. Seguindo os passos do pseudocódigo apresentado na Figura 21, utiliza-se como solução inicial cada uma das soluções do conjunto de elite. Para cada uma delas é executado um procedimento de busca local. Posteriormente inicia-se o processo sucessivo de perturbação, busca e aceitação.

A perturbação das soluções é feita da seguinte maneira, a cada iteração um grupo é selecionado aleatoriamente. Para cada um dos grupos que estiverem a uma distância menor do que $dMax = dAvg \dfrac{dAvg}{\log_{10}(V+G)}$ do grupo selecionado o vértice daquele grupo é substituído na solução por outro do mesmo grupo escolhido aleatoriamente. Feito isto o passo seguinte é aplicar sobre esta solução perturbada o

procedimento de busca local. O procedimento de busca local escolhido para atuar na busca local iterativa foi o BL1, anteriormente descrito na seção 3.6. Como critério de aceitação foi definido que se a solução gerada tivesse um valor de função objetivo melhor do que a solução inicial, esta seria atualizada e o processo seria reiniciado.

A melhor solução gerada durante todo o processo de BLI é armazenada e ao final comparada com a solução incumbente da metaheurística, caso a nova solução possua um custo menor, a solução incumbente é atualizada.

3.10 Busca Tabu BT

Devido a metaheurística busca tabu ter apresentado os melhores resultados nos testes preliminares em relação ao algoritmo genético, optou-se por dar continuidade ao seu desenvolvimento, selecionando-a como metaheurística para solução do PAGMG neste trabalho. Para gerar a solução inicial da busca tabu são executados os cinco algoritmos construtivos descritos anteriormente. Para cada uma das cinco soluções geradas são aplicadas as duas heurísticas de melhoramento. Este processo gera um total de dez soluções das quais a melhor delas é escolhida como solução inicial da metaheurística. Este processo construtivo economiza bastante tempo de processamento da metaheurística, uma vez que a busca tabu inicia a partir de uma solução de qualidade razoável. Na Figura 22 é possível visualizar o diagrama da metaheurística desenvolvida neste trabalho.

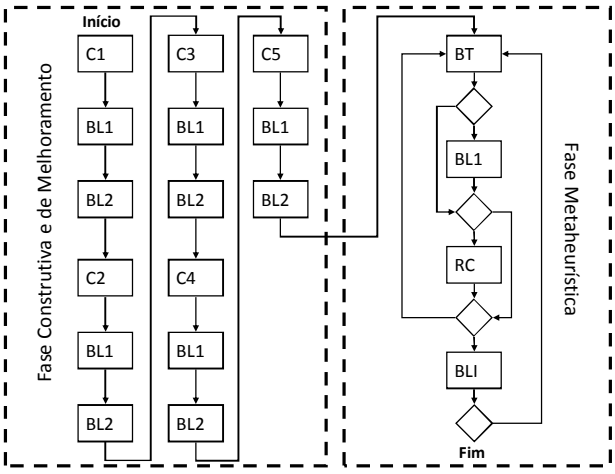


Figura 22 – Diagrama da Metaheurística desenvolvida

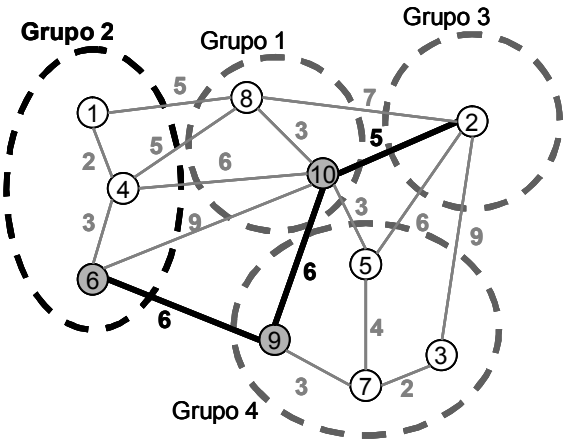
Nas subseções seguintes procurou-se descrever de forma mais detalhada a metaheurística desenvolvida neste trabalho. Cada tópico retrata um dos aspectos relevantes referentes à metaheurística.

3.10.1 Estratégia de Restrição de Vizinhança

No algoritmo de busca tabu utilizou-se quatro critérios de vizinhança N_0, N_1, N_2 e N_3. Em N_0 é selecionada a melhor troca de um vértice por outro de mesmo grupo. Na estratégia de vizinhança N_1 um grupo é selecionado e o vértice deste grupo que está compondo a árvore é substituído por outro vértice do mesmo grupo que possua o menor custo. Em N_2, são testadas todas as trocas envolvendo vértices de 2 grupos selecionados, a troca que proporcionar a solução de menor custo é executada. Em N_3, são testadas todas as trocas envolvendo vértices de 3 grupos selecionados, a troca que proporcionar a solução de menor custo é executada. A Figura 23 traz um exemplo das estratégias de vizinhança N_0, N_1, N_2 e N_3.

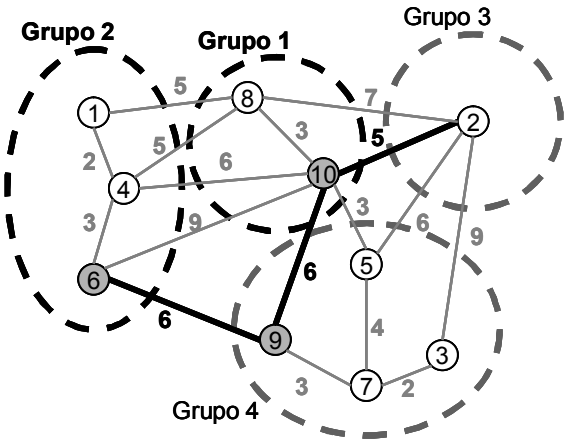
Cada uma das quatro estratégias de vizinhança, N_0, N_1, N_2 e N_3, é utilizada até que um número determinado número de iterações seja executado sem que haja melhoria na solução incumbente, este número de iterações é ajustado pelos parâmetros t_0, t_1, t_2 e t_3. A busca inicia pela vizinhança N_1 segue pela N_2, N_3 e N_0, e a seqüência se repete até que a heurística seja encerrada. Para chegar nesta definição de vizinhança várias estratégias foram analisadas como utilizar somente N_0, N_1, N_2 e N_3, e outras combinações, porém os melhores resultados se mostraram quando as quatro estratégias de vizinhança foram aplicadas em conjunto.

Depois de vários testes buscando definir os valores mais adequados para os parâmetros t_0, t_1, t_2 e t_3 os mesmos foram ajustados em 10, 200, 100 e 50, respectivamente, atribuindo desta maneira um número maior de iterações para as vizinhanças menores. As estratégias de vizinhança N_1, N_2 e N_3 foram também utilizadas por Oncan et al. (2007) e a estratégia N_0 por Golden et al. (2005).



Vizinhança N_1 para o Grupo 2

S_1 [10,1,2,9]=M+11 S_2 [10,4,2,9]=17

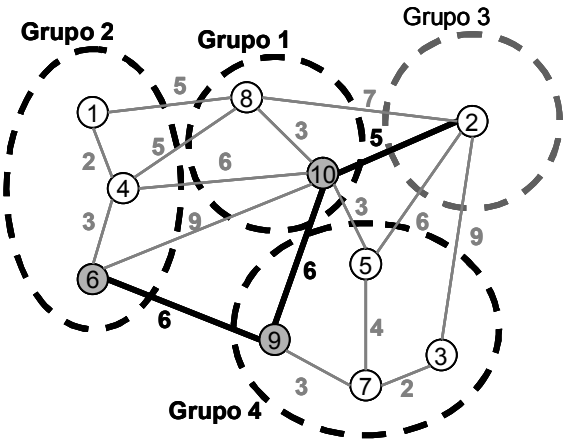


Vizinhança N_2 para os Grupos 1 e 2

S_1 [10,1,2,9]=M+11 S_3 [8,1,2,9]=M+12

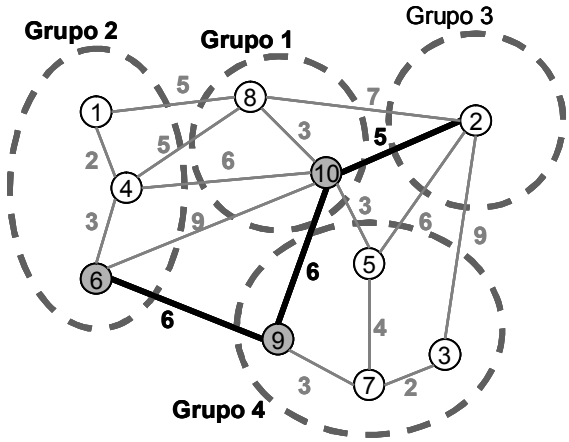
S_2 [10,4,2,9]=17 S_4 [8,4,2,9]=M+12

S_5 [8,6,2,9]=M+13



Vizinhança N_3 para os Grupos 1, 2 e 4

S_1 [10,1,2,9]=M+11	S_3 [8,1,2,9]=M+12
S_2 [10,4,2,9]=17	S_4 [8,4,2,9]=M+12
	S_5 [8,6,2,9]=M+13
S_6 [10,1,2,3]=M+14	S_9 [8,1,2,3]=21
S_7 [10,4,2,3]=20	S_{10} [8,4,2,3]=21
S_8 [10,6,2,3]=23	S_{11} [8,6,2,3]=M+16
S_{12} [10,1,2,5]=M+8	S_{15} [8,1,2,5]=18
S_{13} [10,4,2,5]=14	S_{16} [8,4,2,5]=18
S_{14} [10,6,2,5]=17	S_{17} [8,6,2,5]=M+13
S_{18} [10,1,2,7]=2M+5	S_{21} [8,1,2,7]=M+12
S_{19} [10,4,2,7]=M+11	S_{22} [8,4,2,7]=M+12
S_{20} [10,6,2,7]=M+14	S_{23} [8,6,2,7]=2M+7



Vizinhança N_0

S_1 [10,1,2,9]=M+11

S_2 [10,4,2,9]=17

S_5 [8,6,2,9]=M+13

S_6 [10,6,2,5]=17

S_7 [10,6,2,7]=M+14

S_8 [10,6,2,3]=23

Figura 23 – Exemplo de Restrição de Vizinhança

3.10.2 Seleção dos Grupos de Busca

A seleção dos grupos para efetuar os procedimentos de busca local em vizinhança é feita sorteando-se aleatoriamente os grupos durante 50 iterações. Feito

isto, a escolha dos grupos passa a ser feita de maneira aleatória entre os grupos que foram sorteados um número menor de vezes que o grupo mais sorteado. Este procedimento visa buscar um maior equilíbrio na busca fazendo com que não haja uma discrepância muito grande entre o grupo mais e o menos pesquisado. O procedimento cessa quando todos os grupos tiverem sido escolhidos o mesmo número de vezes. Ao atingir o equilíbrio na freqüência de sorteio dos grupos o método de seleção passa a ser totalmente aleatório novamente por mais 50 iterações. Foram realizados testes empíricos com valores múltiplos de 10, no intervalo [10,100] para este parâmetro, o valor 50 apresentou os melhores resultados.

3.10.3 Lista Tabu

Durante o processo de busca todo vértice retirado da solução corrente é inserido na lista tabu, desta forma não podendo fazer parte da solução por algumas iterações, procurando com isto evitar o fenômeno de ciclagem da busca.

A lista tabu foi implementada na forma de um vetor no qual na posição referente a cada vértice foi armazenada a iteração a partir da qual aquele vértice encontra-se disponível para ser inserido na solução. Desta forma, inicialmente todas as posições do vetor recebem o valor 0 para que todos os vértices estejam disponíveis, a medida que a busca vai transcorrendo os vértices que vão sendo substituídos ficam indisponíveis por algumas iterações.

Para definir o número de iterações em que um vértice ficaria indisponível para ser inserido na solução, inicialmente optou-se por uma quantidade fixa de iterações e foram feitos testes com valores entre 5 e 10. Posteriormente, após novos testes constatou-se que se o número de iterações fosse definido aleatoriamente dentro de um intervalo restrito, $[5,10]$, é possível obter melhores resultados em média. Porém, foi constatado ainda que de acordo com o tamanho da instância é necessário aumentar levemente o “tempo tabu” para que melhores resultados sejam atingidos. Assim foi definida uma estratégia para que a quantidade de iterações variasse de acordo com o tamanho da instância do problema, e então definiu-se o intervalo $[4\log_{10} V - 3{,}4\log_{10} V + 3]$, no qual V é o número de vértices da instância, para o

número iterações de permanência de um vértice na lista tabu. Como os limites do intervalo definido pelas equações podem resultar em números com casas decimais estes são arredondados para inteiros. Dentro deste intervalo é gerado aleatoriamente um número inteiro uniformemente distribuído.

Para que boas soluções não fossem perdidas pelo motivo de um vértice estar definido como tabu, foi necessária a implementação de um vetor que armazenasse o valor da melhor solução obtida para cada vértice do grafo, logo, se para qualquer um dos vértices presentes na solução candidata o melhor valor de solução contendo aquele vértice fosse superior ao atual, a condição tabu deste vértice é retirada, configurando-se esse o critério de aspiração.

3.10.4 Penalidades

Durante o desenvolvimento das metaheurísticas propostas para esta dissertação foi constatada a necessidade de implementação de mecanismos para diversificação da busca. Com base neste aspecto, optou-se inicialmente pelo uso de estruturas de memórias de longo prazo capazes de determinar a freqüência de participação de cada vértice nas soluções geradas durante a execução do algoritmo. Através desta medida de freqüência foram criadas e testadas várias formas de diversificação e também de intensificação da busca. A partir do estudo do artigo de Öncan et al. (2007) foi possível perceber que o mecanismo para penalizar soluções que levam a uma piora no valor da solução corrente poderia ser aplicado com bastante eficiência no método aqui proposto. Neste ponto, optou-se então pela implementação do mecanismo de penalidades de Öncan et al. (2007) que possibilitou a ocorrência de soluções mais diversificadas com isto ampliando inclusive a eficácia do mecanismo de reconexão de caminhos, através do incremento no número de soluções intermediárias devido a uma maior diversificação entre solução de origem e solução de destino.

O método descrito em Oncan et al (2007) penaliza soluções que conduzem a valores de função objetivo piores do que os da solução corrente S através da equação $p(\bar{s}) = c(\bar{s}) + \phi c(s)\sqrt{kn}\sum_{v \in \bar{s} \setminus s} freq_v / iter$, para as demais soluções candidatas $\bar{S}$

o custo não é penalizado, sendo $p(\bar{s}) = c(\bar{s})$. Nestas equações $p(\bar{s})$ representa o custo penalizado, $c(\bar{s})$ o custo da solução candidata, ϕ o fator de diversificação, $c(s)$ o custo da solução corrente, k o número de grupos do grafo, n o número de vértices do grafo, $freq$ é o vetor que armazena o número de inserções de um vértice na solução e $iter$ o número de iterações realizadas pela metaheurística.

3.10.5 Conjunto de Elite CE

O conjunto de elite, CE, consiste em um grupo formado pelas melhores soluções encontradas pela busca tabu. O tamanho de CE é definido através do parâmetro CE_{Tam}, para o qual, foram testados valores entre 1 e 10, tendo o valor 5 apresentado os melhores resultados. Cada solução gerada durante a busca é comparada às do conjunto de elite, se a solução atual possuir um custo inferior ao da pior de CE, ou o número de soluções incluídas em CE for inferior a CE_{Tam}, e a solução atual não é uma duplicata de alguma das soluções de CE então esta passa a fazer parte de CE. Durante a inserção de uma nova solução em CE se o tamanho deste exceder CE_{Tam}, então a pior entre as soluções de CE é excluída.

Como critério para renovação de CE foi estabelecido que quando CE permanece-se por 100 iterações da busca tabu sem ser atualizado, então é realizado um procedimento de reinicialização de CE, no qual todas as soluções de CE são excluídas, com exceção da incumbente.

3.10.6 Critérios de Parada

Como critérios de parada utilizou-se três estratégias, busca de alvo, número limite de iterações ou tempo limite de processamento. Para as instâncias do grupo 1, as quais possuem valor ótimo já calculado em Feremans et al. (2001), o critério utilizado foi o da busca por alvo. Já nas instâncias do grupo 2 o critério estabelecido foi o tempo limite de execução de 3 horas. Durante o desenvolvimento das

heurísticas também foi testado o critério de número limite de iterações, optou-se por não utilizar este critério na versão final devido a dificuldade de encontrar o valor mais adequado para este parâmetro.

3.11 Adaptação para o PM-PAGMG

Para a solução do PM-PAGMG utilizaram-se as mesmas heurísticas já descritas anteriormente, diferenciando-se apenas pela utilização de uma heurística adicional denominada aqui de heurística de inserção. Tal heurística tem por finalidade aprimorar uma solução baseada em um único vértice por grupo, através da inserção de mais vértices na solução.

Tendo por base uma solução S que é formada pelos vértices V' a heurística procura inserir em S o vértice $V \setminus V'$ que mais diminua o custo de S. Assim a cada passo a heurística pesquisa toda a vizinhança $V \setminus V'$ e insere o vértice que ocasione a maior queda no custo da solução. A heurística é finalizada assim que não houver mais vértices a serem incluídos na solução que façam o custo da mesma diminuir.

Durante a modelagem deste problema optou-se por transformar as instâncias geradas por Dror et. al (2000) em grafos totalmente conexos inserindo os arcos ausentes com custos superiores a soma dos custos de todos os demais arcos do grafo.

Para acelerar o processo e filtrar soluções que dificilmente levariam a ótimos locais foram excluídas do processo árvores que incluíssem mais de quatro arcos que não existem no grafo original, além de soluções que possuam custo superior a 1,5 vezes o custo da solução incumbente.

4 TESTES COMPUTACIONAIS

Nos testes computacionais utilizou-se um computador tipo PC (*Personal Computer*) equipado com processador AMD Athlon 64 X2 2800MHz e 2GB de memória executando o sistema operacional GNU/Linux 2.6.18. As heurísticas propostas nesta dissertação foram todas desenvolvidas em linguagem java utilizando o JDK 1.6_u3 e a plataforma de desenvolvimento NetBeans 5.5.1. Os resultados obtidos foram comparados aos trabalhos mais recentes de Golden et al. (2005) e Öncan et al. (2007) os quais apresentam os melhores resultados para as instâncias testadas.

Para a realização dos testes computacionais dividiu-se as instâncias do PAGMG em dois grupos. Um formado por instâncias com o valor ótimo já conhecido e outro com instâncias onde o valor ótimo ainda não é conhecido. Para todas as instâncias os resultados obtidos foram calculados através da média aritmética entre os resultados de cinco execuções consecutivas da metaheurística. Todas as instâncias do PAGMG testadas utilizam as técnicas de agrupamento de vértices em grafos *center clustering* e *grid clustering* (veja Fischetti et al. (1995)).

4.1 Resultados com Instâncias do Grupo 1

O primeiro grupo é composto por 150 instâncias com tamanhos entre 47 e 226 vértices, sendo que, Feremans et al. (2001) apresenta os valores ótimos para todas estas instâncias. Os resultados obtidos com as instâncias do primeiro grupo foram comparados com as heurísticas GA e LS desenvolvidas por Golden et al. (2005) que apresenta os melhores resultados em termos de qualidade de solução e tempo computacional encontrados até o momento na literatura para as instâncias em questão. É importante salientar que os *hardwares* utilizados nos trabalhos são bastante heterogêneos, sendo utilizado por Golden et al. (2005) um computador Pentium III 800MHz com 256MB de memória. Neste primeiro grupo o valor ótimo foi definido como alvo para a busca tabu (BT). Os nomes das instâncias são formados

pelo número de grupos da instância e seu nome na TSPLIB do qual o final representa o número de vértices da instância.

Tabela 1 – Resultados para instâncias *Center Clustering*

	Feremans B&C		Golden LS		Golden GA		BT+RC+BLI	
Instância	Ótimo	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)
15spain47	2393	5	2393	2	2393	7	2393	0,01
27europ47	13085	3	13085	4	13085	13	13085	0,00
50gr96	306	68	306	27	306	57	306	0,07
35gr137	209	110	209	19	209	39	209	0,03
34gr202	135	4558	135	28	135	59	135	1,24
10att48	10923	4	10923	1	10923	6	10923	0,00
10gr48	1282	3	1282	1	1282	6	1282	0,00
10hk48	4119	4	4119	1	4119	6	4119	0,00
11eil51	132	5	132	1	132	6	132	0,00
12brazil58	9206	14	9206	2	9206	7	9206	0,00
14st70	233	20	233	3	233	8	233	0,04
16eil76	186	47	186	3	186	9	186	0,04
16pr76	46514	37	46514	4	46514	11	46514	0,01
20gr96	221	99	221	6	221	14	221	0,08
20rat99	402	83	402	6	402	15	402	0,06
20kroa100	7982	65	7982	7	7982	15	7982	0,20
20krob100	8111	73	8111	6	8111	15	8111	0,07
20kroc100	8041	87	8041	7	8041	15	8041	0,16
20krod100	7643	183	7643	7	7643	15	7643	0,08
20kroe100	8164	66	8164	7	8164	15	8164	0,01
20rd100	2779	55	2779	7	2779	15	2779	0,01
21eil101	204	76	204	7	204	16	204	0,01
21lin105	6728	109	6728	7	6728	17	6728	0,02
22pr107	20398	244	20398	8	20398	18	20398	1,24
24gr120	2255	114	2255	11	2255	22	2255	0,02
25pr124	30174	753	30174	11	30174	23	30174	0,24
26bier127	58150	908	58150	14	58150	25	58150	0,20
28pr136	34104	406	34104	18	34104	31	34104	1,00
28gr137	329	1518	329	15	329	32	329	0,13
29pr144	40055	861	40055	18	40055	34	40055	0,98
30kroa150	9815	426	9815	23	9815	39	9815	0,65
30krob150	10048	849	10048	22	10048	39	10048	0,26
31pr152	39109	1541	39109	23	39109	41	39109	0,13
32u159	18723	592	18723	26	18723	45	18723	2,89
39rat195	751	2120	753	47*	**751**	79	**751**	17,48
40kroa200	11634	2607	11634	55	11634	86	11634	15,81
40krob200	11244	5254	11244	52	11244	84	11244	8,39
Tempo Médio		647,76		13,68		26,59		1,39

Na Tabela 1 é possível verificar os resultados obtidos pela metaheurística desenvolvida quando aplicada a 37 instâncias que utilizam o método de agrupamento *center clustering*. Nestas instâncias tanto o GA como a BT chegaram ao ótimo para todas as instâncias, sendo que a LS não alcançou o valor ótimo para a instância 39rat195. Nestas instâncias o método proposto leva em média

aproximadamente um décimo do tempo necessário pela heurística LS e pouco mais de um vigésimo do tempo do GA.

A Tabela 2 traz os resultados obtidos em instâncias do grupo 1 utilizando a técnica de agrupamento *Grid Clustering* com $\mu = 3$, nestas instâncias a heurística LS não alcançou o valor ótimo para as instâncias 60pr136 e 72kroa200, já o GA bem como a metaheurística proposta neste trabalho atingiram o valor ótimo em todos os casos. O tempo de processamento necessário a BT+RC+BLI foi cerca de oito vezes e meia inferior ao de LS e quatorze vezes e meia inferior ao do GA.

Tabela 2 – Resultados para instâncias *Grid Clustering* $\mu = 3$

	Feremans B&C		Golden LS		Golden GA		BT+RC+BLI	
Instância	Ótimo	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)
18att48	16521	2	16521	3	16521	9	16521	0,01
25eil51	242	1	242	4	242	12	242	0,00
24st70	297	5	297	6	297	14	297	0,03
36eil76	306	13	306	13	306	28	306	0,24
31pr76	58038	27	58038	11	58038	20	58038	0,12
33gr96	298	131	298	13	298	28	298	0,20
36rat99	521	22	521	18	521	34	521	0,46
43kroa100	11914	26	11914	23	11914	46	11914	0,32
44krob100	12561	51	12561	28	12561	49	12561	1,47
42kroc100	12284	24	12284	24	12284	44	12284	0,07
42krod100	11827	34	11827	25	11827	45	11827	0,19
42kroe100	12292	67	12292	25	12292	44	12292	0,07
36rd100	3978	33	3978	18	3978	34	3978	0,17
36eil101	295	59	295	18	295	35	295	1,91
42lin105	9280	136	9280	31	9280	54	9280	0,39
45pr107	23290	140	23290	22	23290	44	23290	0,05
42pr124	37837	453	37837	31	37837	57	37837	0,16
50bier127	71221	721	71221	42	71221	75	71221	0,81
60pr136	52817	340	52824	61*	**52817**	107	**52817**	5,91
49gr137	391	647	391	43	391	80	391	13,68
48pr144	43725	897	43725	39	43725	74	43725	0,43
57kroa150	14050	387	14050	81	14050	119	14050	7,76
56krob150	13845	236	13845	66	13845	109	13845	1,54
54pr152	44253	1253	44253	56	44253	99	44253	4,47
58u159	24214	658	24214	73	24214	115	24214	1,97
51rat195	1111	604	1111	190	1111	290	1111	14,10
72kroa200	14881	1920	14897	155*	**14881**	259	**14881**	71,45
76krob200	15320	2122	15320	186	15320	336	15320	26,00
Tempo Médio		393,18		46,61		80,71		5,50

Na Tabela 3 são apresentados os resultados para instâncias criadas com agrupamento *Grid Clustering* com $\mu = 5$, nestas tanto o GA quanto a LS propostas por Golden não atingiram o valor ótimo para a instância 48krob200. A metaheurística proposta neste trabalho atingiu o ótimo para todas as instâncias. O tempo de

processamento necessário pelo método proposto neste trabalho foi quatro vezes e meia inferior ao tempo de LS e sete vezes e meia inferior ao de GA. Nestas instâncias houve uma diferença no tempo computacional menor do que nas demais, porém, ainda assim trata-se de uma vantagem considerável para o método proposto.

Tabela 3 – Resultados para instâncias *Grid Clustering* $\mu = 5$

	Feremans B&C		Golden LS		Golden GA		BT+RC+BLI	
Instância	Ótimo	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)
13att48	13189	5	13189	2	13189	6	13189	0,01
16eil51	158	6	158	2	158	8	158	0,03
16st70	214	13	214	3	214	9	214	0,00
16eil76	149	16	149	3	149	9	149	0,05
16pr76	29788	22	29788	3	29788	9	29788	0,03
22gr96	234	84	234	7	234	16	234	0,03
25rat99	410	50	410	9	410	19	410	0,04
23kroa100	8054	82	8054	9	8054	17	8054	0,07
25krob100	7880	40	7880	11	7880	19	7880	0,04
25kroc100	8084	80	8084	11	8084	20	8084	0,13
24krod100	8741	59	8741	9	8741	18	8741	0,65
25kroe100	8401	52	8401	9	8401	19	8401	0,05
24rd100	3077	38	3077	8	3077	18	3077	0,01
25eil101	217	73	217	10	217	19	217	0,11
30lin105	7410	99	7410	14	7410	27	7410	0,28
22pr107	19877	403	19877	8	19877	16	19877	0,01
25pr124	27156	220	27156	12	27156	22	27156	0,03
26bier127	58989	689	58989	14	58989	23	58989	0,06
34pr136	37735	198	37735	25	37735	40	37735	1,30
32gr137	338	1038	338	18	338	35	338	0,53
30pr144	36279	778	36279	18	36279	33	36279	0,08
36kroa150	10101	387	10101	35	10101	50	10101	1,00
36krob150	9780	527	9780	31	9780	50	9780	2,69
33pr152	38143	1294	38143	25	38143	42	38143	0,07
32u159	17059	391	17059	25	17059	40	17059	0,35
49rat195	796	1581	796	77	796	114	796	21,43
47kroa200	11628	968	11628	78	11628	108	11628	10,95
48krob200	11113	1262	11115	78*	11115	114*	**11113**	82,25
Tempo Médio		373,39		19,79		32,86		4,37

A Tabela 4 apresenta os resultados para instâncias do grupo 1 desenvolvidas a partir do método de agrupamento *Grid Clustering* $\mu = 7$. LS não atingiu o ótimo para a instância 36rat195, enquanto GA e BT+RC+BLI, proposta neste trabalho, atingiram o valor ótimo para todas as instâncias. Em relação ao tempo de processamento é possível observar uma vantagem considerável do método proposto em relação aos demais usados para comparação. Neste grupo de instâncias o método proposto BT+RC+BLI chegou ao valor ótimo para todas as instâncias em um tempo computacional aproximadamente vinte e oito vezes e meia menor do que o

tempo necessário por LS. Já em relação ao GA a BT+RC+BLI foi cerca de cinqüenta e duas vezes mais rápida.

Tabela 4 – Resultados para instâncias *Grid Clustering* $\mu = 7$

	Feremans B&C		Golden LS		Golden GA		BT+RC+BLI	
Instância	Ótimo	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)
7att48	6667	1	6667	1	6667	5	6667	0,00
9eil51	100	4	100	1	100	6	100	0,00
16st70	214	13	214	3	214	9	214	0,01
16eil76	149	16	149	3	149	9	149	0,05
16pr76	29788	22	29788	3	29788	9	29788	0,01
15gr96	186	85	186	3	186	9	186	0,03
16rat99	308	66	308	4	308	11	308	0,24
16kroa100	5987	86	5987	5	5987	11	5987	0,06
16krob100	6058	59	6058	5	6058	11	6058	0,03
16kroc100	5534	47	5534	5	5534	11	5534	0,00
16krod100	5904	87	5904	5	5904	11	5904	0,06
16kroe100	6450	32	6450	5	6450	11	6450	0,00
16rd100	2287	32	2287	4	2287	11	2287	0,04
16eil101	141	64	141	4	141	11	141	0,03
16lin105	4542	113	4542	5	4542	11	4542	0,10
16pr107	17547	435	17547	5	17547	11	17547	0,12
19pr124	23164	101	23164	7	23164	15	23164	0,07
19bier127	52097	2563	52097	7	52097	15	52097	0,15
20pr136	22541	504	22541	8	22541	17	22541	0,13
22gr137	264	956	264	9	264	20	264	0,37
21pr144	33947	365	33947	9	33947	19	33947	0,02
25kroa150	7944	390	7944	15	7944	27	7944	1,67
25krob150	7293	480	7293	17	7293	28	7293	0,12
24pr152	35429	766	35429	14	35429	25	35429	0,03
23u159	12659	161	12659	14	12659	23	12659	0,06
36rat195	639	1439	648	44*	**639**	66	**639**	3,59
35kroa200	9640	1677	9640	45	9640	63	9640	0,21
36krob200	9742	1007	9742	45	9742	67	9742	3,27
Tempo Médio		413,25		10,54		19,36		0,37

Na Tabela 5 apresentam-se os resultados obtidos em instâncias do tipo *Grid Clustering* $\mu = 10$ do grupo 1, onde todas as heurísticas obtiveram como resultado final os valores ótimos para todas as instâncias. O método proposto consegue alcançar o valor ótimo sessenta e uma vezes e meia mais rápido do que LS e cento e vinte e nove vezes mais rápido do que GA.

O método proposto foi o único entre os utilizados na comparação a atingir o valor ótimo para todas as 150 instâncias. Embora o *hardware* utilizado em Golden et. al (2005) seja diferente do utilizado neste trabalho acredita-se que a diferença de tempo computacional seja relevante entre os métodos. Outro dado importante é que

em todas as instâncias nas quais os métodos chegaram ao valor ótimo BT+RC+BLI foi mais rápida.

Tabela 5 – Resultados para instâncias *Grid Clustering* $\mu = 10$

	Feremans B&C		Golden LS		Golden GA		BT+RC+BLI	
Instância	Ótimo	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)
7att48	6667	1	6667	1	6667	5	6667	0,00
9eil51	100	4	100	1	100	5	100	0,00
9st70	147	8	147	1	147	6	147	0,00
9eil76	94	13	94	1	94	6	94	0,01
9pr76	20501	8	20501	1	20501	6	20501	0,00
15gr96	186	85	186	3	186	9	186	0,03
16rat99	308	66	308	4	308	11	308	0,22
16kroa100	5987	86	5987	4	5987	11	5987	0,08
16krob100	6058	59	6058	5	6058	11	6058	0,06
16kroc100	5534	47	5534	5	5534	11	5534	0,00
16krod100	5904	86	5904	5	5904	11	5904	0,05
16kroe100	6450	32	6450	5	6450	11	6450	0,00
16rd100	2287	32	2287	4	2287	11	2287	0,04
16eil101	141	64	141	4	141	11	141	0,03
16lin105	4542	113	4542	5	4542	11	4542	0,12
12pr107	16754	556	16754	3	16754	8	16754	0,15
14pr124	18554	145	18554	4	18554	10	18554	0,00
14bier127	43778	438	43778	4	43778	10	43778	0,05
16pr136	21732	292	21732	5	21732	13	21732	0,07
15gr137	197	410	197	4	197	12	197	0,11
16pr144	32510	42	32510	5	32510	13	32510	0,01
16kroa150	5229	512	5229	7	5229	14	5229	0,06
16krob150	5494	291	5494	7	5494	14	5494	0,09
16pr152	33340	1781	33340	6	33340	14	33340	0,02
23u159	12659	160	12659	14	12659	23	12659	0,06
25rat195	482	1372	482	20	482	34	482	0,79
25kroa200	6895	1263	6895	23	6895	35	6895	0,66
25krob200	6922	1181	6922	24	6922	36	6922	0,47
27pr226	43389	3929	43389	22	43389	40	43389	0,08
Tempo Médio		450,90		6,79		14,21		0,11

4.2 Resultados com Instâncias do Grupo 2

Com base nos bons resultados obtidos na primeira fase de testes, na qual o algoritmo aqui proposto encontrou o valor ótimo para todas as instâncias do primeiro grupo em um tempo computacional inferior a outras heurísticas propostas na literatura, deu-se início a uma segunda fase de testes na qual foram utilizadas as instâncias do segundo grupo que não possuem valor ótimo conhecido.

O segundo grupo de testes é composto por 102 instâncias de grande porte para o PAGMG todas utilizadas nos testes de Öncan et al. (2007). Com a finalidade de encontrar o ajuste de parâmetros mais adequado para as heurísticas aqui desenvolvidas para processar estas instâncias fez-se diversos testes utilizando um grupo de 19 instâncias com valor ótimo não conhecido. Como critério de parada para estes testes foi definido o tempo limite de execução em 10800 segundos, ou seja, 3 horas. Os resultados destes testes são apresentados na Tabela 6. Os valores indicados na coluna mais a direita da tabela são relativos aos tempos necessários para que a heurística chegasse a melhor solução encontrada. Na instância 45ts225 foi possível chegar um custo inferior ao encontrado por Öncan et. al (2007), nas demais os custos são idênticos. Por este motivo, a média dos resultados obtidos foi ligeiramente favorável a BT+RC+BLI. Os testes de Öncan et. al (2007) foram realizados em um computador Pentium IV 3GHz com 2GB de memória.

Tabela 6 – Resultados para instâncias de porte médio

	Öncan TS		BT+RC+BLI	
Instância	Custo	Tempo (s)	Custo	Tempo (s)
Center Clustering				
40d198	7044	31	7044	16,42
41gr202	242	33	242	19,64
45ts225	62269	37	**62268**	1299,30
46pr226	55515	39	55515	1,59
Grid Clustering $\mu = 3$				
67d198	8283	58	8283	145,96
68gr202	293	64	293	36,76
75ts225	79019	70	79019	139,16
84pr226	62527	84	62527	40,96
Grid Clustering $\mu = 5$				
40d198	7098	31	7098	9,16
41gr202	232	34	232	17,42
45ts225	60588	37	60588	792,47
50pr226	56721	43	56721	5,87
Grid Clustering $\mu = 7$				
32d198	6501	23	6501	1,41
31gr202	203	23	203	3,22
35ts225	50813	26	50813	5,91
33pr226	48249	26	48249	0,65
Grid Clustering $\mu = 10$				
25d198	6185	17	6185	1,13
21gr202	177	14	177	1,31
25ts225	40339	18	40339	2,34
Média	29068,31		**29068,26**	

Após encontrar o ajuste mais adequado para os parâmetros das heurísticas executou-se os testes com as 102 instâncias do grupo 2. Na Tabela 7 são apresentados os resultados para as instâncias que utilizam a técnica de agrupamento *center clustering*. Para a instância 115u574, em duas das cinco rodadas consecutivas, o método proposto superou o resultado obtido por Oncan et. al (2007) atingindo os valores de 15031 e 15036. Na instância 157rat783 em dois dos cinco testes chegou-se ao valor de 3012. Nas instâncias 88pr439, 131p654, 134gr666 e 145u724 a BT+RC+BLI superou os valores encontrados em Öncan et al. (2007). Nas demais instâncias ambas heurísticas alcançaram os mesmos resultados. Na média o método proposto obteve resultados ligeiramente superiores aos encontrados na literatura para estas instâncias.

Tabela 7 – Resultados para instâncias *Center Clustering*

	Öncan TS		BT+RC+BLI	
Instância	Custo	Tempo (s)	Custo	Tempo (s)
53gil262	887	74	887	16,82
53pr264	21872	72	21872	18,16
60pr299	20290	94	20290	39,91
64lin318	18471	130	18471	149,00
80rd400	5868	208	5868	145,96
84fl417	7935	233	7935	145,71
87gr431	86885	233	86885	124,98
88pr439	51760	574	**51749**	213,17
89pcb442	19571	266	19571	418,46
99d493	16493	587	16493	284,54
107att532	12001	597	12001	496,48
107ali535	11403	683	114303	1124,01
107si535	12791	573	12791	210,16
113pa561	864	702	864	435,01
115u574	**15037**	517	15042	9742,40
115rat575	2170	762	2170	2765,23
131p654	22208	1045	**22207**	2840,93
132d657	19427	1056	19427	4019,19
134gr666	144756	1365	**144737**	1519,96
145u724	15905	1290	**15904**	4327,29
157rat783	**3017**	1916	3024,8	9666,68
Média	29167,19		**29166,28**	

Em Öncan et. al (2007) os autores executaram seus testes também com a instância 46gr229 obtendo o custo final de 59740. Nas instâncias fornecidas pelos autores não se encontra disponível o conjunto de dados correspondente ao agrupamento *Center Clustering* para esta instância. Utilizando um arquivo de agrupamento *Center Clustering* criado para esta instância desenvolvido por Feremans et al. (2002) chegou-se ao custo de 60241 para esta instância. Este custo tão diferente do encontrado em Öncan et. al (2007) indica que provavelmente há

uma discrepância entre o agrupamento feito por estes autores e o feito por Feremans et. al (2002) e utilizando também por Wang et. al (2006). Utilizando a tabela de custos das arestas e de agrupamentos disponibilizada por Feremans et. al (2002), Wang et. al (2006) chega ao custo de 787 para esta instância, este mesmo custo também foi alcançado pela heurística desenvolvida neste trabalho para este conjunto de dados.

Tabela 8 – Resultados para instâncias *Grid Clustering* $\mu = 3$

	Öncan TS		BT+RC+BLI	
Instância	Custo	Tempo (s)	Custo	Tempo (s)
81gr229	74792	67	74792	7,03
95gil262	1255	204	1255	96,60
101pr264	29199	102	29199	14,60
102pr299	23096	399	23096	128,57
108lin318	24092	206	24092	161,87
135rd400	7632	548	7632	1108,21
142fl417	**8353**	681	8354,2	7616,00
149gr431	103844	1697	103844	977,19
163pr439	64953	932	64953	6768,21
156pcb442	**27513**	810	27549,6	9164,20
171d493	**20269**	1104	20281,2	7650,60
182att532	**15652**	1569	15659,4	6696,87
181ali535	**134362**	1576	134372,6	7731,50
198u574	**19699**	1340	19741,6	11133,43
196rat575	**2884**	2080	2896,8	11025,10
230p654	23553	4127	**23550,6**	11928,85
221d657	**25703**	3027	25742	11272,04
224gr666	**174671**	3105	174688,6	10599,70
266u724	**21761**	5201	21794,4	13350,88
285rat783	**4211**	7180	4247,6	10816,81
Média	**40374,7**		40387,08	

A Tabela 8 apresenta os resultados para 20 instâncias do grupo 2, as quais foram criadas pelo método de agrupamento *Grid Clustering* $\mu = 3$. Na instância 142fl417 embora o algoritmo desenvolvido não tenha conseguido superar os resultados de Öncan et al. (2007) na média das cinco execuções por duas vezes este resultado foi igualado. Em 156pcb442 foi possível igualar o resultado em uma das execuções. Na 171d493 o resultado foi igualado em dois dos testes. Na instância 182att532 por três vezes o resultado foi igualado. Na 181ali535 em quatro dos cinco testes obteve-se o mesmo resultado de TS. Nas instâncias 198u574, 196rat575, 221d657 e 285rat783 não foi possível superar TS. Em 224gr666, em um dos testes atingiu-se o valor 174655. Na 266u724 dois testes obtiveram resultados melhores do que TS atingindo os valores 21755 e 21751. Na instância 230p654 foi

possível encontrar um custo menor do que o obtido por Öncan et. al (2007), sendo que por três vezes obteve-se o valor de 23549. Apesar de em média TS ter superado BT+RC+BLI pontualmente em algum dos cinco testes realizados o método proposto consegue igualar ou superar TS na maioria das instâncias.

Neste caso em particular, apresentado na Tabela 9, nas instâncias criadas por *Grid Clustering* $\mu = 5$ do grupo 2 a metaheurística proposta neste trabalho mostrou-se bastante vantajosa alcançando melhores resultados de função objetivo para as instâncias 95pcb442, 127u574, 121rat575, 134p654, 137d657 e 166u724, empatando nas demais instâncias e sendo superada somente na instância 169rat783 pela TS de Öncan et al. (2007). Na avaliação do custo médio o método proposto superou TS para estas instâncias.

Tabela 9 – Resultados para instâncias *Grid Clustering* $\mu = 5$

	Öncan TS		BT+RC+BLI	
Instância	Custo	Tempo (s)	Custo	Tempo (s)
47gr229	54305	43	54305	59,90
63gil262	984	63	984	1741,54
55pr264	21351	67	21351	21,27
69pr299	18582	105	18582	1381,04
64lin318	17667	116	17667	50,66
81rd400	5434	284	5434	1782,88
93fl417	7952	230	7952	882,18
87gr431	81856	244	81856	2241,75
95pcb442	19411	357	**19383**	4208,96
96pr439	54230	587	54230	1745,45
102d493	16132	725	16132	1031,91
110att532	11896	641	11896	1461,84
108ali535	108201	632	108201	877,96
127u574	15255	1037	**15252**	3487,03
121rat575	2018	705	**2014**	4655,01
134p654	22379	1316	**22377**	887,44
137d657	19846	1249	**19826**	2555,78
139gr666	144077	2192	144077	2319,92
166u724	15458	2064	**15435**	7884,70
169rat783	**2832**	2424	2842,4	11079,94
Média	31993,3		**31989,82**	

A Tabela 10 apresenta os resultados obtidos com instâncias do tipo *Grid Clustering* $\mu = 7$ do grupo 2. Na instância 80att532 foi possível encontrar um custo inferior ao de TS. Na 119u724 em uma rodada chegou-se ao valor de 13086 estabelecendo um novo limitante superior para a instância. Em 121rat783 o valor 2229 foi encontrado em duas rodadas. Nas demais instâncias obteve-se custos

iguais aos encontrados em Öncan et al. (2007). Na média os dois métodos estão muito próximos com TS levando uma pequena vantagem.

Tabela 10 – Resultados para instâncias *Grid Clustering* $\mu = 7$

	Öncan TS		BT+RC+BLI	
Instância	Custo	Tempo (s)	Custo	Tempo (s)
34gr229	46049	13	46049	5,31
49gil262	802	63	802	654,55
43pr264	20438	28	20438	62,68
47pr299	15238	37	15238	99,91
49lin318	14909	44	14909	120,52
64rd400	4581	97	4581	949,74
61fl417	7446	99	7446	75,96
64gr431	71415	111	71415	447,91
74pr439	47101	140	47101	261,30
64pcb442	14644	117	14644	772,03
78d493	14152	212	14152	929,36
80att532	10206	992	**10204**	2370,43
83ali535	94149	1012	94149	1379,58
92u574	12186	304	12186	3881,31
4100rat575	1735	601	1735	3122,35
96d657	16542	440	16542	950,48
96gr666	119271	452	119271	2949,56
100p654	21770	462	21770	532,08
119u724	**13101**	710	13147,6	10445,89
121rat783	**2230**	850	2234,8	10803,38
Média	**27398,25**		27400,72	

Tabela 11 – Resultados para instâncias *Grid Clustering* $\mu = 10$

	Öncan TS		BT+RC+BLI	
Instância	Custo	Tempo (s)	Custo	Tempo (s)
23gr229	39793	28	39793	7,98
36gil262	639	33	639	6,84
27pr264	16546	28	16546	2,44
35pr299	11624	63	11624	10,07
36lin318	10119	31	10119	1,10
49rd400	3825	99	3825	150,55
42fl417	6986	75	6986	24,49
52gr431	62722	110	62722	51,58
48pr439	40518	117	40518	79,08
48pcb442	11941	135	11941	289,95
52d493	11121	180	11121	203,82
57att532	8497	187	8497	493,45
57ali535	73359	213	73359	242,66
64u574	9755	234	9755	383,23
64rat575	1235	284	1235	1846,99
69p654	20739	274	20739	18,57
73d657	13495	386	13495	1910,22
70gr666	97198	315	97198	970,20
80u724	9608	365	9608	881,44
81rat783	1682	468	1682	2968,15
Média	22570,1		22570,1	

Na Tabela 11 são apresentados os resultados para instâncias *Grid Clustering* $\mu = 10$ do grupo 2 onde não houve diferença entre a heurística proposta e o trabalho de Öncan et al. (2007) no que se refere ao custo da melhor solução encontrada pela metaheurística.

Com base nos resultados apresentados até aqui envolvendo 271 instâncias de pequeno, médio e grande porte para o PAGMG, é possível afirmar que o método proposto demonstra robustez e eficácia na solução deste problema.

4.3 Resultados com Instâncias para o PM-PAGMG

A Tabela 12 apresenta os resultados obtidos pela metaheurística proposta em sua versão adaptada para o PM-PAGMG. Esta versão adaptada conta apenas com um método adicional para inserção de vértices adicionais em uma solução contendo um vértice de cada grupo. Não foram necessários ajustes nos parâmetros da metaheurística, sendo aplicados os mesmos definidos para o PAGMG. Os resultados apresentados são comparados com Öncan et al. (2007) e Haouari et al. (2006) os quais possuem os melhores resultados até o momento para as instâncias testadas. As 20 instâncias utilizadas nos testes possuem valor ótimo já relatado em Feremans et al. (2001), sendo este utilizado como alvo para a busca tabu, além de um tempo limite de execução de 1200 segundos, considerado suficiente para que a heurística chegasse a resultados satisfatórios.

A metaheurística proposta se destaca nas instâncias dhc16, dhc18 e dhc19 nas quais supera o GA de Haouari e a TS de Öncan, nas instâncias dhc17 e dhc20 obteve-se resultados inferiores aos das outras heurísticas, e nas demais instâncias todas as heurísticas obtiveram os mesmos resultados variando apenas em relação ao tempo de execução necessário para atingir tal valor.

Pode-se observar que na média as três heurísticas possuem resultados bastante próximos uma da outra e também em relação ao ótimo. Um aspecto interessante a respeito da heurística desenvolvida neste trabalho é que para 18 das 20 instâncias deste problema ela atinge o valor ótimo em pelo menos uma das cinco execuções.

Tabela 12 – Resultados para instâncias do PM-PAGMG

	Feremans	BT+RC+BLI		Haouari GA		Öncan TS	
Instância	Ótimo	Custo	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)
dhc1	23	23	0,00	23	0,02	23	0,08
dhc2	41	41	0,01	41	0,01	41	0,06
dhc3	36	36	0,01	36	0,03	36	0,1
dhc4	18	18	0,04	18	0,01	18	0,6
dhc5	27	27	0,06	27	0,11	27	0,41
dhc6	55	55	0,01	55	0,12	55	0,48
dhc7	67	67	0,43	67	0,2	67	0,59
dhc8	53	53	0,95	53	0,3	53	0,68
dhc9	37	37	1,21	37	0,26	37	0,63
dhc10	48	48	3,51	48	1,64	48	2,43
dhc11	50	50	5,06	50	0,46	50	1,01
dhc12	68	68	53,75	68	3,24	68	7,03
dhc13	44	44	94,31	44	2,36	44	5,38
dhc14	50	50	18,51	50	7,13	50	16,9
dhc15	60	60	0,03	60	8,25	60	19,6
dhc16	117	**117**	547,17	118	45	118	134
dhc17	88	91	1201,23	**89**	115	**89**	269
dhc18	85	**85,2**	428,60	86	246	86	395
dhc19	88	**90**	1117,91	91	331	94	594
dhc20	105	127,6	1242,29	**120**	379	124	617
	58	59,39	235,76	**59,05**	57,01	59,4	103,249

5 CONCLUSÃO

Atendendo aos objetivos propostos para este trabalho inicialmente fez-se um estudo de diversas publicações a respeito do PAGMG e do PM-PAGMG, muitas delas citadas ao longo do trabalho e relacionadas nas referências bibliográficas deste texto. Num segundo momento foi definida a linguagem Java para implementação dos algoritmos por ser orientada a objetos, amplamente difundida no meio acadêmico, multiplataforma e possuir uma vasta quantidade de materiais de apoio disponíveis para os desenvolvedores. Selecionada a linguagem foi necessário escolher um ambiente de desenvolvimento entre os muitos disponíveis para Java como NetBeans, Eclipse, J Builder entre outros, o ambiente NetBeans foi selecionado por ser mais amigável e contar com um grande número de ferramentas adicionais para o desenvolvedor. Encerrada esta primeira etapa de estudos e seleção de ferramentas, passou-se então para o desenvolvimento.

Inicialmente seguiu-se duas linhas de desenvolvimento implementando uma metaheurística Busca Tabu e um Algoritmo Genético, sendo, conforme descrito ao longo do trabalho, a primeira a apresentar os melhores resultados e ser selecionada para a continuidade dos trabalhos. Para alcançar a evolução necessária na qualidade dos resultados foram desenvolvidas cerca de trinta versões da busca tabu aprimorando e corrigindo os códigos até chegar aos resultados finais. A cada nova versão eram realizados dezenas de testes para verificar a eficácia dos algoritmos. À medida que o desenvolvimento avançava novas características foram incorporadas a busca tabu.

Uma das primeiras heurísticas adicionais a ser incorporada a busca tabu foi a de reconexão de caminhos que possibilitou uma grande melhoria nos resultados. Posteriormente foram incorporadas questões de restrição de vizinhança, penalidades e conjunto de soluções de elite entre outras funcionalidades de menor importância. Por último a busca tabu foi incrementada ainda com o procedimento de busca local iterativa que possibilitou uma melhoria em algumas instâncias onde a busca tabu ainda estava com dificuldades para atingir os melhores resultados encontrados na literatura.

Durante a realização deste trabalho pode-se testar o funcionamento conjunto das técnicas de busca tabu, reconexão de caminhos e busca local iterativa para solução do problema da árvore geradora mínima generalizado e sua variante com pelo menos um vértice por grupo. Para ambos os problemas as heurísticas propostas obtiveram resultados equivalentes aos melhores encontrados na literatura para um conjunto muito amplo de instâncias de teste. Este fato comprova a robustez do método proposto neste trabalho o qual obteve resultados muito bons para 271 instâncias do PAGMG e 20 instâncias do PM-PAGMG.

Como sugestões para futuros trabalhos pode-se aprimorar os algoritmos propostos tornando-os mais ágeis, de modo que, possam atingir os valores desejados, dentro do tempo de processamento estabelecido, nas instâncias onde isto não foi possível, além de gerar novas instâncias para o PM-PAGMG.

Acredita-se que com mais algum tempo de estudo e dedicação seja possível em um trabalho futuro atingir o ótimo para todas as instâncias do PM-PAGMG. Neste trabalho das vinte instâncias testadas o ótimo foi alcançado em dezesseis, e nas quatro restantes foram obtidos resultados compatíveis com os melhores encontrados na literatura.

REFERÊNCIAS BIBLIOGRÁFICAS

CORMEN, T. H.;LEISERSON, C. E.;RIVEST, R.L.; STEIN, C. **Introduction to Algorithms.** 2. ed. Cambridge, Massachusetts: The MIT Press, 2003, 1180p.

CRISTO, F.; MÜLLER, F. M.; FERREIRA, L.; BORENSTEIN, D. **A Tabu Search Algorithm for the Generalized Minimum Spanning Tree Problem.** Anais do XIII International Conference on Industrial Engineering and Operations Management, Foz do Iguaçu: ABEPRO, 2007a, 9p. Disponível em: <http://www.abepro.org.br/biblioteca/ENEGEP2007_TI620461_9431.pdf>. Acesso em: 23 out. 2007.

CRISTO, F.; MÜLLER, F. M.; FERREIRA, L.; BORENSTEIN, D. **Algoritmo de Busca Tabu para o Problema da Árvore Geradora Mínima Generalizado.** Anais do XXVII Encontro Nacional de Engenharia de Produção, Foz do Iguaçu: ABEPRO, 2007b, 9p. Disponível em: <http://www.abepro.org.br/biblioteca/ ENEGEP2007_TR620461_9472.pdf>. Acesso em: 23 out. 2007.

CRISTO, F.; MÜLLER, F. M.; FERREIRA, L.; BORENSTEIN, D.; LIMA, P. R. D. B.; BAYER, F. M.; FRICK M.; ROSA, M. **Metaheurísticas para a solução do Problema da Árvore de Cobertura Mínima Generalizado.** Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional, Fortaleza: SOBRAPO, 2007c, 11p. Disponível em: <http://www.cafw.ufsm.br/~fernando/Cristo-SBPO_2007 .pdf>. Acesso em: 23 out. 2007.

DROR, M.; HAOUARI, M.; CHAOUACHI, J. **Generalized Spanning Trees.** European Journal of Operational Research, 120,p. 583–592, 2000.

FEREMANS, C. **Generalized Spanning Trees and Extensions.** 2002. 162f. Tese (Doutorado em Orientação em Pesquisa Operacional) – Universidade Livre de Bruxelas, Bruxelas, 2002.

FEREMANS, C.; LABBÉ, M.; LAPORTE, G. **On generalized minimum spanning trees.** European Journal of Operational Research, 134, p. 457-458, 2001.

FERREIRA, C. M. S.; OCHI, L. S.; MACAMBIRA, E.M. **GRASP com Memória Adaptativa para o Problema da Árvore de Cobertura Mínima Generalizado.** Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional, Goiânia: SOBRAPO, 2006, 12p. Disponível em: <http://www.ic.uff.br/~satoru/conteudo/ artigos/SBPO2006-Cristiane.pdf>. Acesso em: 29 set. 2006.

FERREIRA, C. M. S.; OCHI, L. S. **Metaheurística GRASP com Memória Adaptativa para a solução do Problema da Árvore Geradora Mínima Generalizado**. In: VI Encontro Nacional de Inteligência Artificial (VI ENIA) - em conjunto com SBC2007, 2007, Rio de Janeiro. Anais do VI ENIA/SBC2007. RJ : SBC, 2007. v. 1. p. 1202-1211.

FERREIRA, L.; CRISTO, F.; MÜLLER, F. M.; BORENSTEIN, D. **Algoritmo Genético para o Problema da Árvore Geradora Mínima Generalizado.** Anais

do XXVII Encontro Nacional de Engenharia de Produção, Foz do Iguaçu: ABEPRO, 2007, 10p. Disponível em: <http://www.abepro.org.br/biblioteca/ENEGEP2007_TR620461_9134.pdf>. Acesso em: 23 out. 2007.

FISCHETTI, M.; SALAZAR, J.J.; TOTH, P. **The Symmetric Generalized Travelling Salesman Polytope.** Networks, 26, p. 113-123, 1995.

GHOSH, D. **Solving medium to large sized Euclidean generalized minimum spanning tree problems.** Technical Report NEP-CMP-2003-09-28, Indian Institute of Management, Research and Publication Department, Ahmedabad, India, 2003. 15p.

GLOVER, F. **Future paths for integer programming and links to artificial intelligence.** Computers and Operations Research, v. 13, p. 533-549, 1986.

GLOVER, F.; LAGUNA, M.; MARTÍ, R. **Fundamentals of scatter search and path relinking**. Control and Cybernetics, 29, p. 653-684, 2000.

GOLDEN, B.; RAGHAVAN, S.; STANOJEVIC, D. **Heuristic search for the generalized minimum spanning tree.** INFORMS Journal on Computing, 17, p. 290–304, 2005.

HAOUARI, M.; CHAOUACHI, J.S. **Upper and Lower Bounding Strategies for the Generalized Minimum Spanning Tree Problem.** European Journal of Operational Research, 172, p. 632-647, 2006.

IHLER, E.; REICH, G.; WIDMAYER, P. **Class Steiner Trees and VLSI-design.** Discrete Applied Mathematics, 90, p. 173–194, 1999.

KRUSKAL, J. B. **On the shortest spanning subtree and the traveling salesman problem.** Proceedings of the American Mathematical Society, 7, p. 48–50, 1956.

LOURENÇO, H. R.; MARTIN, O.; STÜTZLE, T. **A Beginner's Introduction to Iterated Local Search**. Proceedings of MIC 2001, p. 1–6, 2001.

LOURENÇO, H. R.; MARTIN, O.; STÜTZLE, T. **Iterated Local Search**. Handbook of Metaheuristics,. Kluwer Academic Publishers, p. 321 – 353, 2002.

MYUNG, Y. S.; LEE, C. H.; TCHA, D. W. **On the Generalized Minimum Spanning Tree Problem.** Networks, 26, p. 231-241, 1995.

ONCAN, T.; CORDEAU J. F.; LAPORTE, G. **A tabu search heuristic for the generalized minimum spanning tree problem.** European Journal of Operational Research, 2007, 25p.

PRIM, R. C. **Shortest connection networks and some generalizations.** Bell System Technical, Journal 36, p. 1389–1401, 1957.

REEVES, C. R. **Modern Heuristic Techniques for Combinatorial Problems**. New York: John Wiley & Sons, Inc, 1993, 320p.

REINELT, G. **TSPLIB – A Traveling Salesman Problem Library**. INFORMS Journal on Computing, v. 3, p. 376-384, 1991

ROCHA, M. L.; ALVARENGA, F. V. **Uma Metaheurística GRASP para o Problema da Árvore Geradora de Custo Mínimo com Grupamentos.** Anais do VII Encontro de Estudantes de Informática do Estado do Tocantins, Palmas: ULBRA, 2005, 4p. Disponível em: < http://www.ulbra-to.br/ensino/43020/artigos/anais2005/anais/GRASP.pdf>. Acesso em: 13 ago. 2007.

WANG, Z.; CHE, C.H.; LIM, A. **Tabu Search for Generalized Minimum Spanning Tree Problem**. Trends in Artificial Intelligence, p. 918-922, 2006.